Cobalt: Optimizing Mining Rewards in Proof-of-Work Network Games

Arti Vedula, Abhishek Gupta and Shaileshh Bojja Venkatakrishnan

The Ohio State University

{vedula.9, gupta.706, bojjavenkatakrishnan.2}@osu.edu

Abstract—Mining in proof-of-work blockchains has become an expensive affair requiring specialized hardware capable of executing several megahashes per second at huge electricity costs. Miners earn a reward each time they mine a block within the longest chain, which helps offset their mining costs. It is therefore of interest to miners to maximize the number of mined blocks in the blockchain and increase revenue. A key factor affecting mining rewards earned is the connectivity between miners in the peer-to-peer network. To maximize rewards a miner must choose its network connections carefully, ensuring existence of paths to other miners that are on average of a lower latency compared to paths between other miners. We formulate the problem of deciding whom to connect to for miners as a combinatorial bandit problem. Each node picks its neighbors strategically to minimize the latency to reach 90% of the hash power of the network relative to the 90-th percentile latency from other nodes. A key contribution of our work is the use of a network coordinates based model for learning the network structure within the bandit algorithm. Experimentally we show our proposed algorithm outperforming or matching baselines on diverse network settings.

Index Terms—Mining Rewards, Proof-of-work, Combinatorial Bandit, Network Games

I. INTRODUCTION

Blockchain is rapidly emerging as a transformational technology for realizing trustless, decentralized and secure peer-to-peer (p2p) applications at large scales over the internet. Cryptocurrency, the earliest proposed blockchain application, has grown to be a trillion dollar market today, while thousands of decentralized applications (dapps) are enjoying massive popularity in myriad domains including healthcare, social networks, decentralized web and beyond. The continued demand for cryptocurrencies and dapps, despite recent turbulence in market sentiments, also underscores the value of blockchains as protocols for realizing truly democratized applications [8].

A blockchain is an append-only distributed ledger of transactions (e.g., payments) maintained by nodes of a p2p network. Nodes run a *consensus* protocol for ensuring consistent transaction ordering in the ledgers across the entire network. In Proof-of-Work (PoW) consensus—the prototypical blockchain consensus algorithm used by Bitcoin, Ethereum 1.0 and many other systems—transactions are packaged in to blocks that are regularly added to the blockchain through a process called mining. A block is proposed (mined) by a node (miner) through solving a computationally difficult cryptographic puzzle, for which the miner receives a monetary fee as reward. To prevent Sybil attacks, the cryptographic puzzle difficulty is set so high today that miners use custom hardware within highly optimized mining farm facilities. The high cost of mining pits the miners in direct competition with each other, with each miner vying to mine as many blocks as possible and maximize profits.

Miners resort to various strategies to increase their earnings, such as favoring locations with cheap electricity, or increasing hardware power consumption and cooling efficiencies of the mining farms [9], [15], [42], [44]. They also organize themselves in to mining pools, apportioning mining rewards across pool miners for a steadier income over time. However, a fundamental factor affecting rewards earned by a miner is the latency of block propagation in the p2p network. For a block mined by a miner to be included in the blockchain, the block must be propagated to other miners as quickly as possible through gossip over the p2p network. In the gossip process a mined block is immediately sent to the miner's neighbors, who then validate the block and forward the block to their neighbors and so on. Delayed block propagation causes the block to be 'forked' which nulls the reward for the block. Similarly, blocks mined by other miners must also reach a miner fast to prevent mining on a forked chain.

Propagating a block through a worldwide network of miners takes between 100s of milliseconds to a few seconds on popular blockchains today [18], [20]. Speed of light delay over the vast distances between miners, and the number of hops blocks have to be relayed over in a sparse p2p network topology are major factors contributing to the propagation delay. A fresh block is mined once every few seconds on many blockchains—notably Ethereum, and Ethereum Classic which have a 'block time' of 12 seconds on average [46]. The high delay in block propagation relative to block time means a speedup of even few 10s of milliseconds in a miner's network translates to significant gains in mining rewards for the miner over time [25].

With a general understanding that faster the network the higher are the rewards earned, miners often subscribe to high-speed (few Gbps) Internet access links and make use of low-latency private backbone networks of cloud providers when possible [5], [6]. Increasingly pool operators are also choosing private block relay networks (e.g., BloXroute [1], [29]) for a more efficient block dispersion. However, recent work has

shown that the dependence of mining rewards on propagation latency is more intricate than this [35]. Specifically, an honest miner that is well connected with other miners inadvertently creates efficient, low latency paths for other miners by acting as a centrally located bridge between the miners. However, to maximize the marginal gains in reward due to the network, it is important for a miner to have paths to other miners that are, on average, of a lower delay *relative* to the delays of paths between other miners. For example, if miners are arranged as a star topology with links of unit delay and uniform compute power across nodes, the central node receives a higher reward compared to the leaf nodes by including more blocks on the blockchain. On the other hand, on a complete graph topology with unit delay links and uniform compute power as before, all nodes receive the same reward. A node identically connected to other nodes in the two cases (i.e., the central node in the star topology and any arbitrary node in the complete graph topology: both have direct links to all other nodes) receives different rewards, as rewards depend not only on the node's own connections but also on how other nodes' connections. Thus, there is an inherent tension for a miner in increasing her own connectivity to the rest of the network while simultaneously ensuring that the connectivity between other miners do not significantly increase. A systematic research of this tension, and efficient connection policies to maximize marginal mining reward gain due to the network, have not been done to our best knowledge.

