When Blockchain Meets AI: Optimal Mining Strategy Achieved By Machine Learning

Taotao Wang, Soung Chang Liew, and Shengli Zhang

***Abstract*—This work applies reinforcement learning (RL) from the AI machine learning field to derive an optimal Bitcoin-like blockchain mining strategy. A salient feature of the RL learning framework is that an optimal (or near optimal) strategy can be obtained without the knowing the details of the blockchain network model. Previously, the most profitable mining strategy was believed to be honest mining encoded in the default blockchain protocol. It was shown later that it is possible to gain more mining rewards by deviating from honest mining. In particular, the mining problem can be formulated as a Markov Decision Process (MDP) which can be solved to give the optimal mining strategy. However, solving the mining MDP requires knowing the values of various parameters that characterize the blockchain network model. In real blockchain networks, these parameter values are not easy to obtain and may change over time. This hinders the use of the MDP model-based solution. In this work, we employ RL to dynamically learn a mining strategy with performance approaching that of the optimal mining strategy. Since the mining MDP problem has a non-linear objective function (rather than linear functions of standard MDP problems), we design a new multi-dimensional RL algorithm to solve the problem. Experimental results indicate that, without knowing the parameter values of the mining MDP model, our multi-dimensional RL mining algorithm can still achieve optimal performance over time-varying blockchain networks.**

***Index Terms*—Blockchain, Proof-of-work, Selfish Mining, MDP, Reinforcement Learning.**

I. Introduction

THE early digital cryptocurrencies rely on central authorities to settle transactions. Digital cryptocurrencies did not flourish, until the advent of Bitcoin [1], [2]. To avoid single points of failure, Bitcoin is designed as a decentralized system without a central authority that could be compromised by corruption and attacks [1]. Since the birth of Bitcoin in 2008, it has become a widely accepted currency all over the world. In early 2018, the market price of Bitcoin went as high as 20,000 US dollars, reflecting robust demands and enthusiasm for Bitcoin by the public.

The security of Bitcoin is built on the foundation technology of blockchain. Blockchain contains several key technical components, including its chained data structure, peer-to-peer network protocol, and distributed consensus algorithm [3]–[5]. Blockchain has become a cutting-edge technology in FinTech [6], Internet of Things (IoT) [7], [8], and supply chains [9]. The Bitcoin's blockchain is not controlled by a central authority; it is assembled by peers in the network independently in a distributed manner. In order that the blockchains maintained by different peers are consistent, the peers must agree on a single universal truth about the transactions of Bitcoin through a consensus-building process.

Consensus in the Bitcoin network is achieved by the proof-of-work (PoW) consensus algorithm. The idea of PoW originated in [10] and is rediscovered and exploited in the implementation of Bitcoin. PoW provides strong probabilistic consensus guarantee with resilience against up to 1/2 malicious nodes [11], [12]. The successful operation of Bitcoin demonstrates the practicality of using PoW to achieve consensus. Subsequent to Bitcoin, many other cryptocurrencies, such as Litecoin [13], Ethereum [14], also adopt the PoW consensus algorithm.

Peers running the PoW consensus algorithm are miners who compete to solve a difficult cryptographic hash puzzle, called the PoW problem. The miner who successfully solves the PoW problem obtains the right to extend the blockchain with a block consisting of valid transactions. In doing so, the miner receives a reward in the form of a newly minted coin written into the added block. Solving the PoW problem for rewards is called mining, just like mining for precious metals.

Miners commit computation resources to solve the PoW problem. Previously, it was believed that the most profitable mining strategy is honest mining, wherein a miner will broadcast the newly added block as soon as it has solved the PoW problem. Let α be the ratio of a particular miner's computing power over the computing powers of all miners. This ratio is also the probability that the miner can solve the PoW problem before others in each round of an added block [3]. Over the long term, the rewards to a miner that executes the honest mining strategy are therefore α fraction of the total rewards issued by the Bitcoin network. This is reasonable since miners share the pie in proportion to their investments. Not known were whether there are other mining strategies more profitable than honest mining.

Later, the authors of [15] developed a selfish mining strategy that can earn higher rewards than honest mining. A selfish miner does not broadcast its mined block immediately; it carries out a block-withholding attack by secretly linking its future mined blocks to the withheld mined block. If the selfish miner can mine two successive blocks before other miners do, it can broadcast its two blocks at the same time to override the block mined by others. Since Bitcoin has an inherent

T. Wang and S. Zhang are the College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: ttwang@szu.edu.cn; zsl@szu.edu.cn).

S. Liew is with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China (e-mail: soung@ie.cuhk.edu.hk)

This research was funded by the National Key R&D Program of China (2018YFB2100705) and the Natural Science Fund of Guangdong Province (2020A1515010708).

self-adjusting mechanism to ensure that on average only one block is added to the blockchain every 10 minutes [16], by invalidating the blocks of others (hence, removing them from the blockchain), the selfish miner can increase its own profits. For example, with computing power ratio $\alpha = 1/4$, the rewards obtained by selfish mining can be up to $1/3$ fraction of the total rewards [15]. Based on this observation, [17] further proposed various selfish mining strategies with even higher rewards. Despite the many versions of selfish mining, the optimal (i.e., most-profitable) mining strategy remained elusive until [18].

The authors of [18] formulated the mining problem as a general Markov Decision Process (MDP) with a large state-action space. The objective of the mining MDP, however, is not a linear function of the rewards as in standard MDPs. Thus, the mining MDP cannot be solved using a standard MDP solver. To solve the problem, [18] first transformed the mining MDP with the non-linear objective to a family of MDPs with linear objectives, and then employed a standard MDP solver over the family of MDPs to iteratively search for the optimal mining strategy.

The approach in [18] is model-based in that various parameter values (e.g., α) must be known before the MDP can be set up. In real blockchain networks, the exact parameter values are not easy to obtain and may change over time, hindering the practical adoption of the solution. In this paper, we propose a model-free approach that solves the mining MDP using machine learning tools. In particular, we solve the mining MDP using reinforcement learning (RL) without the need to know the parameter values in the mining MDP model.

RL is a machine-learning paradigm, where agents learn successful strategies that yield the largest long-term reward from trial-and-error interactions with their environment [19], [20]. Q-learning is the most popular RL technique [21]. It can learn a good policy by updating a state-action value function without an operating model of the environment. RL has been successfully applied in many challenging tasks, e,g., playing video games [22] and Go [23], and controlling robotic movements [24].

The original RL algorithm cannot deal with the nonlinear objective function of our mining problem. In this paper, we put forth a new multi-dimensional RL algorithm to tackle the problem. Experimental results indicate that our multi-dimensional RL mining algorithm can successfully find the optimal strategy. Importantly, it demonstrates robustness and adaptability to a changing environment (i.e., parameter values changing dynamically over time).

II. Blockchain Preliminaries

Blockchain is a decentralized append-only ledger for digital assets. The data of blockchain is replicated and shared among all participants. Its past recorded data are tamper-resistant and participants can only append new data to the tail-end of the chain of blocks. The state of blockchain is changed according to transactions, and transactions are group into blocks that are appended to the blockchain. The header of the block encapsulates the hash of the preceding block, the hash of this block, the Merkle root[1] of all transactions contained in this block, and a number called nonce that is generated by PoW. Since each block must refer to its preceding block by placing the hash of its preceding block in its header, all the blocks form a chain of blocks arranged in chronological order. Fig. 1 illustrates the data structure of blockchain.

A. Proof of Work and Mining

In this paper, we focus on a Bitcoin-like blockchain that adopts the PoW consensus protocol to validate new blocks in a decentralized manner.[2] In each round, the PoW protocol selects a leader that is responsible for packing transactions into a block and appends this block to the blockchain. To prevent adversaries from monopolizing the blockchain, the leader selection must be approximately random. Since Bitcoin-like blockchain is permissionless and anonymity is inherently designed as the goal, it must consider the Sybil attack where an adversary simply creates many participants with different identities to increase its probability of being selected as the leader. To address the above issues, the key idea behind PoW is that a participant will be randomly selected as the leader of each round with a probability in proportion to its computing power.

In particular, blockchain implements PoW using computational hash puzzles. To create a new block, the nonce placed into the header of the block must be a solution to the hash puzzle expressed by the following inequality

$$\mathcal{H}(n, p, m) < D \tag{1}$$

where the nonce n, the hash of the previous block p, the Merkle root of all included transactions m are taken as the input of a cryptographic hash function $\mathcal{H}(\cdot)$ and the output of the hash function should fall below a target D that is small with respect to the whole range of the hash function outputs. The used hash function (e.g., SHA-256 hash is used for Bitcoin) satisfies the property of puzzle friendliness [26]: it is challenging to guess the nonce to fulfill (1) by a one-shot try. The only way to solve (1) is to try a large number of nonces one by one to check if (1) is fulfilled until one lucky nonce is found. Therefore, the probability of finding such a nonce is proportional to the computing power of the participant—the faster the hash function in (1) can be computed in each trial, the more nounces can be tried per unit time. Using the blockchain terminology, the process of computing hashes to find a nonce is called *mining*, and the participants involved are called *miners*.

