Optimizing Block Propagation in Bitcoin Network with Region-based Neighbor Selection Using Reinforcement Learning

***Abstract*—Bitcoin proved the potential of blockchain technology through decentralized, transparent, and immutable transactions. However, there are still challenges for fast and stable transitions. Optimizing block propagation times within the network is one of them. Prolonged propagation times can restrict efficiency, scalability, and security. This paper presents a novel approach to reducing block propagation time through leveraging reinforcement learning (RL) for the node's neighbor selection strategies. We implemented a Deep Q-Network (DQN) model in minimizing block receive times at each node, thereby impacting overall block propagation time. We used a model that defines node states based on latencies of outbound connections, which present the node's region. By evaluating this model through simulations using SimBlock, a robust Bitcoin network simulator, we observed a significant reduction in block propagation time approximately 30% for smaller networks and 20% for larger ones. Our analysis extended to node connections generated by our model and comparative evaluation against existing methodologies.**

***Index Terms*—Blockchain, Bitcoin, Reinforcement Learning, Deep Q-Network, Block Propagation**

I. Introduction

Bitcoin [1] is a pioneering digital currency that operates on a decentralized peer-to-peer network. Initially outlined in a 2008 whitepaper by an anonymous entity or group known as Satoshi Nakamoto, Bitcoin illuminated the immense potential of blockchain technology as a decentralized ledger system, ensuring transparency and immutability [2]. The transparent nature of its blockchain permits public scrutiny of all transactions, fostering trust, while its immutability structure guarantees that once information is recorded, it remains unchangeable. Although primarily utilized as a digital currency facilitating online transactions, Bitcoin extends its reach into diverse sectors such as supply chain management, and healthcare, leveraging its capabilities in innovative ways [3]. Additionally, the strength of Bitcoin lies in its decentralized nature, which removes the need for intermediaries, fostering direct peer-to-peer transactions and enhancing autonomy in the digital landscape.

In the expansive landscape of blockchain's possibilities, there are still numerous technical challenges. Reducing block propagation time is also an important problem for scalability and security issues. As new blocks are added to the chain, they must swiftly propagate across the network for validation by nodes. Longer propagation times can hinder the efficiency and speed of the blockchain ecosystem. They serve as a bottleneck, leading to scalability issues by limiting transaction processing and potentially causing delays in achieving consensus [4]. Moreover, prolonged propagation times can cause security risks such as double-spending by increasing the likelihood of temporary forks [5]. Solutions to tackle this challenge involve various approaches, including network optimization strategies [6], relay protocols [7], and neighbor selection algorithms [5], [8]. While these approaches have yielded commendable progress, there are still some limitations such as resource utilization and residual vulnerabilities in terms of security protocols [5].

In this study, we present a novel approach focusing on reducing block propagation time by neighbor selection strategy drawn by reinforcement learning (RL). Reinforcement learning is a branch of machine learning where an agent learns to make decisions through interaction with an environment to maximize rewards [9]. It is being increasingly utilized across various industries and applications due to advancements in algorithms, computational power, and the availability of vast amounts of data. With RL techniques, our primary goal is to reduce the block receive time at each node. Since block propagation time is the same as the block receive time of a node that received the block at last, we expected that this approach would lead to a reduction of block propagation time.

We constructed an RL model by setting each node's state as latencies of the node's 8 outbound connections. These latencies represent the outbound nodes' region. Node's actions are selecting one of these 8 outbound nodes to interchange with another randomly chosen node, or maintaining present status. The reward is computed by comparing a node's block receive time before and after an action. We evaluated this RL-driven approach for reducing block propagation time, by executing a simulation with SimBlock [10], a blockchain simulator that has the strength to simulate the Bitcoin network and evaluate block propagation time. Our result showed that it was reduced by about 30% for small networks and 20% for large networks. We also analyzed node connections made by our model and compared our results with other work.

The remainder of this paper is as follows. In section II, we describe the background of SimBlock which is a Bitcoin network simulator for evaluating block propagation time, and related work. In section III, we present the design and implementation of our RL model and integration with SimBlock. Section IV evaluates the performance of our model by measuring block receive time and block propagation time. Section V discusses the evaluation results and limitations.

Then, we conclude this paper with possible future work.

II. Background and Related Work

This section provides some background about SimBlock and reviews related work about neighbor node selection algorithms.

A. SimBlock

SimBlock [10] is an event-driven open-source blockchain network simulator that has the strength to observe the block propagation of the network. This simulator provides a realistic environment by manipulating block, node, and network parameters. Simblock uses the Bitcoin routing protocol to connect nodes. When executing Simblock, nodes are generated following the parameters such as the location of the node and the mining power of the node. For each node, its outbound nodes are randomly selected among the whole nodes. After the connection process is done for every node, the mining process begins. Considering the mining power, a node is selected as a mining node, and the mined block is propagated across the network. If the block height meets the parameter value in the configuration, the simulation ends. Varying the whole parameters, the following output can be observed through simulation:

- Block receive time of each node
- Fork information, block height, and ID
- Events and contents while simulation
  - Node connections
  - Block transmissions

The source code and detailed information for the parameters and output can be found in [11].