In this work, we formalize the p2p topology construction problem as a game between miners and present Cobalt, a decentralized policy for optimizing reward. We consider a simplified setting where only a single node chooses its connections, while the rest of the network's topology is fixed. We assume that the global topology of the p2p network is unknown to miners. We thus model the problem of optimizing rewards by the connections-deciding miner node as a Markov decision process (MDP) with no state and an action set with a combinatorial number of actions.

We derive the optimal neighbor selection policy using a combinatorial multi-armed bandit (MAB) approach [14]. In the MAB algorithm, the agent (miner) explores various candidate connection configurations, and gradually adapts its connections based on past experience to gain the most mining rewards. A key contribution of our work is a network coordinates based model for efficiently learning the MAB environment [19]. In this model, miners are assigned real-valued vectors from an Euclidean space, which capture the relative location of miners with respect to each other in the network. The coodinates are continuously updated based upon the reward feedback the agent receives from the environment. Thus, despite not having global knowledge of the network initially, we show that it is possible for an agent to learn about the network by just using the observed reward information.

To enable the deployment of MAB algorithm, we have built a simulator. To simplify the reward computation in the simulator, rather than simulating the actual mining process at each step of the MDP, we consider a computationally easier function that only depends on the pairwise shortest path lengths between miners. Importantly, our MDP reward function captures the property that a miner's mining gains depends on how small the shortest path lengths between the agent and other miners are relative to the shortest path lengths between other miners. Experimentally we show Cobalt outperforms or matches heuristics on diverse network settings.

II. RELATED WORK

P2P network design for optimizing mining rewards has remained a relatively under-explored topic in the community. The work that is closest to our is Perigee [34] which proposes an adaptive peer-selection algorithm for minimizing block propagation latency in the network. However, Perigee does not model the game-theoretic competition between miners. Subsequent works [11], [43] consider optimizing the network to maximize extractable value (MEV) from transactions. A number of prior works have exposed the impact of the network on mining [12], [26], [28], [37], [40], [47], [48]. While these works generally suggest that better network connectivity translates to higher mining rewards earned, the competitive effects of network connectivity and methods to optimize them have not been discussed. Other related works include KadCast [38] which proposes a Kadmila-based structured overlay for efficient block broadcast, and relay networks such as BloXroute [29] for transports blocks quickly across vast geographic distances.

The idea of network coordinates for p2p networks has been prominently explored in the network systems literature since the turn of the millenium, including distributed approaches to learn them [19], [32], [36]. More recently, a number of theoretical works have studied using low-distortion embeddings in finite metrics (i.e., over finite graphs) for various applications, e.g., sparse spanner construction [10], [13], [16], [21].

Game theory of blockchains, especially at the consensus layer, has received considerable attention. For example, Lewenberg et al. [33] use game theory to study how mining rewards can be shared across members of a mining pool. On the other hand, prior works have considered various network games outside the context of blockchains [24], [39]. Our work is the first (to our best knowledge) to consider network games in blockchains.

III. PROBLEM FORMULATION

Let us consider a complete directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of directed edges. Each node in the graph represents a mining server. The hash rate of the mining server v is denoted by H_v. We use $\boldsymbol{H}$ to denote the hash rate vector $\boldsymbol{H} := (H_v)_{v \in \mathcal{V}}$. A directed edge $(v_1, v_2) \in \mathcal{E}$ represents a (TCP) link between the nodes v_1, v_2.[1] The directed edge represents that node v_1 can send messages (e.g., transactions, blocks etc.) to v_2 as and when required by the protocol. The time take for a message sent from v_1 to reach v_2 along the link (v_1, v_2) is denoted by $l(v_1, v_2) \geq 0$.

---

[1]We assume if $(v_1, v_2) \in \mathcal{E}$ then $(v_2, v_1) \in \mathcal{E}$ for all $v_1, v_2 \in \mathcal{V}$.

For $(v_1, v_2) \notin \mathcal{V}$, we let $l(v_1, v_2)$ denote the latency of sending a message from v_1 to v_2 had there been a link between v_1 and v_2. We refer to $L := [l(v_1, v_2)]_{v_1, v_2} \in \mathbb{R}_+^{\mathcal{V} \times \mathcal{V}}$ as the latency matrix. In practice, the latency matrix is nearly a symmetric matrix; asymmetry in the latency between two nodes can arise occasionally if the forward and reverse IP paths between the nodes have significantly different lengths. We assume that the hash rate of the nodes is publicly known by all the nodes. For instance, by inspecting the frequency with which blocks are mined by different miners on the blockchain, one can estimate the relative hash power of different nodes. Each node v has knowledge of latency $l(v, u)$ between itself and other nodes $u \in \mathcal{V}$, but does not have knowledge of the latency between other pairs of nodes. To get an estimate of the latency $l(v, u)$ to a node u, the node v can simply issue a ping to u and measure the round-trip-time.