B. Honest Mining Strategy

When a miner tries to append a new block to the latest legal block by placing the hash of the latest block in the header of the new block, we say that the miner mines on the latest block. The blockchain is maintained by miners in the following manner.

[1]The Merkle root of the transactions is the hash value of the Merkle tree whose leaves are the transactions [25].

[2]There are also blockchains adopting other several consensus algorithms, such as Proof of Stake (PoS), and Byzantine fault tolerance (BFT) [4].

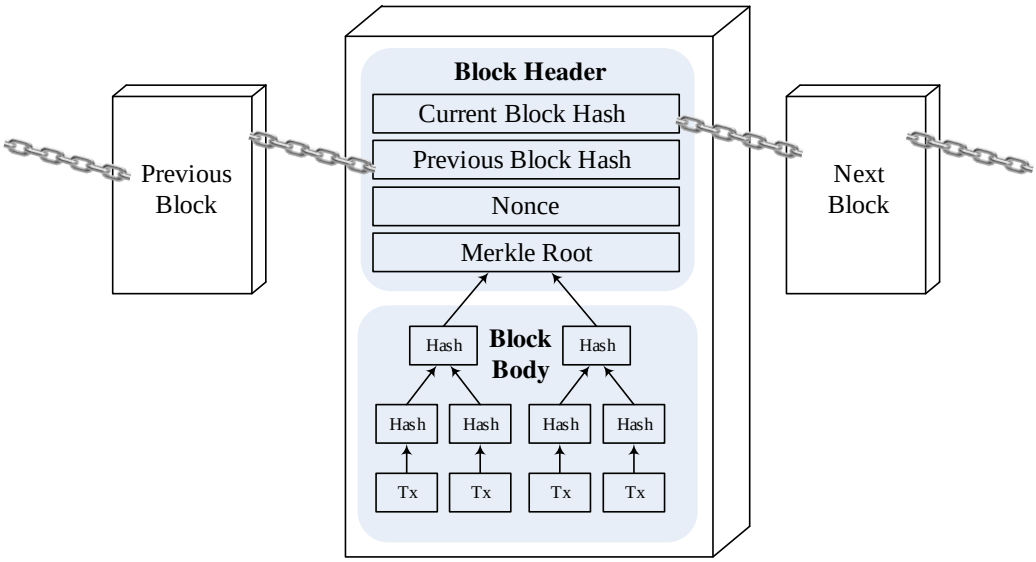

Fig. 1: Data structure of blockchain.

To encourage all miners to mine on, and maintain, the current blockchain, a *reward* is given as an incentive to the miner by placing a coin-mint transaction in its mined block that credits the miner with some new coins. If the block is verified and accepted by other peers in the blockchain network, the reward is effective and thus can be spent on the blockchain. When a miner has found an eligible nonce, it publishes his block to the whole blockchain network. Other miners will verify the nonce and transactions contained in that block. If the verification of the block is passed, other miners will mine on the block (implicitly accepting the block); otherwise, other miners discard the block and will continue to mine on the previous legal block.

If two miners publish two different legal blocks that refer to the same preceding block at the same time, the blockchain is then forked into two branches. This is called *forking* of blockchain. Forks in the blockchain because they are manifestations of disagreement among peers on the blockchain structure. It can also compromise the integrity and security of the blockchain [27]. To resolve a fork, PoW prescribes that only the rewards of the blocks on the longest branch (called the main chain) are effective. Then, miners are incentivized to mine on the longest branch, i.e., miners always add new blocks after the last block on the longest main chain that is observed from their local perspectives. If the forked branches are of equal length, miners may mine subsequent blocks on either branch randomly. This is referred to as the rule of the longest chain extension. Eventually, one branch will predominate and the other branches are discarded by peers in the blockchain network.

The mining strategy adhering to the rule of the longest chain extension and publishing a block immediately after the block is mined is referred to as *honest mining* [3]–[5]. The miners that comply with honest mining are called honest miners. It was widely believed that the most profitable mining strategy for miners is honest mining; and that when all miners adopt honest mining, each miner is rewarded in proportion to its computing power [3]–[5]. As a result, any rational miner will not deviate from honest mining. This belief was later shown to be ill-founded and that other mining strategies with higher profits are possible [15], [17], [18]. We will briefly discuss these mining strategies in the next section. For a more concrete exposition, we will first present the mining model.

III. Blockchain Mining Model

In this section, we present the Markov Decision Process (MDP) model for blockchain mining. Ref. [15] first developed an MDP mining model and used the model to construct a selfish mining strategy with higher rewards than honest mining. Then, [17] proposed even more profitable selfish mining strategies. Recently, [18] extended the MDP mining models of [15], [17] to a more general form. In this work, we adopt the mining model of [18].

Without loss of generality, we assume the network is split into two mining pools: one is an adversary that controls a fraction α of the whole network's computing power; the other is the network of honest miners that controls a fraction $1-\alpha$ of the computing power of the whole network.

Even if the adversary and an honest miner release their newly mined blocks to the network simultaneously, the blocks will not be received by all miners simultaneously due to propagation delays and network connectivity. We model the communication capability of the adversary using the parameter γ, defined as the fraction of the honest miners that will first receive the block from the adversary when the adversary and one honest miner release their blocks approximately at a same time—more specifically, $\gamma(1-\alpha)$ is the computing power of the honest network that will mine on the block of the adversary when the adversary and an honest miner release their blocks simultaneously.

As in [18], we model blockchain mining as a single-player MDP $M = \langle S, A, P, R\rangle$, where S is the state space, A is the action space, P is the transition probability matrix and R is the reward matrix. Each transition is triggered by the event of

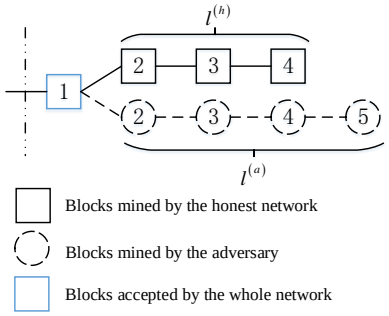

Fig. 2: An illustrating example of the state in the adopted MDP.

a miner mining a new block, whether the block is mined by the adversary or one of the honest miners. The action taken by the adversary based on the previous state, together with the event, determines the next state to which the system evolves.

The objective of the adversary is to earn rewards higher than its computational power. To achieve this, the adversary will generally deviate from honest mining by building a private chain of blocks without releasing them the moment the blocks are mined; the adversary will release several blocks from its private chain at a time to undo the honest chain opportunistically.

State: Each state in the state space is represented by a three-tuple form $\big(l^{(a)}, l^{(h)}, fork\big)$, where $l^{(a)}$ and $l^{(h)}$ are respectively the lengths of the adversary's chain and the honest network's chain after the latest fork (as illustrated in Fig. 2). In general, $fork$ can take three possible values $(irrelevant, relevant, active)$. Their meanings will be explained later.

Action: The action space A includes four actions that can be executed by the adversary.

- *Adopt*: The adversary accepts the honest chain and mines on the last block of the honest chain. This action discards the $l^{(a)}$ blocks in the chain of the adversary and it renews the attack from the new starting point without a fork. This action is allowed by the MDP model for all $l^{(a)}$ and $l^{(h)}$.
- *Override*: The adversary publishes one block more than the honest chain (i.e., $l^{(h)}+1$ blocks) to the whole network. This action overrides the conflicting blocks of the honest chain. This action is allowed when $l^{(a)} > l^{(h)}$.
- *Match*: The adversary publishes the same number of blocks as the honest chain (i.e., $l^{(h)}$ blocks) to the whole network. This action creates a fork deliberately and initiates an open mining competition between the two branches of the adversary and the honest network. This action is allowed when $l^{(a)} \geq l^{(h)}$ and $fork = relevant$.
- *Wait*: The adversary does not publish blocks and it just keeps mining on its own chain. This action is always feasible.

One remark about the actions of the MDP mining model is that some actions that can generally be performed are deliberately removed from the action-state space because these actions are not gainful for the adversary. For example, when $l^{(a)} < l^{(h)}$, the adversary can still release a certain number of its blocks. However, since releasing fewer blocks than the number of blocks on the honest chain will not increase its probability of mining the next block compared to mining it privately, these actions thus are excluded from the allowed actions.