Let $t \in \mathbb{N}$ denote a time slot (or a round number). The duration of each time slot is a design parameter, and could be multiple minutes or hours long depending on the block production rate in the blockchain. Based on the information available to each node v, node v must decide on a certain number of nodes to connect to for the running the PoW consensus protocol. We assume each node can only connect to a maximum of $\delta \in \mathbb{N}$ nodes. To make a new connection, a node first sends a connection request to the recipient who either accepts or rejects the request. Each node can have at most $\gamma \in \mathbb{N}$ incoming connections at any point in time. The numbers δ and γ are public information and part of the consensus protocol specification.

If every node determines the connections strategically in every time slot to increase its expected reward, it leads to a multi-agent game problem. Since games are generally more difficult to solve, we first simplify the problem by assuming that the network topology is fixed and only one node is strategically picking the connections to the other nodes. This leads to a more tractable Markov decision process, which we describe next.

A. Markov Decision Problem Formulation

We now formulate the node connection problem as an MDP from the point of view of an individual node with the rest of the network being fixed. As we will see, this is a MDP with no state and has a combinatorial action set.

Let $v_0 \in \mathcal{V}$ be the node that needs to determine the connections to other nodes. Let $\mathcal{F} \subset \mathcal{E}$ denote the links in $\mathcal{G}$ that have been made by nodes other than v_0, i.e., $\mathcal{F} = \{(u, u') \in \mathcal{E} : u \neq v_0\}$. The edges in $\mathcal{F}$ denote the fixed part of the network. The set of edges in $\mathcal{F}$ will be augmented with the connections picked by v_0.

The MDP is parametrized by the hash rate vector $\boldsymbol{H}$, the roundtrip delay matrix L, and the parameters δ and γ. We let the complete parameters be denoted by $\theta := (\boldsymbol{H}, L, \delta, \gamma)$.

The action of the node v_0 at time t is a set $a_{S,t} = \{u_{t,1}, u_{2,t}, \ldots, u_{\delta,t}\}$ of nodes in $\mathcal{V} \setminus \{v_0\}$ to whom v_0 connects to.

Once the action is picked, the neighbors of the node v_0 are determined for the time t. The set of edges in the network at time t is $\mathcal{F}_t := \mathcal{F} \cup \{(v_0, v) : v \in \boldsymbol{a}_t\} \subset \mathcal{E}$ over which the consensus protocol runs.

Let $\mathcal{P}_t(v, v')$ denote the set of paths from node v to v' for $v' \in \mathcal{V} \setminus \{v_0\}$. Each path $P \in \mathcal{P}_t(v, v') \subset \mathcal{F}_t$ is an ordered collection $((v, u_1), (u_1, u_2), \ldots, (u_k, v'))$. Define $\bar{l}(v, v')$ as

$$\bar{l}(v, v') = \min_{P \in \mathcal{P}_t(v, v')} \sum_{(u, u') \in P} l(u, u'), \qquad (1)$$

which measures the latency for messages starting from node v to reach node v'.

We let $\mathbb{1}_\mathcal{V}$ denote a vector of all 1s of size $|\mathcal{V}|$. Let $\mathcal{U}_v$ denote a collection of subsets of nodes with aggregate hash power greater than 90% of the total hash power in the network as follows:

$$\mathcal{U}_v(\mathcal{F}_t; \theta) = \left\{ \mathcal{U} \subset \mathcal{V} \setminus \{v\} : \sum_{u \in \mathcal{U}} H_u \geq 0.9 \mathbb{1}_\mathcal{V}^T \boldsymbol{H} \right\}. \qquad (2)$$

Let $A_v(\mathcal{F}_t; \theta)$ denote the 90 percentile network latency for node v. This is determined by

$$A_v(\mathcal{F}_t; \theta) = \min_{U \in \mathcal{U}_v(\mathcal{F}_t; \theta)} \max_{u \in U} \bar{l}(v, u). \qquad (3)$$

Define $\bar{A}(\mathcal{F}_t; \theta)$ as the average 90-th percentile latency of the network:

$$\bar{A}(\mathcal{F}_t; \theta) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} A_v(\mathcal{F}_t; \theta). \qquad (4)$$

For a node v, the amount of cryptocurrency earned during the mining process is fairly complicated function of the network topology. To keep the problem tractable, we pick a simple reward function for the node that attempts to minimize the 90 percentile network latency in comparison to the average 90 percentile network latency of all the nodes. Accordingly, the reward of the node v_0 is given by

$$R_v(\boldsymbol{a}_t) = -\beta \frac{A_v(\mathcal{F}_t; \theta)}{\bar{A}(\mathcal{F}_t; \theta)}, \qquad (5)$$

where β is a positive constant that depends on the blockchain protocol and the length of the timeslot. Eq. (5) captures the intuition that a miner receives greater than its fair share of rewards if the miner has network connectivity that is on average better than the network connectivity of other miners. In practice the reward during a time slot can be computed by measuring the cumulative mining fees earned over the duration of the time slot. The goal of the node v_0 is to pick its neighbors at every time step strategically so that the long term average expected reward is maximized, where the randomization (if any) comes from the miner's neighbor selection policy. Thus, we arrive at an MDP with no state and a combinatorial number of actions.

(a) Base network   (b) Player v connects to node b (c) Player v connects to node c

Fig. 1: Example of a network showing the effect of player v's action on reward.

B. Key Issues

From a node's perspective, the hash rate vector of the network and the parameters δ and γ are public knowledge and therefore known to the node. However, each node v can only observe the latency $l(v, \cdot)$ to all other nodes in the network. Thus, the node does not know the latency other nodes in the network are observing. Moreover, the edges associated with the other nodes in the network are also unknown to the node v. Thus, each node v has partial knowledge of L and $\mathcal{F}$.

Since we have a stateless MDP and each node does not know some graph and network parameters, we arrive at a "multi-agent combinatorial bandit problem". If every node is strategic in the network and picks their neighbors strategically, then the multi-agent combinatorial bandit problem is very difficult to solve. Thus, for simplicity, we assume that all nodes except v_0 are non strategic, due to which the network edges $\mathcal{F}$ is fixed (though node v_0 still has only partial knowledge of $\mathcal{F}$). The goal of this paper is to solve the combinatorial bandit problem associated with node v_0's optimization problem.

C. Computation of $A_v(\mathcal{F}_t; \theta)$

Note that a naïve computation of $A_v(\mathcal{F}_t; \theta)$ (Eq. (3)) requires combinatorial number of operations. In this section, we identify a simple algorithm which computes the value $A_v(\mathcal{F}_t; \theta)$ in polynomial time.

Algorithm 1 Computing $A_v(\mathcal{F}_t; \theta)$

1: **Input:** v, graph $\mathcal{F}_t$, latency matrix L, hash power $\mathbf{H}$
2: **Output:** $A_v(\mathcal{F}_t; \theta)$
3: Compute $\bar{l}(v, v')$ for all $v, v' \in \mathcal{V}$ in graph $\mathcal{F}_t$
4: Sort nodes in $\mathcal{V} \setminus \{v\}$ as $(v_1, v_2, \ldots, v_{n-1})$ such that $\bar{l}(v, v_1) \leq \bar{l}(v, v_2) \leq \ldots \bar{l}(v, v_{n-1})$. Here, $n = |\mathcal{V}|$.
5: Initialize $k \leftarrow 1, s_k \leftarrow \bar{l}(v, v_k)$.
6: **while** $H_v + \sum_{i=1}^{k+1} H_{v_i} \leq 0.9 \mathbb{1}_\mathcal{V}^T \mathbf{H}$ **do**
7:     $s_{k+1} \leftarrow \max\{s_k, \bar{l}(v, v_{k+1})\}$
8:     $k \leftarrow k + 1$
9: **end while**
10: **return** $\max\{s_k, \bar{l}(v, v_{k+1})\}$

Lemma 1: Algorithm 1 computes $A_v(\mathcal{F}_t; \theta)$ as defined in Eq. (3), and does so in polynomial time.

IV. MOTIVATION

Before we present our algorithm, we motivate the problem with a simple, toy example. Consider a network of 5 nodes as shown in Fig. 1a, with node v being the player node. The solid edges show existing links in the network, while dotted edges are potential links that can be added by the player if it chooses to. For each link, Fig. 1a also indicates the latency of the link. For simplicity, suppose player v is interested in choosing $\delta = 1$ new connection in the network. Further, in this example we define A_v as the 100-th percentile latency for any node v, instead of the 90-th percentile latency as in Eq. (3). The precise hash power distribution across nodes therefore does not matter for this example. The reward earned by the player is as in Eq.(5) with $\beta = 1$.

Fig. 1b and 1c show two possible actions for player v. In Fig. 1b node v makes the connection to node b, while in Fig. 1c node v makes the connection to node c. If v connects to b, the 100-th percentile latency values of nodes a, b, c, d, v are $5, 4, 5, 4, 3$ respectively, which gives a reward of -0.71 to v. On the other hand, if v connects to c, the 100-th percentile latency values of nodes a, b, c, d, v are $5.7, 5.7, 4.7, 4, 3$ respectively, which now provides a reward of -0.65 to the player.

Thus, the amount of reward earned by the player depends on the choice of connections made by the player. The problem is non-trivial even if the player has to choose just a single new connection. Intuitively, in Fig. 1 no matter what action node v chooses, its 100-th percentile latency remains the same. However, the choice of connections crucially impacts the 100-th percentile latency values of the other nodes in the network. For example, in Fig. 1b, while v's connection to b does not provide any benefit to v it significantly shortens the path for b to reach a, which ultimately reduces the reward earned by v. In Fig. 1c, however, by connecting to c, node v ensures b's path length to a remains large thus incurring a greater reward than in Fig. 1b. The key observation with this example is that if a node is already well-connected in the network, then any additional connections made by the node must not significantly help other nodes in reducing their latency. With larger sized networks, non-uniform hash power distribution, and $\delta > 1$, the problem becomes even more complex.