We now explain the three values of the entry $fork$ in the three-tuple state.

- $Relevant$: The value of relevant means that the latest block is mined by the honest network. Now, if $fork = relevant$ and $l^{(a)} \geq l^{(h)}$, the action $match$ is allowed. For example, if the previous state is $\big(l^{(a)}, l^{(h)}-1, \bullet\big)$ and now the honest network successfully mines one block, the state then changes to $\big(l^{(a)}, l^{(h)}, relevant\big)$. If at this time, $l^{(a)} \geq l^{(h)}$, $match$ is allowed. We remark that $match$ here may be gainful for the adversary because $\gamma(1-\alpha)$ computing power of the honest network would be dedicated to mining on the adversary chain because of the near-simultaneous releases of the latest block of the adversary chain and the latest block of the honest chain. In this state, as far as the public is concerned, there no fork yet, since the $l^{(a)}$ mined blocks of the adversary are private and hidden from the public. However, if the adversary execute a match from this state, then a fork will be made known to the public and an active competition between the two branches will follow.
- $Irrelevant$: The value of $irrelevant$ means that the latest block is mined by the adversary and the blocks published by the honest network have been already received by (the majority of) the honest network. Now, even if $l^{(a)} \geq l^{(h)}$, the action $match$ is not allowed. For example, if the previous state is $\big(l^{(a)}-1, l^{(h)}, \bullet\big)$ and now the adversary successfully mines a new block, the state changes to $\big(l^{(a)}, l^{(h)}, irrelevant\big)$. We emphasize that $match$ is disallowed here even if $l^{(a)} \geq l^{(h)}$, not because it cannot be performed in the blockchain, but rather $match$ here is not gainful for the adversary. If $match$ were to be performed here, no computing power of the honest network would shift to mining on the adversary chain because the miners in the honest network would have received the latest block of the honest chain first (well before the current transition triggered by the adversary mining a new block) and would have dedicated to mining on the honest chain already. Again, in this state, there is no fork as far as the public blockchain is concerned.
- $Active$: The value of $active$ means that the adversary has executed the action $match$ from the previous state, and the blockchain is now split into two branches. For example, if the previous state is $\big(l^{(a)}, l^{(h)}, relevant\big)$ with $l^{(a)} \geq l^{(h)}$ and the adversary executed the action $match$. If the new transition is triggered by the honest network mining a new block, then the state transitions to $\big(l^{(a)}-l^{(h)}, 1, active\big)$. In short, $active$ means a fork is made known to the public and that an active competition between the two branches of the fork is ongoing.

Transition and Reward: After the execution of an action, the occurrence of each state transition is triggered by the

creation of a new block (either by the adversary or by the honest network) and the corresponding transition probability is the probability of the block created by the adversary (α) or by the honest network ($1-\alpha$). The initial state is $(1, 0, irrelevant)$ with probability α or $(0, 1, irrelevant)$ with probability $1-\alpha$. Different actions performed by the adversary will have different effects on the state transitions. The specific description is as follows:

- The state transitions after the execution of action $adopt$: By executing the $adopt$ action, the adversary accepts all the blocks on the branch mined by the honest network and mines on the latest block on the honest chain together with the honest network. An illustrating example of the state transitions after the execution of action $adopt$ is given in Fig. 3. As shown in Fig. 3, with the probability of α, the adversary can successfully mine the next block and then the state transits to $(1, 0, relevant)$; with the probability of $1-\alpha$, the honest network can successfully mine the next block and then the state transits to $(0, 1, relevant)$.

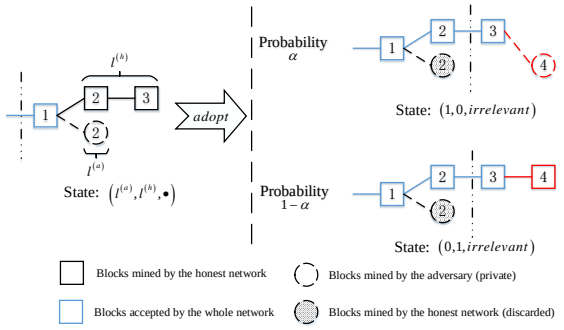

Fig. 3: An illustrating example of the state transitions after the execution of action $adopt$.

- The state transition after the execution of action $override$: The adversary can only perform action $override$ when the number of the blocks on its private branch is greater than the number of the blocks on the honest branch (i.e., when $l^{(a)} > l^{(h)}$). By performing $override$, the adversary publishes $l^{(h)}+1$ blocks from its private branch to overwrite the latest $l^{(h)}$ blocks on the honest branch. After that, the branch of the adversary becomes the main chain and the whole network mines on the latest block of the adversary's branch. An illustrating example of the state transitions after the execution of action $override$ is given in Fig. 4. As shown in Fig. 4, the adversary has the probability of α to successfully mine the next block and makes the state transit to $\left(l^{(a)}-l^{(h)}, 0, irrelevant\right)$; the honest network has the probability of $1-\alpha$ to successfully mine the next block and makes the state transit to $\left(l^{(a)}-l^{(h)}-1, 1, relevant\right)$.
- The state transition after the execution of action $match$: The $match$ action can only be executed when $fork = relevant$ and when the number of blocks on the private branch of the adversary is greater than or equal to the number of blocks on the public branch of the honest network (i.e., when $l^{(a)} \geq l^{(h)}$). After the adversary performs the $match$ action, a fork will be formed on the blockchain that is observed by all the miners. After that, the adversary is still mining on its own branch; however, due to the fork, a γ fraction of the honest network will mine on the branch published by the adversary, and the other $1-\gamma$ fraction of the honest network will mine on the branch published by the honest network. An illustrating example of the state transitions after the execution of action $match$ is given in Fig. 5. As shown in Fig. 5, the next block may be published by the adversary on its own branch such that the state transits to $\left(l^{(a)}+1, l^{(h)}, active\right)$ with the probability of α; the next block may be published by the honest network on the branch of the adversary such that the state transits to $\left(l^{(a)}-l^{(h)}, 1, relevant\right)$ with the probability of $\gamma(1-\alpha)$; the next block may be published by the honest network on the branch of the honest network such that the state transits to $\left(l^{(a)}, l^{(h)}+1, relevant\right)$ with the probability of $(1-\gamma)(1-\alpha)$. We must emphasize that after the execution of action $match$, among the $l^{(a)}$ blocks of the adversary, some of the blocks may be private while other blocks are public. Which parts of blocks are private/public are implied by the state implicitly. For example, suppose that the previous state is $\left(l^{(a)}, l^{(h)}, relevant\right)$ with $l^{(a)} > l^{(h)}$ (as illustrated in the left part of Fig. 5) and the action $match$ is performed. If the adversary subsequently mines a new block on its own branch, then the state changes to $\left(l^{(a)}+1, l^{(h)}, active\right)$, where there are $l^{(a)}+1-l^{(h)}$ private blocks and $l^{(h)}$ public blocks among the $l^{(a)}+1$ blocks owned by the adversary (as illustrated by the first case in the right part of Fig. 5). If the honest miners mine a new block on the adversary's branch, the state changes to $\left(l^{(a)}-l^{(h)}, 1, relevant\right)$, where there are $l^{(a)}-l^{(h)}$ private block left for the adversary (as illustrated by the second case in the right part of Fig. 5). If the hones miners mine a new block on the honest network's branch, the state changes to $l^{(a)}-l^{(h)}$, where there are $l^{(a)}-l^{(h)}$ private blocks and $l^{(h)}$ public blocks among the $l^{(a)}$ blocks owned by the adversary (as illustrated by the third case in the right part of Fig. 5).

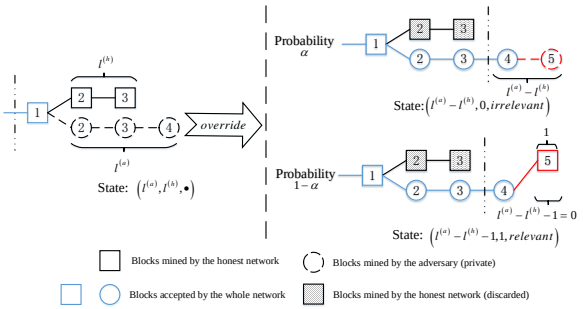

Fig. 4: An illustrating example of the state transitions after the execution of action $override$.