V. COBALT DESIGN

We now present Cobalt, a combinatorial bandit algorithm for choosing efficient actions by the player. Cobalt is a fully decentralized algorithm that miners can use to determine effective peers to connect with to maximize their mining rewards. The algorithm is inspired by a line of theoretical works in the combinatorial bandit literature [14], [22], [23], [27], [30], [41]. However, prior works typically assume a "simple" and known underlying model (or, oracle) for the bandit environment which allows for efficient algorithms and tractable analysis. In our case, it is a priori not clear whether the blockchain p2p networking environment seen from the perspective of an individual miner can be fit reasonably to any of the existing combinatorial bandit models in the literature. A core contribution of Cobalt is an efficient model based on the idea of network coordinates [17], [19], [31], [32], [36], which we claim can be used to learn an effective representation of the bandit environment from past observations. The network-coordinate model can then be used to estimate rewards obtained through playing different actions, using which the best action to take can be inferred.

In a network-coordinate model, each node $v \in \mathcal{V}$ of the networks is assigned a real-valued vector of coordinates $x_v \in \mathbb{R}^k$ in an Euclidean space (e.g., $k = 5$ in our evaluations). A node's coordinates represents its location in the network, relative to other nodes. For instance, two nodes with a low round-trip-time delay to each other should have coordinates that are close to each other in Euclidean distance, and vice-versa. Network coordinates are real-valued which makes them amenable to be learned via gradient descent techniques within auto-differentiation packages. Thus, purely based on past bandit action-reward information observed by a player, our model creates an estimate of the network structure and computes effective actions the player can take. While IP addresses of miners may be known publicly (e.g., through blockchain monitoring services [3]), an IP address does not reveal fine-grained information about a miner's relative position in the network. Moreover, IP addresses can easily be spoofed and do not reveal the topological structure of the network. We remark that alternative network models are possible: for instance, we can try to learn the miner-to-miner link latency directly. This approach requires $O(n^2)$ parameters ($n = |\mathcal{V}|$), as opposed to the $O(n)$ parameters in our proposed network coordinates model. Learned link latency values also do not reveal topological information.

Algorithm 2 presents an overview of Cobalt. At each time step, based on the current set of node network coordinates the algorithm computes the best action (function BESTACTION) the player can take. Upon playing the recommended action, the player receives a reward from the bandit environment. The player compares the reward obtained against its estimated reward for the action (function ESTIMATEREWARD), and uses the discrepancy between the two to revise its network coordinate model (function UPDATEMODEL). To avoid over-fitting, we follow an $\epsilon-$greedy schedule for exploring random

---

Algorithm 2 Algorithm template for Oracle at node v

1: Initialize: Oracle model and its parameters $\theta = [x_1, x_2, \ldots, x_n]$ where x_i is estimate of network coordinate for node i; exploration parameter ϵ;
2: **for** each time step t **do**
3:     Sample R uniformly randomly between $[0, 1]$;
4:     **if** $R < \epsilon$ **then**
5:         Choose action $a_v(t)$ randomly and observe rewards $R_v(t)$;
6:     **else**
7:         Choose $a_v(t) \leftarrow$ BESTACTION(θ) as the best action and observe reward $R_v(t)$;
8:     **end if**
9:     Predicted reward $\leftarrow$ ESTIMATEREWARD($\theta, a_v(t)$)
10:     Update Oracle model parameters: $\theta \leftarrow$ UPDATEMODEL($\theta, a_v(t), R_v(t)$)
11: **end for**

---

actions occasionally. We discuss the subroutines below.

BESTACTION(): For a candidate action a_v with network coordinates θ, the ESTIMATEREWARD(θ, a_v) function computes an estimate of the reward earned if the player plays action a_v. To compute what is the best action that can be played, the BESTACTION(θ) function simply exhaustively calls the ESTIMATEREWARD(θ, a_v) routine for all possible candidate actions a_v, and selects the action having the highest estimated reward. An exhaustive search is possible for small networks; for larger networks we can consider a random subsample of candidate actions or other local search heuristics to reduce complexity.

ESTIMATEREWARD(): To compute an estimate of the reward given a candidate action a_v, we first estimate the link latency between any two nodes $u, u' \in \mathcal{V}$ as $\hat{l}(u, u') = ||x_u - x_{u'}||_2$ where $|| \cdot ||_2$ is the L2 norm. Next, we compute an estimate of the network topology $\hat{\mathcal{E}}$. For the player v, we let links from v to nodes in a_v be the set of links incident to v in $\hat{\mathcal{E}}$. For nodes $u, u' \in \mathcal{V}, u, u' \neq v$, we heuristically estimate whether $(u, u') \in \hat{\mathcal{E}}$ based on θ. A simple heuristic, for instance, is to randomly decide whether $(u, u') \in \hat{\mathcal{E}}$. Here whether $(u, u') \in \hat{\mathcal{E}}$ is estimated once at time step 0, and fixed afterwards. More complex heuristics that also take in to account the current values θ of the network coordinates to estimate $\hat{\mathcal{E}}$ are also possible. We have adopted the random estimation heuristic in our evaluations. We also discuss advantages of knowing additional information about the true topology in our experimental results. With both the network topology $\hat{\mathcal{E}}$ and link latencies estimated, we can compute the 90-th percentile latency A_u for all $u \in \mathcal{V}$ in the graph $(\mathcal{V}, \hat{\mathcal{E}})$ and hence the reward estimate $\hat{R}_v(a_v)$.