- The state transition triggered by action $wait$: The $wait$ action means that the adversary does not perform any

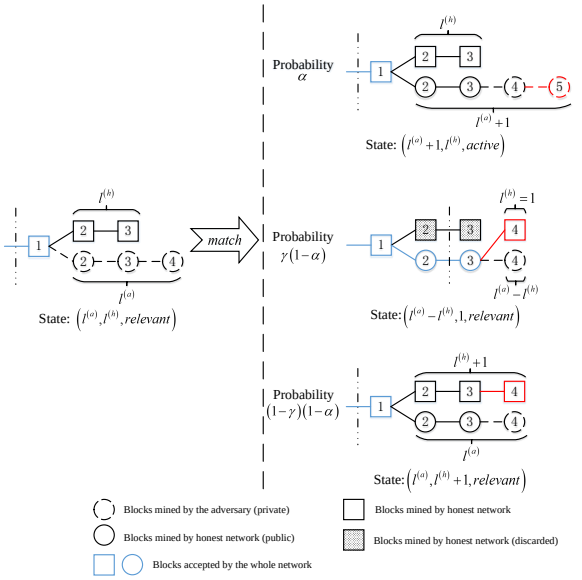

Fig. 5: An illustrating example of the state transitions after the execution of action $match$.

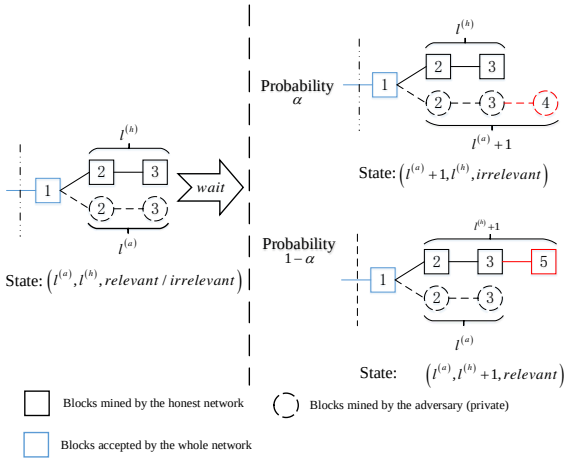

Fig. 6: An illustrating example of the state transitions after the execution of action $wait$ when $fork \neq active$.

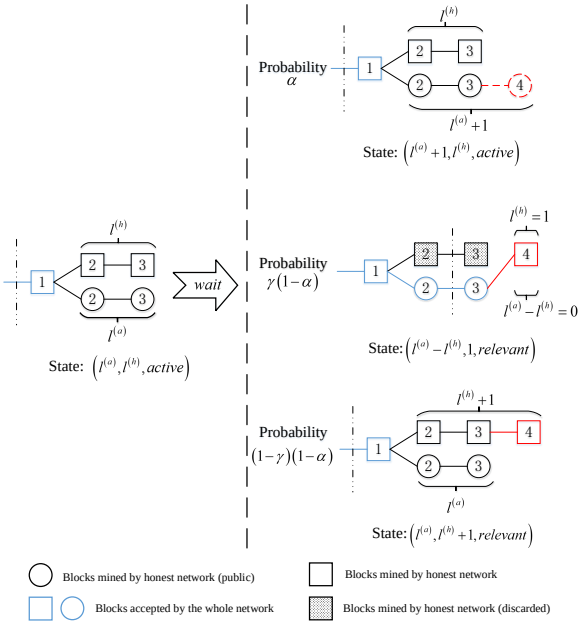

Fig. 7: An illustrating example of the state transitions after the execution of action $wait$ when $fork = active$.

actions and continues to mine on its private branch. After the action $wait$ is executed, if $fork \neq active$, the adversary and the honest network mine on their own branches respectively. An illustrating example of the state transitions after the execution of action $match$ when $fork \neq active$ is given in Fig. 6. As shown in Fig. 6, when $fork \neq active$, the next new block may be mined by the adversary on its own private branch such that the state changes to $\left(l^{(a)}+1, l^{(h)}, irrelevant\right)$ with the probability of α; or the next new block may be mined by the honest network on the public branch such that the state changes to $\left(l^{(a)}, l^{(h)}+1, relevant\right)$ with the probability of $1-\alpha$. After the action $wait$ is executed, if $fork = active$, due to the fork that can be observed by the whole network, the mining behaviors of all miners are the same as that after the execution of the $match$ action. An illustrating example of the state transitions after the execution of action $match$ when $fork = active$ is given in Fig. 7. As shown in Fig. 7, when $fork = active$, the next new block may be mined by the adversary on its own branch such that the state changes to $\left(l^{(a)}+1, l^{(h)}, active\right)$ with probability α; or the next new block may be mined by the honest network on the branch of the adversary such that the state changes to $\left(l^{(a)}-l^{(h)}, 1, relevant\right)$ with probability $\gamma(1-\alpha)$; or the next new block may be mined by the honest network on the branch of the adversary such that the state changes to $\left(l^{(a)}, l^{(h)}+1, relevant\right)$ with probability $(1-\gamma)(1-\alpha)$.

The reward is given as a tuple $\left(r^{(a)}, r^{(h)}\right)$, where $r^{(a)}$ denotes the number of blocks mined by the adversary and accepted by the whole network, and $r^{(h)}$ denotes the number of blocks mined by the honest network and accepted by the whole network. The state transitions and reward matrices are given in TABLE I.

Objective Function: The objective of the adversary is to find the optimal mining strategy that can earn as much reward as possible. Since blockchain keeps adjusting the mining difficulty (i.e., the mining target on the RHS of inequality (1)) to ensure that on average one valid block is introduced to the overall blockchain per *valid block interval* (e.g., one block per 10 minutes for Bitcoin, and per 10-20 seconds for Ethereum), the mining objective of the adversary is not to maximize its absolute cumulative reward, but to maximize the ratio of its cumulative rewards over the cumulative rewards of the whole network (i.e., the cumulative rewards of the whole network advance by one reward per block interval—rewards of all miners/Time is fixed to 1 per block interval; then maximizing adversary rewards/Time is equivalent to maximizing the ratio of adversary rewards/Time to rewards of all miners/Time = adversary rewards/rewards of all miners). We emphasize

TABLE I: The state transitions and reward matrices of the MDP mining model.

Current State, Action	Next State	Transition Probability	Reward
$\left(l^{(a)}, l^{(h)}, \bullet\right), adopt$	$(1, 0, irrelevant)$	α	$\left(0, l^{(h)}\right)$
	$(0, 1, irrelevant)$	$1-\alpha$	
$\left(l^{(a)}, l^{(h)}, \bullet\right), override$	$\left(l^{(a)}-l^{(h)}, 0, irrelevant\right)$	α	$\left(l^{(h)}+1, 0\right)$
	$\left(l^{(a)}-l^{(h)}-1, 1, relevant\right)$	$1-\alpha$	
$\left(l^{(a)}, l^{(h)}, irrelevant\right), wait$	$\left(l^{(a)}+1, l^{(h)}, irrelevant\right)$	α	$(0, 0)$
$\left(l^{(a)}, l^{(h)}, relevant\right), wait$	$\left(l^{(a)}, l^{(h)}+1, relevant\right)$	$1-\alpha$	$(0, 0)$
$\left(l^{(a)}, l^{(h)}, active\right), wait$	$\left(l^{(a)}+1, l^{(h)}, active\right)$	α	$(0, 0)$
$\left(l^{(a)}, l^{(h)}, relevant\right), match$	$\left(l^{(a)}-l^{(h)}, 1, relevant\right)$	$\gamma(1-\alpha)$	$\left(l^{(h)}, 0\right)$
	$\left(l^{(a)}, l^{(h)}+1, relevant\right)$	$(1-\gamma)(1-\alpha)$	$(0, 0)$

The action $override$ is allowed when $l^{(a)} > l^{(h)}$; the action $match$ is allowed when $l^{(a)} \geq l^{(h)}$.

that blocks mined by the adversary and the honest network that are discarded due to losing out in the competition are not considered as having been successfully introduced to the blockchain. Thus, the principle behind the strategy of the adversary is to maximize the number of blocks mined by the honest network that are later discarded while reducing its own discarded blocks.

As in [18], we define the following $relative\ mining\ gain$ (RMG) as the objective function for blockchain mining:

$$RMG = E\left[\lim_{T\to\infty} \frac{\sum_{\tau=t}^{t+T-1} r_{\tau+1}^{(a)}}{\sum_{\tau=t}^{t+T-1} r_{\tau+1}^{(a)} + \sum_{\tau=t}^{t+T-1} r_{\tau+1}^{(h)}}\right] \quad (2)$$

where $\left(r_t^{(a)}, r_t^{(h)}\right)$ is the tuple of rewards issued in the block interval t, T is the size of the observing window. The objective of the adversary is to maximize this relative mining gain.

Under the above MDP mining model, we can now interpret honest mining, selfish mining [15], lead stubborn mining [17] as examples of different mining strategies.

Honest Mining: For honest mining, miners will follow the rule of the longest chain extension. Thus, they will not maintain a private chain: when they have a new block, they will immediately publish it. The honest mining strategy can be written as

$$HM\left(l^{(a)}, l^{(h)}, \bullet\right) = \begin{cases} adopt & l^{(a)} < l^{(h)} \\ wait & l^{(a)} = l^{(h)} \\ override & l^{(a)} > l^{(h)} \end{cases} \quad (3)$$

where we note that $l^{(a)}$, $l^{(h)}$ can only take a value of 0 or 1.

Selfish Mining: The main idea of selfish mining [15] is described as follows. If one block is found by the adversary, it does not publish it immediately and it keeps mining on its private chain. When the adversary already has one private block and then honest network publishes one block (immediately after an honest miner mines a new block), the adversary chooses to publish its block to match the honest network. This causes $\gamma(1-\alpha)$ computing power of the honest network to mine on the adversary's chain. When the adversary already has some private blocks and then honest network catches up with only one block less than the adversary ($l^{(h)} = l^{(a)} - 1 \geq 1$), the adversary overrides the honest network's block by publishing all its blocks. The selfish mining strategy can be written as

$$SM\left(l^{(a)}, l^{(h)}, \bullet\right) = \begin{cases} adopt & l^{(a)} < l^{(h)} \\ match & l^{(a)} = l^{(h)} = 1 \\ override & l^{(h)} = l^{(a)} - 1 \geq 1 \\ wait & otherwise \end{cases} \quad (4)$$

Lead Stubborn Mining: Lead stubborn mining [17] is different from selfish mining in the following way. A lead stubborn miner always publishes one block from its private chain to match with the honest network when the honest network mines a new block if $l^{(a)} \geq l^{(h)}$. The adversary never executes the action override. The lead stubborn mining can be written as

$$LSM\left(l^{(a)}, l^{(h)}, fork\right) = \begin{cases} adopt & l^{(a)} < l^{(h)}, \forall fork \\ match & otherwise \\ wait & l^{(a)} > l^{(h)}, fork = irrelevant \end{cases} \quad (5)$$

It is shown that this lead stubborn mining can achieve higher profits than selfish mining [17].

Optimal Mining: Although there are many possible mining strategies that can obtain profits higher than honest mining, the optimal mining strategy is not obvious. Since the state-action space of the MDP is huge, it is not straightforward to derive the optimal mining strategy. The relative mining gain objective (2) is a nonlinear function of the rewards, and thus the corresponding MDP cannot be solved using standard MDP solvers to give the optimal mining strategy. To solve this problem, [18] first transformed the MDP with the nonlinear objective to a family of MDPs with linear objectives, and then employed a standard MDP solver combined with a numerical search over the family of MDPs to find the optimal mining strategy. As shown in [18], its solution indeed can find the optimal mining strategy. However, the solution of [18] is model-based approach: it must know the parameters that characterize the MDP model exactly (i.e., the computing power distribution α, the communication capability γ). In real blockchain networks, these parameters are not easy to obtain and may change over time, hindering the use of the solution proposed in [18]. We propose a model-free approach that solves the MDP with the nonlinear objective using RL.

IV. Mining Through RL

This section first provides preliminaries for RL and then presents a new RL algorithm that can derive the optimal

mining strategy without knowing the parameters of the environment. We propose the new RL mining algorithm based on Q-learning, one popular algorithm from the RL family.

A. Preliminaries for Original Reinforcement Learning Algorithm

In RL, an agent interacts with an environment in a sequence of discrete time steps, $t = 0, 1, 2, ...$, as shown in Fig. 8. At time t, the agent observes the state of the environment, s_t; it then takes an action, a_t. As a result of the state-action pair, (s_t, a_t), the agent receives a scalar reward r_{t+1}, and the environment moves to a new state s_{t+1} at time $t+1$. Based on s_{t+1}, the agent then decides the next action a_{t+1}. The goal of the agent is to effect a series of rewards $\{r_t\}_{t=1,2,...}$ through its actions to maximize some performance criterion. For example, for Q-learning [21], the performance criterion to be maximized at time t is the discounted accumulated rewards going forward $R_t = \sum_{\tau=t}^{\infty} \lambda^{\tau-t} r_{\tau+1}$, where $\lambda \in (0, 1)$ is a discount factor for weighting future rewards [19]. In general, the agent takes actions according to some decision policy π. RL methods specify how the agent changes its policy as a result of its experiences. With sufficient experiences, the agent can learn an optimal decision policy π^* to maximize the long-term accumulated reward [19].

The desirability of state-action pair (s_t, a_t) under a decision policy decision π is captured by a Q function, defined as $Q(s, a) = [R_t | s_t = s, a_t = a, \pi]$, i.e., the expected discounted accumulated reward going forward given the current state-action pair (s_t, a_t). The optimal decision policy π^* is one that maximizes Q function. In Q-learning, the goal of the agent is to learn the optimal policy π^* through an online-iterative process by observing the rewards while it takes action in successive time steps. In particular, the agent maintains the Q function, $Q(s, a)$, for all state-action pairs (s, a), in a tabular form.

Let $q(s, a)$ be the estimated action-value function during the iterative process. At time step t, given state s_t, the agent selects a greedy action $a_t = \arg\max_a q(s_t, a)$ based on its current Q function. This will cause the system to return a reward r_{t+1} and move to state s_{t+1}. The experience at time step t is captured by the quadruplet $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$. At the end of time step t, experience e_t is used to update $q(s_t, a_t)$ for entry (s_t, a_t) as follows:

$$q(s_t, a_t) \leftarrow (1-\beta)\, q(s_t, a_t) + \beta \left[r_{t+1} + \lambda \max_{a'} q(s_{t+1}, a') \right] \tag{6}$$

where $\beta \in (0, 1]$ is a parameter that governs the learning rate. Q-learning learns from experiences gathered over time, $\{e_t\}_{t=0,1,...}$, through the iterative process in (6). Note that Q-learning is a model-free learning framework in that it tries to learn the optimal policy without having a model that describes the operating behavior of the environment beyond what can be observed through the experiences.

As a deviation from the above description, a caveat in Q-learning is that the so-called ε-greedy algorithm is often adopted in action selection. For the ε-greedy algorithm, the action $a_t = \arg\max_a q(s_t, a)$ is only chosen with probability $1-\varepsilon$. With probability ε, a random action is chosen uniformly from the set of possible actions. This is to avoid the algorithm from zooming in to a local optimal policy and to allow the agent to explore a wider spectrum of different actions in search of the optimal policy [19].

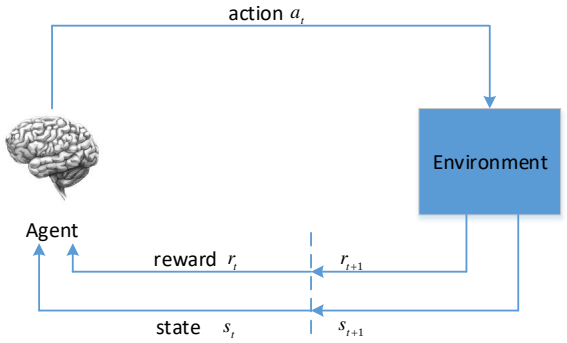

Fig. 8: The agent-environment interaction process of RL algorithm.

It has been shown that in a stationary environment that can be fully captured by an MDP, the Q-values will converge to optimality if the learning rate decays appropriately and each action in the state-action pair (s, a) is executed an infinite number of times in the process [19].

B. New Reinforcement Learning Algorithm for Mining

The original RL algorithm as presented in Section IV.A cannot be directly applied to maximize the mining objective function expressed in (2); there is one fundamental obstacle that must be overcome. The obstacle is the nonlinear combination of the rewards in the objective function. The original RL algorithm can only maximize an objective that is a linear function of the scalar rewards, e.g., the weighted sum of scalar rewards. To address this issue, we put forth a new algorithm that aims to optimize the original mining objective: the multi-dimensional RL algorithm.