UPDATEMODEL(): The last step during a round is to update the network coordinate values θ by comparing the estimated reward against the actual reward obtained. We define a loss function $\lambda(t) = (R_v(t) - \text{ESTIMATEREWARD}(\theta, a_v(t)))^2$, where $R_v(t), a_v(t)$ denote the reward obtained and action

Algorithm	Amsterdam		Atlanta		Shanghai		Tokyo	
	A_v	Reward	A_v	Reward	A_v	Reward	A_v	Reward
Random choice	205.097	-1.299	205.662	-1.106	208.901	-1.494	200.159	-1.229
Least Latency	204.104	-1.114	204.308	-0.9663	204.344	-1.079	199.852	-1.211
Most Hash Power	204.11	-1.089	204.457	-1.022	208.572	-1.332	204.913	-1.433
Cobalt(Ours)	204.105	-1.04	204.298	-0.9674	205.003	-1.094	203.246	-1.16

TABLE I: Average Reward obtained under real world hash power distribution by player nodes in Europe, North America and Asia respectively.

taken during time step t, respectively. The coordinates are then updated as

$$\theta(t+1) \leftarrow \theta(t) - \eta \nabla_\theta \lambda(t), \quad (6)$$

where $\eta > 0$ is a step-size parameter. Note that computing $\nabla_\theta \lambda(t)$ requires taking the gradient of ESTIMATERE-WARD$(\theta, a_v(t))$ with respect to θ, which in turn requires computing the gradient of the 90-th percentile latency A_u for $u \in \mathcal{V}$. This can easily be done as A_u for any u can be expressed as a sum of link latencies (estimated via network coordinates) along the 90-th percentlie latency path which is differentiable with respect to θ.

VI. SIMULATION SETUP AND EVALUATION

In this section, we illustrate the effectiveness of Cobalt through experimental evaluation. We build a custom simulator on Python following the model described in §III-A.

A. Dataset Used

A custom dataset has been generated using publicly available Ethereum blockchain data. Training dataset was created by selecting the major Ethereum Mining pools responsible for mining the most blocks, with their available network hash rates, and their estimated server locations [2], [4]. The round trip times have been collected from publicly available ping data for major cities based on where these Ethereum nodes were estimated to be located [7]. Further synthetic distributions of hash powers have also been added to the training data, (we consider uniform hash rates as well as those drawn from an exponential distribution, apart from the true values). The ground-truth topology of the network is assumed to be static, and has been abstracted from the previous blocks of the Ethereum blockchain network [4].

B. Baselines

We consider the following baselines in our evaluations:
- Random Selection: In this, a playing node selects four connections randomly. This is the standard practice for a current node while connecting to a blockchain network.
- Least Latency: This heuristic involves a player node simply choosing the 4 neighbors that provide the lowest link latencies.
- Most Hash Power: Similar to LeastLatency, the player node selects 4 nodes that provide the maximum hash rate.

We compare the above baselines to our slightly more strategic approach that takes a look at the 90 percentile network latency using a multi-armed bandit approach.

C. Implementation Details

We run the experiment for $T = 300$ rounds. We restrict the number of allowable connections to 4, that are chosen according to our neighbor selection policy. Here, $\beta = 1$ and exploration parameter $\epsilon=0.1$. For a single node v_0, our action can only be to either connect to a new node after disconnecting from an existing node, or to keep the connection with an existing node, such that the total number of connections does not exceed 4 at any point of time. We initialize our network coordinates for the nodes randomly, and assume a random topology for the Oracle as described in §V. The coordinate system used here is a five dimensional one. We use the coordinates for estimating reward as described in §V.

D. Results

Table I displays our results for the listed method compared to baseline heuristics. We observe the results for different initializations of v_0 being the cities Amsterdam, Atlanta, Shanghai and Tokyo which are some of the major hubs for mining Ethereum blocks in the North American, European and Asian continents respectively.

We perform detailed experiments on player nodes Shanghai and Amsterdam, where Amsterdam is considered a "well connected" node and Shanghai, a "poorly connected" node.

1) Real world hash: In current blockchain networks, we observe that most of the hash power is concentrated in a few key regions (greater than 50% in the United States and Europe for Ethereum nodes [3]) , whereas the remainder is distributed among other scattered nodes. As such, a node located in a region without this concentration of hash power (such as Asia), may face many challenges to improving their mining rewards as compared to a node located in a well connected region (such as North America). One of the solutions of this problem in a topology like this for a node wishing to increase their rewards without heavily investing into additional infrastructure would be to shift away from randomly selecting their neighbors and picking their connections more strategically, by following some kind of policy to choose the nodes. I affirms that this is indeed a natural next step, as the rewards obtained by random selection are always suboptimal.

In this topology motivated by the real world, Fig. 2 describes the actual rewards obtained by a player as well as the prediction curve by our algorithm on both well connected and poorly connected players, and we see that for a complex network, our model is able to successfully learn the optimal action to take to converge to the best rewards.