We formulate the multi-dimensional RL algorithm as follows. At *mined block interval* [3] t ($t = 0, 1, 2, \cdots$), the state $s_t \in S$ takes a value from the state space S as defined in the MDP model of blockchain mining, and the action $a_t \in A$ is chosen from the action space A. The state transition occurs according to TABLE I. The reward is the pair $\left(r_{t+1}^{(a)}, r_{t+1}^{(h)}\right)$ whose value is assigned according to TABLE I. The experience at the end of mined block interval t is given by $e_t = \left(s_t, a_t, s_{t+1}, r_{t+1}^{(a)}, r_{t+1}^{(h)}\right)$. The objective of the multi-

[3] A mined block interval is different from a valid block interval. A valid block interval separates two valid blocks that are ultimately adopted by the blockchain. The average duration of a valid block interval is a constant in many blockchain systems (e.g., 10 min in bitcoin). The average duration of the valid block interval is defined by the system designer and its constancy is maintained by adjusting the mining target. A mined block interval separated two mined (by either the adversary of the honest network), regardless of whether the blocks becomes valid later. In the MDP model, each transition is triggered by the mining of a new block. Thus the average duration of a mined block interval is the average time separates two adjacent transitions. Due to the actions of the adversary, some of the mined blocks (by the adversary of the honest network) may be discarded later.

dimensional RL algorithm is to maximize the relative mining gain as expressed in (2).

For a state-action pair (s_t, a_t), instead of maintaining an action-value scalar $Q(s,a)$, the multi-dimensional RL algorithm maintains an action-value pair $\left(Q^{(a)}(s,a), Q^{(h)}(s,a)\right)$ corresponding to the Q function values of the adversary and the honest network, respectively. The Q functions defined by Q learning are the expected cumulative discounted rewards. Specifically, $Q^{(a)}(s,a)$ and $Q^{(h)}(s,a)$ are defined as

$$\begin{aligned} Q^{(a)}(s,a) &= E\left[\lim_{T\to\infty}\sum\nolimits_{\tau=t}^{T}\lambda^{\tau-t}r_{\tau+1}^{(a)}\,|s_t=s,a_t=a,\pi\right] \\ Q^{(h)}(s,a) &= E\left[\lim_{T\to\infty}\sum\nolimits_{\tau=t}^{T}\lambda^{\tau-t}r_{\tau+1}^{(h)}\;|\,s_t=s,a_t=a,\pi\right] \end{aligned} \tag{7}$$

Suppose that at mined block interval t, the Q functions in (7) are estimated to be $q^{(a)}(s,a)$, $q^{(h)}(s,a)$. For action selection, we still adopt the ε-greedy approach. To select the greedy action, we construct the following objective function:

$$f(s,a) = \frac{q^{(a)}(s,a)}{q^{(a)}(s,a)+q^{(h)}(s,a)} \tag{8}$$

After taking action a_t, the state transitions to s_{t+1} and the reward pair $\left(r_{t+1}^{(a)}, r_{t+1}^{(h)}\right)$ is issued. With the experience $e_t = \left(s_t, a_t, s_{t+1}, r_{t+1}^{(a)}, r_{t+1}^{(h)}\right)$, the multi-dimensional RL algorithm updates the two Q functions as follows:

$$\begin{aligned} &q^{(a)}(s_t,a_t) \\ &\leftarrow (1-\beta)\,q^{(a)}(s_t,a_t)+\beta\left[r_{t+1}^{(a)}+\lambda q^{(a)}\left(s_{t+1},a'\right)\right] \\ &q^{(h)}(s_t,a_t) \\ &\leftarrow (1-\beta)\,q^{(h)}(s_t,a_t)+\beta\left[r_{t+1}^{(h)}+\lambda q^{(h)}\left(s_{t+1},a'\right)\right] \end{aligned} \tag{9}$$

where $a' = \arg\max_a f(s_{t+1}, a)$. Note that the update rule of (9) is very similar to the update rule of Q learning, except that the greedy action a' is chosen by maximizing the constructed objective function in (8) rather than maximizing the Q function itself as in Q learning. From the expressions in (7) and (8), we can verify that the adopted objective function in (8) is consistent with the relative mining gain objective function defined in (2), except the discount terms $\lambda^{\tau-t}$ used in the computation of the Q functions. The use of discount terms can ensure that the Q functions can converge to some bounded values; however, adding discount terms to the rewards will change the original mining objective. One simple way to ensure strict objective consistency is to set $\lambda = 1$. Although the setting of $\lambda = 1$ will result in unbound values for the Q functions as the RL iteration gradually progresses to infinite time steps, this is not a big problem as long as the Q function values do not overflow during the execution of the algorithm. In practice, we can also set λ to be very close to one.

The RL algorithm expressed by the Q function updates in (9) is our multi-dimensional RL algorithm. We introduce one additional technical element to the ε-greedy action selection, as explained in the next paragraph.

As described above, when we select the action, we adopt the ε-greedy strategy that allows us to select the current best action ($a_t = \arg\max_a f(s_t,a)$) with probability $1-\varepsilon$ and to randomly select an action with probability ε. This random action selection is used to explore some unseen states and can avoid trapping at local optimal maximums. However, the tuning of parameter ε is not straightforward. A large ε reduces the possibility of trapping at local optimal maximums but it also decrease the average reward, since it wastes a fraction of the time to explore non-optimal states. In our algorithm, we adopt the following strategy for dynamically tuning the parameter ε. Denote the number of times state was visited by $V(s_t)$. Then, the ε parameter used at state s_t for performing ε-greedy action selection is given by

$$\varepsilon(s_t) = \exp\left(-\frac{V(s_t)}{T_\varepsilon}\right) \tag{10}$$

where T_ε is a temperature parameter that governs how fast we gradually reduce the ε parameter. The pseudo-code of our multi-dimensional RL algorithm for blockchain mining is given in Algorithm 1.

Algorithm 1 Multi-dimensional RL Algorithm for Blockchain Mining

Initialize $q^{(a)}(s,a) = 0, \forall s, \forall a$;
Initialize $q^{(h)}(s,a) = 0, \forall s, \forall a$;
Initialize $V(s) = 0, \forall s$;
Initialize T_ε, λ, β;
for $t = 0, 1, 2, \cdots$ **do**
  Receive s_t, r_t from the blockchain environment;
  Generate action $a_t = \text{SELECTACTION}(s_t)$;
  Input a_t to the blockchain environment;
  Observe s_{t+1}, $r_{t+1}^{(a)}$, $r_{t+1}^{(h)}$ from the blockchain environment;
  Compute $a' = \arg\max_a \frac{q^{(a)}(s_{t+1},a)}{q^{(a)}(s_{t+1},a)+q^{(h)}(s_{t+1},a)}$
  Update
  $q^{(a)}(s_t,a_t)$
  $\leftarrow (1-\beta)\,q^{(a)}(s_t,a_t)+\beta\left[r_{t+1}^{(a)}+\lambda q^{(a)}\left(s_{t+1},a'\right)\right]$;
  $q^{(h)}(s_t,a_t)$
  $\leftarrow (1-\beta)\,q^{(h)}(s_t,a_t)+\beta\left[r_{t+1}^{(h)}+\lambda q^{(h)}\left(s_{t+1},a'\right)\right]$;
end for
procedure SELECTACTION(s_t)
  Compute $\varepsilon(s_t) = \exp\left(-\frac{V(s_t)}{T_\varepsilon}\right)$;
  if random $< \varepsilon(s_t)$ **then**
    randomly select an action a_t from A;
  else
    $a_t = \arg\max_{a\in A} \frac{q^{(a)}(s_t,a)}{q^{(a)}(s_t,a)+q^{(h)}(s_t,a)}$;
  end ifreturn a_t
end procedure

V. PERFORMANCE EVALUATIONS

We have conducted simulations to investigate our proposed RL mining strategy. Following the simulation approach used in [15], we constructed a Bitcoin-like simulator that captures all the relevant PoW network details described in previous

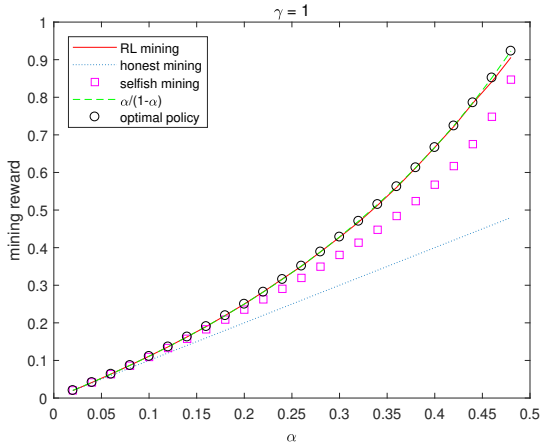

Fig. 9: Achieved mining gain versus α for $\gamma = 1$.

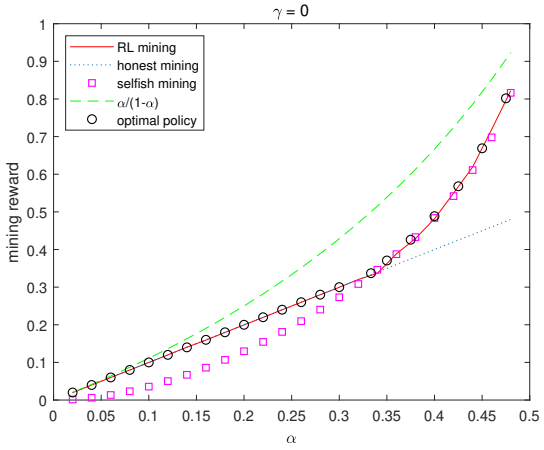

Fig. 11: Achieved mining gain versus α for $\gamma = 0$.

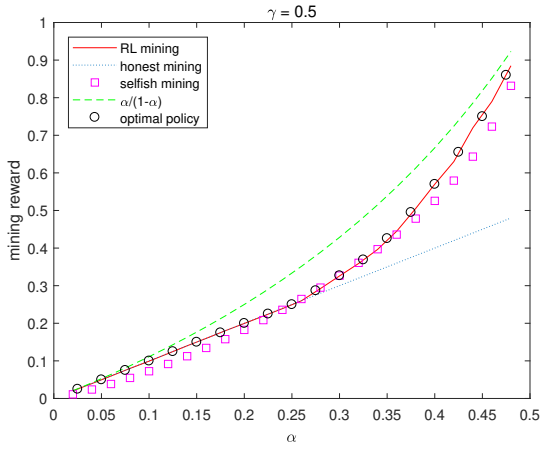

Fig. 10: Achieved mining gain versus α for $\gamma = 0.5$.

sections, except that the crypto puzzle solving processing was replaced by a Monte Carlo simulator that simulates the time required for block discovery without actually attempting to compute a hash function. We simulated 1000 miners mining at identical rates (i.e., they each can have one simulated hash test at each time step of the Monte Carlo simulation). A subset of the 1000 miners (1000α miners) forms an adversary pool running a malicious mining strategy that co-exists with honest mining adopted by the other $1000(1-\alpha)$ miners. When co-existing with honest mining, the malicious mining strategy is one of the following mining strategies: i) our RL mining strategy, ii) the optimal mining strategy derived in [18] or iii) the selfish mining strategy derived in [15]. Upon encountering two subchains of the same length, we divide the honest miners such that a fraction γ of them mine on the attacking pool's branch while the rest mine on the other branch. The performance metric used is the relative mining gain (RMG) computed over a window consisting of $T_w = 10^5$ time steps: $\sum_{\tau=t}^{t+T_w-1} r_{\tau+1}^{(a)} \Big/ \left(\sum_{\tau=t}^{t+T_w-1} r_{\tau+1}^{(a)} + \sum_{\tau=t}^{t+T_w-1} r_{\tau+1}^{(h)}\right)$. The hyper-parameters used in the RL algorithm are set to as $\lambda = 0.999$, $\beta = 0.05$.

We first compare the performances of our RL mining, the optimal-policy mining, and the selfish mining. Fig. 9-11 plots the mining reward of the adversary versus α for different values of γ and $\gamma \in \{0, 0.5, 1\}$. Note that the value of γ ranges in the interval [0, 1]. Therefore, $\gamma = 0$, $\gamma = 0.5$, $\gamma = 1$ respectively means a low, a median, and a high communication capability for the adversary. We just take these three values for γ to demonstrate that the adversary with our RL mining can dynamically adapt to the optimal mining when it has different communication capability. The relative mining gain of $\alpha/(1-\alpha)$ is treated as a bound for the mining problem and it can only be achieved by optimal-policy mining for $\gamma = 1$. To derive the optimal policy, we adopt the search algorithm proposed in [18] and set the search error to a very tiny number of 10^{-5}. As in [18], we truncate the MDP at $l^{(a)} = 100$ or $l^{(a)} = 100$ for both of optimal-policy mining and RL mining. The temperature parameter T_ε is set to as $T_\varepsilon = 10^4$ and it is reset to $T_\varepsilon = 0$ after $t = 10^8$ time steps when convergence is attained. All the results of RL mining are given after the algorithm has converged. From the results, we can see that the performance of our RL mining can converge to the performance of optimal-policy mining without knowing the details about the environment model.

We next consider the impact of the temperature parameter T_ε on the convergence of RL mining. Fig. 12-14 present the mining rewards obtained by RL mining with different T_ε over time for $\gamma \in \{0, 0.5, 1\}$, respectively (α is fixed to 0.45). In fact, the parameter of T_ϵ determines the extent of the exploration performed by the RL algorithm in its learning process. A larger T_ϵ encourages more explorations, and eventually, the learning process can converge, although a larger T_ϵ needs more time to converge. The optimal value of T_ϵ can lead to the convergence of RL by enough explorations and does not waste learning time by having unnecessary explorations. How to tune to the optimal T_ϵ is an interesting research direction. In this work, we just investigate the impact of T_ϵ on our RL mining by simulations. From the simulation results in Fig. 12-14, we can see that generally, RL mining with larger T_ε can have more explorations and can converge more closely to the optimal performance; however, RL mining with larger T_ε also have longer exploration phases that slow down the convergence process. Fig. 15 presents the mining

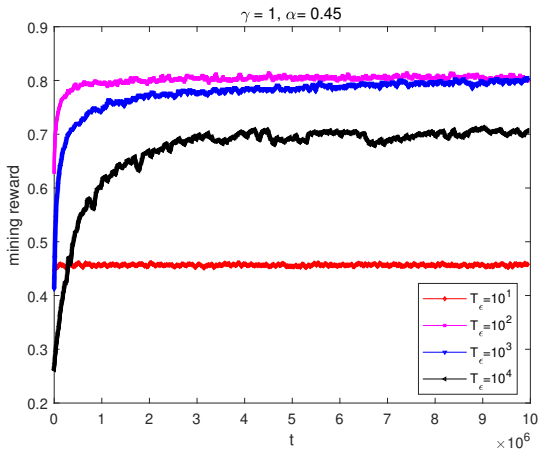

Fig. 12: Achieved mining gain versus time step for different T_ε and $\gamma = 1$, $\alpha = 0.45$.

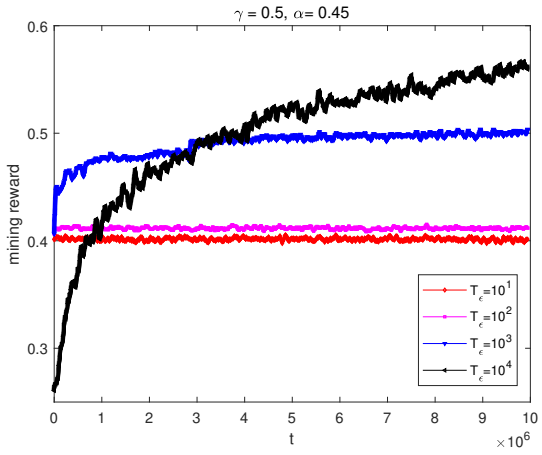

Fig. 13: Achieved mining gain versus time step for different T_ε and $\gamma = 0.5$, $\alpha = 0.45$.

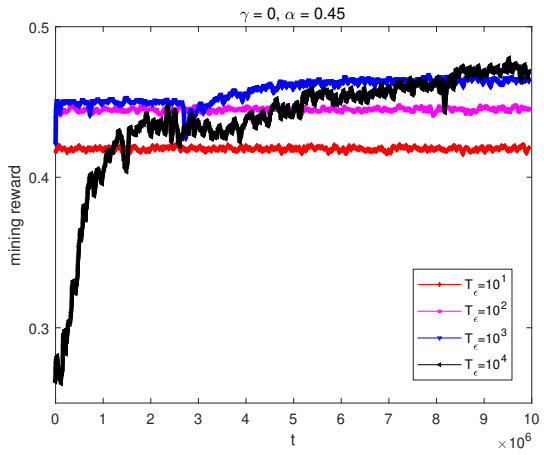

Fig. 14: Achieved mining gain versus time step for different T_ε and $\gamma = 0$, $\alpha = 0.45$.

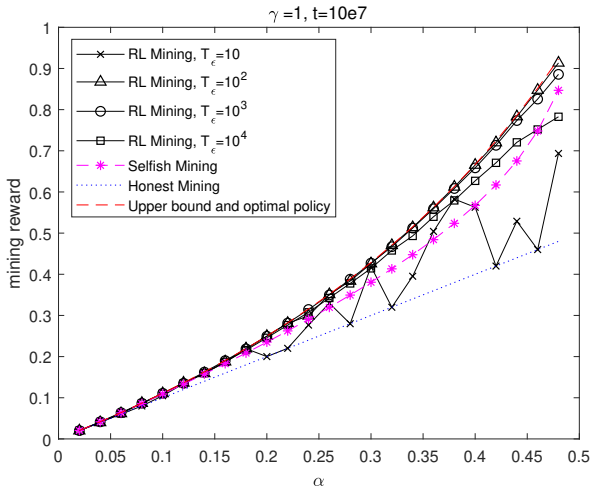

Fig. 15: Achieved mining gain versus the α for $\gamma = 1$ and different T_ε.

rewards of RL mining with different T_ε for different α (γ is fixed to 1). The mining reward results are given after $t = 10^7$ time steps and without resetting $T_\varepsilon = 0$. We see that for larger α, we need larger T_ε to ensure the convergence of RL mining, although it will slow down the convergence process. In practice, we can dynamically reduce the value of T_ε when we find that the mining gain has already converged.