(a) Player node Amsterdam

(b) Player Node Shanghai

Fig. 2: Prediction estimates of reward obtained by well-connected and poorly connected players and their actual rewards obtained.

(a) Player node Amsterdam

(b) Player Node Shanghai

Fig. 3: Reward obtained by well-connected and poorly connected players v/s random selection under uniform hash power.

2) Uniform hash and exponential hash: Apart from the real world scenario, we also look at other variations of topologies, starting with the ideal case, where we assume that all nodes have equal hash power. Here too (Fig. 3), it is reasonable to expect to outperform random selection for both well connected and poorly connected nodes, simply by virtue of the playing node being unable to select the optimal arm each time. However, here as our problem has been simplified, the only parameter that matters to the playing node becomes the link latencies, and the optimal action would be to connect to the neighbors closest to it. Indeed, our results reflect this case, and the best action selected by our algorithm eventually converges to the least latency arm.

A slightly more complex example would be if we vary the hash rate. Here, we use data that contains nodes with hash powers that have been drawn from an exponential distribution. Fig. 4 suggests that for a playing node, simple heuristics might be a better choice from the perspective of ease of use, however Cobalt is very competitive in terms of the rewards obtained. Picking the least latency heuristic might be the cheapest in case of a network where there is a certain degree of confidence that the hash rates are unequal, but not very disparate from each other. For such a kind of network topology, a prudent approach could be to consider a simpler heuristic.

3) Role of topology estimate in Oracle: To compute a reliable estimate of the reward function given a candidate action, Algorithm 2 requires an estimate of the underlying network topology. In the results so far, we have assumed a random estimate of the topology.

From the perspective of a playing node, knowing the quality of active links between its neighbors is very useful in order to estimate the delays between the blocks being received by this node in the blockchain network, but typically, the underlying topology of the network is unknown to a player node, as a result learning a good action makes more sense. Fig. 5 shows that our algorithm learns improved policies if the topology estimate used in the Oracle matches with the actual topology of the network.

4) Interpreting the network coordinates: Fig. 6 shows the t-SNE visualization [45] of the network coordinates that have been estimated by our model. We start with a random model, however as time passes, we observe our model trying to mimic the topology of the ground truth coordinates. For instance, we observe that even though the coordinates for Shanghai (shown by the large green point in Fig. 6) starts out being the middle, as time progresses the Shanghai's coordinates tends to move out to the periphery which is consistent with Shanghai's remote location relative to other nodes in our dataset.

(a) Player node Amsterdam

(b) Player Node Shanghai

Fig. 4: Reward obtained by well-connected and poorly connected players for all heuristics vs chain bandits for an exponentially distributed hash power setting

(a)

(b)

Fig. 5: Reward and error curves under random and exact topology estimates in the Oracle. Random topology estimate is referred to as 'perturbed' in the legend.

VII. CONCLUSION

In this paper, we have introduced Cobalt, a network coordinates based model that maximizes mining rewards earned by a miner through careful choice of neighbor connections on the P2P topology. We modeled the problem as a combinatorial bandit problem in which a node has only partial knowledge of the rest of the network and show that it is useful for a playing node to consider a careful selection of neighboring nodes instead of randomly choosing their connections.

A. Future Work

Our future work involves expanding this to a dynamic system, where all nodes are allowed to choose their neighbors and we attempt to understand the equilibrium of this game problem.

Another future direction is the development of other network topology estimates for the oracle. Our current model uses a random estimate, however others (such as a coordinate dependent topology estimate) and their impact on the model would be an interesting study.

REFERENCES

[1] Bloxroute. https://bloxroute.com/.
[2] Ethereum mining locations. https://investoon.com/charts/mining-map/eth.
[3] Ethernodes. https://ethernodes.org/.
[4] Etherscan. https://etherscan.io/.
[5] How to build an Ethereum mining pool. https://medium.com/dragonfly-research/how-to-build-an-ethereum-mining-pool-6be356520b7a.
[6] The secret weapon f2pool used to tackle its uncle rate. https://medium.com/bloxroute/the-secret-weapon-f2pool-used-to-tackle-its-uncle-rate-1ecb6fe47ef8.
[7] Wondernetwork ping statistics. https://wondernetwork.com/.
[8] The crypto market crashed. they're still buying bitcoin., 2022. https://www.nytimes.com/2022/08/02/technology/crypto-bitcoin-maximalists.html.
[9] This map shows the best states for bitcoin mining, 2022. https://www.cnbc.com/2021/09/30/this-map-shows-the-best-us-states-to-mine-for-bitcoin.html.
[10] I. Abraham, Y. Bartal, J. Kleinberg, T.-H. Chan, O. Neiman, K. Dhamdhere, A. Slivkins, and A. Gupta. Metric embeddings with relaxed guarantees. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 83–100. IEEE, 2005.
[11] K. Babel and L. Baker. Strategic peer selection using transaction value and latency. In *Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security*, pages 9–14, 2022.
[12] T. Cao, J. Decouchant, J. Yu, and P. Esteves-Verissimo. Characterizing the impact of network delay on bitcoin mining. In *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, pages 109–119. IEEE, 2021.
[13] T.-H. H. Chan, M. Li, L. Ning, and S. Solomon. New doubling spanners: Better and simpler. *SIAM Journal on Computing*, 44(1):37–53, 2015.
[14] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International conference on machine learning*, pages 151–159. PMLR, 2013.
[15] S. Chow and M. E. Peck. The bitcoin mines of China. *IEEE Spectrum*, 54(10):46–53, 2017.
[16] V. Cohen-Addad, A. Filtser, P. N. Klein, and H. Le. On light spanners, low-treewidth embeddings and efficient traversing in minor-free graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 589–600. IEEE, 2020.

(a) Initial Network Coordinates

(b) Final Network Coordinates

Fig. 6: t-SNE visualization of initial and final Network Coordinates for player Shanghai

[17] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Practical, distributed network coordinates. *ACM SIGCOMM Computer Communication Review*, 34(1):113–118, 2004.

[18] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, et al. On scaling decentralized blockchains. In *International conference on financial cryptography and data security*, pages 106–125. Springer, 2016.

[19] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. *ACM SIGCOMM Computer Communication Review*, 34(4):15–26, 2004.

[20] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[21] A. Filtser. Hop-constrained metric embeddings and their applications. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 492–503. IEEE, 2022.

[22] Y. Gai, B. Krishnamachari, and R. Jain. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *2010 IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN)*, pages 1–9. IEEE, 2010.

[23] Y. Gai, B. Krishnamachari, and R. Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking*, 20(5):1466–1478, 2012.

[24] A. Galeotti, S. Goyal, M. O. Jackson, F. Vega-Redondo, and L. Yariv.

Network games. *The review of economic studies*, 77(1):218–244, 2010.

[25] Y. Gao, J. Shi, X. Wang, Q. Tan, C. Zhao, and Z. Yin. Topology measurement and analysis on ethereum p2p network. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7. IEEE, 2019.

[26] A. E. Gencer, S. Basu, I. Eyal, R. v. Renesse, and E. G. Sirer. Decentralization in bitcoin and ethereum networks. In *International Conference on Financial Cryptography and Data Security*, pages 439–457. Springer, 2018.

[27] S. Gupta, S. Chaudhari, G. Joshi, and O. Yağan. Multi-armed bandits with correlated arms. *IEEE Transactions on Information Theory*, 67(10):6711–6732, 2021.

[28] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey. Measuring ethereum network peers. In *Proceedings of the Internet Measurement Conference 2018*, pages 91–104, 2018.

[29] U. Klarman, S. Basu, A. Kuzmanovic, and E. G. Sirer. bloxroute: A scalable trustless blockchain distribution network whitepaper. *IEEE Internet of Things Journal*, 2018.

[30] B. Kveton, Z. Wen, A. Ashkan, and C. Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In *Artificial Intelligence and Statistics*, pages 535–543. PMLR, 2015.

[31] J. Ledlie, P. Gardner, and M. I. Seltzer. Network coordinates in the wild. In *NSDI*, volume 7, pages 299–311, 2007.

[32] J. Ledlie, P. Pietzuch, and M. Seltzer. Stable and accurate network coordinates. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 74–74. IEEE, 2006.

[33] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pages 919–927. Citeseer, 2015.

[34] Y. Mao, S. Deb, S. B. Venkatakrishnan, S. Kannan, and K. Srinivasan. Perigee: Efficient peer-to-peer network design for blockchains. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 428–437, 2020.

[35] Y. Mao and S. B. Venkatakrishnan. Less is more: Fairness in wide-area proof-of-work blockchain networks. *arXiv preprint arXiv:2204.02461*, 2022.

[36] T. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 170–179. IEEE, 2002.

[37] S. Park, S. Im, Y. Seol, and J. Paek. Nodes in the bitcoin network: Comparative measurement study and survey. *IEEE Access*, 7:57009–57022, 2019.

[38] E. Rohrer and F. Tschorsch. Kadcast: A structured approach to broadcast in blockchain networks. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 199–213, 2019.

[39] T. Roughgarden. Algorithmic game theory. *Communications of the ACM*, 53(7):78–86, 2010.

[40] M. Saad, V. Cook, L. Nguyen, M. T. Thai, and A. Mohaisen. Partitioning attacks on bitcoin: Colliding space, time, and logic. In *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*, pages 1175–1187. IEEE, 2019.

[41] A. Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

[42] C. Stoll, L. Klaaßen, and U. Gallersdörfer. The carbon footprint of bitcoin. *Joule*, 3(7):1647–1661, 2019.

[43] W. Tang, L. Kiffer, G. Fanti, and A. Juels. Strategic latency reduction in blockchain peer-to-peer networks. *arXiv preprint arXiv:2205.06837*, 2022.

[44] M. B. Taylor. The evolution of bitcoin hardware. *Computer*, 50(9):58–66, 2017.

[45] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[46] D. Vujičić, D. Jagodić, and S. Ranđić. Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)*, pages 1–6. IEEE, 2018.

[47] L. Wan, D. Eyers, and H. Zhang. Evaluating the impact of network latency on the safety of blockchain transactions. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 194–201. IEEE, 2019.

[48] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou. Modeling the impact of network connectivity on consensus security of proof-of-work blockchain.

In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1648–1657. IEEE, 2020.