Last, we investigate the mining performances of different mining strategies when the blockchain environment changes. The experimental results are given in Fig. 16-18. The blockchain environment starts with parameter values of $(\alpha = 0.35, \gamma = 1)$ and the values of (α, γ) change sequentially in the experiment. The temperature parameter T_ε of RL mining is fixed to $T_\varepsilon = 10^3$. The optimal-policy mining strategy adopts the optimal policy for the blockchain environment with $(\alpha = 0.35, \gamma = 1)$. We derived the optimal policy for $(\alpha = 0.35, \gamma = 1)$ by iteratively exploit the MDP solver [28] to search over the policy space, as proposed in [18]. TABLE II describes the found optimal policy for $(\alpha = 0.35, \gamma = 1)$ when $l^{(a)} \leq 8$ and $l^{(h)} \leq 8$. The performances of the optimal policy for $(\alpha = 0.35, \gamma = 1)$, and the selfish mining are treated as benchmarks for our RL mining in the changing blockchain environment. In Fig. 16-18, for different values of the parameters (α, γ), the performances of optimal selfish mining are still obtained using the policy of the optimal selfish mining under model parameters $(\alpha, \gamma) = (0.35, 1)$. From the simulation results, we can see that when the environment has changed, the optimal-policy mining strategy derived from the MDP model is not optimal anymore; our RL mining can adaptively learn the optimal policy for different environments. This demonstrates the advantage of RL mining over these model-based mining strategies.

VI. Conclusion

We employed RL algorithms to solve the mining MDP problem of Bitcoin-like blockchains. We showed that, without knowing parameters about the blockchain network model, our RL mining can achieve the mining reward of the optimal policy that requires knowledge of the parameters. Therefore, in a dynamic environment in which the parameter values can change over time, RL mining can be more robust.

TABLE II: The optimal policy for the blockchain environment with $(\alpha = 0.35, \gamma = 1)$ when $l^{(a)} \leq 8$ and $l^{(h)} \leq 8$.

$l^{(a)} l^{(h)}$	1	2	3	4	5	6	7	8
1	***	*a*	***	***	***	***	***	***
2	w**	*m*	*w*	*a*	***	***	***	***
3	w**	*oo	w**	*w*	*a*	***	***	***
4	w**	*m*	oo*	w**	*w*	*a*	***	***
5	w**	*mw	*m*	oo*	w**	*w*	*w*	*a*
6	w**	*mw	*mw	*m*	oo*	w**	ww*	*w*
7	w**	*mw	*mw	*mw	*m*	oo*	w**	ww*
8	w**	*mw	*mw	*mw	*mw	*m*	oo*	w**

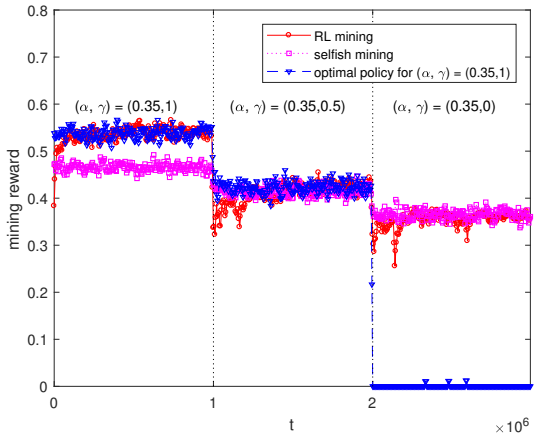

Fig. 16: Achieved mining gain when the environment is changing and the values of (α, γ) change in the following order: (0.35,1), (0.35, 0.5), and (0.35,0).

Going forward, we will investigate two issues that need to be addressed before RL mining can be practical:

1. More complete MDP model as proposed in [29] for blockchain networks—This model incorporates detailed blockchain features, such as stale block rate, double spending attack, and eclipsed attack, that have been precluded by the model in the current paper. The large action-space of the complete model will make it more challenging for RL mining to learn an optimal strategy.

2. Cost of lagged time in convergence—Since miners need to pay for their hardware and consume electricity to mine blocks, fast convergence of the mining algorithm is important from the economical standpoint. Deep RL [22] that incorporates deep neural networks into RL can potentially speed up the convergence rate. We will consider the exploit of deep RL in our future work.

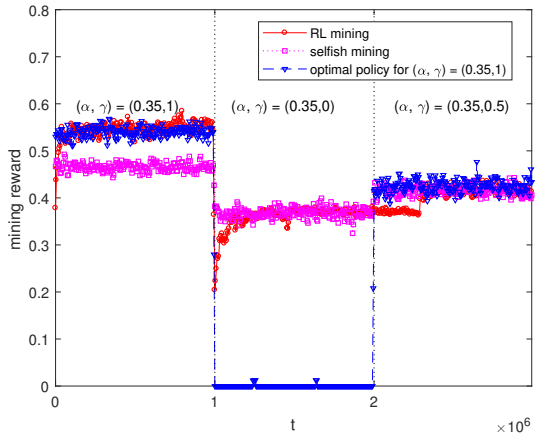

Fig. 17: Achieved mining gain when the environment is changing and the values of (α, γ) change in the following order: (0.35,1), (0.35, 0), and (0.35,0.5).

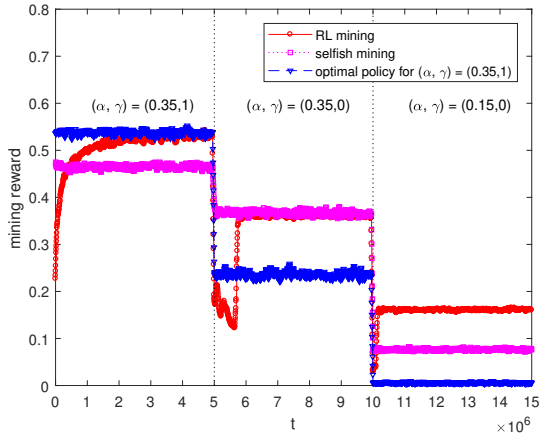

Fig. 18: Achieved mining gain when the environment is changing and the values of (α, γ) change in the following order: (0.35,1), (0.35, 0), and (0.15,0).

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *[Online]. Available: http://bitcoin.org*, 2008.

[2] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. O'Reilly Media, Inc., 2014.

[3] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[4] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22 328 – 22 370, 2019.

[5] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 6–14, 2018.

[6] K. Fanning and D. P. Centers, "Blockchain and its coming impact on financial services," *Journal of Corporate Accounting & Finance*, vol. 27, no. 5, pp. 53–57, 2016.

[7] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the internet of things: Research issues and challenges," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2188–2204, 2018.

[8] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.

[9] S. A. Abeyratne and R. P. Monfared, "Blockchain ready manufacturing supply chain using distributed ledger," 2016.

[10] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual International Cryptology Conference*. Springer, 1992, pp. 139–147.

[11] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.

[12] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the*

Theory and Applications of Cryptographic Techniques. Springer, 2017, pp. 643–673.

[13] C. Lee, "Litecoin-open source p2p digital currency," *[Online]. Available: https://litecoin.com/en/*, 2011.

[14] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," *[Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper*, 2014.

[15] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[16] D. Kraft, "Difficulty control for blockchain-based consensus systems," *Peer-to-Peer Networking and Applications*, vol. 9, no. 2, pp. 397–413, 2016.

[17] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 305–320.

[18] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 515–532.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[21] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[23] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[24] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[25] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*. Springer, 1987, pp. 369–378.

[26] M. Wang, M. Duan, and J. Zhu, "Research on the security criteria of hash functions in the blockchain," in *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*. ACM, 2018, pp. 47–55.

[27] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Deconstructing the blockchain to approach physical limits," *arXiv preprint arXiv:1810.08092*, 2018.

[28] I. Chadès, G. Chapron, M.-J. Cros, F. Garcia, and R. Sabbadin, "Mdptoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems," *Ecography*, vol. 37, no. 9, pp. 916–920, 2014.

[29] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM, 2016, pp. 3–16.