Currently, Simblock supports two types of consensus algorithms, Proof-of-Work (PoW) and Proof-of-Stake (PoS). For network configuration, it provides latency for the year 2015 and 2019 among 6 regions: North America, Europe, South America, Asia Pacific, Japan, and Australia. Region distribution for Bitcoin is also included. SimBlock has enabled numerous studies and they are described in [12]. The case studies presented here prove that SimBlock is a flexible and effective blockchain simulator by replicates blockchain scenarios.

B. Related Work

There are many studies focused on reducing block propagation time by enhancing neighbor selection algorithms. Fadhil et al. [13] and Owenson et al. [14] used clustering based on locality and ping time for neighbor selection, respectively. However, Sudhan et al. [15] claimed that these approaches have drawbacks in that they are vulnerable to an eclipse attack or network partition attack. To solve these problems, they proposed a neighbor selection algorithm based on the combination of geographical proximity and random selection. Wang et al. [16] used bandwidth-based neighbor selection. To prevent an eclipse attack, a node disconnects a neighbor if its bandwidth is less than a specific threshold and randomly selects a new neighbor. Jiang et al. [17] introduced an algorithm for neighbor selection that calculates a neighbor node's propagation ability with the neighbor's degree, local clustering coefficient, and mining power.

Some studies evaluated their neighbor selection algorithm using SimBlock. Aoki et al. [8] investigated proximity-based strategies for selecting neighboring nodes in a blockchain network. The authors proposed an algorithm in which each node rates its neighbor nodes based on the time between block generation time and the reception time of an inventory message. Using these node scores, each node regularly reselects all of its neighbor nodes. By experiment, they concluded that the block propagation time was minimum when 1 of the neighbor node was selected randomly. Their algorithm was compared to a fixed-neighbor network and it showed lower propagation times. While this algorithm successfully improved block propagation time, it has some drawbacks in that each node incurred a computational burden and it is vulnerable to eclipse attacks.

Matsuura et al. [5] introduced a neighbor selection method based on the node's regional information. They made their method robust to eclipse attacks by randomly selecting neighbor nodes considering the node's region. If a neighbor node's region is the same as the node, it is called an inside neighbor. Otherwise, it is called an outside neighbor. They evaluated the optimal number of the outside neighbors varying the number of the nodes. They also used Simblock and concluded that the block propagation time is shortest when the number of outside neighbors is 2 for the networks with 1500 or more nodes. In a small network with 500 nodes, it was effective when the number of outside neighbors was 1. They also compared their method to the method proposed in Aoki et al. [8] and showed that their method is superior to it. Compared to selecting neighbors randomly, the block propagation time was reduced by about 23% for the large network of 6,500 nodes and 32% for the small network of 500 nodes.

In contrast to this previous work, our proposed method has the advantage that each node can employ different strategies based on a trained model, and there is no need for complicated calculations for each node. This flexibility creates resistance to security attacks. Since our model also uses a random selection strategy, it is also resilient to eclipse attacks. Furthermore, our method can be easily applied to each node by using a trained model with the latencies of its neighbor nodes and removing the connection with the output node of the model.

III. Design and Implementation

In this section, we provide our work's design and implementation details. We have utilized the Deep Q-Network (DQN) model described in [18] as the foundational framework for our reinforcement learning (RL) approach. The environment in which this model operates is finely tuned to the Bitcoin networks. Table I describes the notations in our paper.

A. State-Action-Reward

- *State:* In our context, the state is defined as the set of latency of a node's outbound connections. Since the

TABLE I
NOTATION

Variable	Description
$S_i^{j,k}$	State of node *i* in epoch *j*, iteration *k*
$A_i^{j,k}$	Action of node *i* in epoch *j*, iteration *k*
$R_i^{j,k}$	Reward of node *i* in epoch *j*, iteration *k*
$BRT_i^{j,k}$	Block receive time of node *i* in epoch *j*, iteration *k*

default setting for the maximum number of outbound connections in Bitcoin Core [19] is 8, we fixed every state containing a set of 8 latency values, representing the outbound links of the node. The latencies signify the duration for data to travel through these particular links, offering crucial observations on network functionality.

- *Action:* The action space is constructed with precision aimed at reducing the block receive time of a node. Every action involves removing one among the 8 outbound nodes from a given node. Additionally, there exists a ninth action, denoted as *"stay"*, providing the choice to retain the present state without any alterations. Taken together, the action space specifies 9 unique options available for each node after the action of removing one of the given node's outbound nodes, a random node from the entire network is added as a new outbound node of the given node.
- *Reward:* The reward mechanism is designed to align with our objective, which is reducing the block receive time of each node to reduce the block propagation time. To achieve our goal, we utilize the difference between the block receive times before and after the execution of an action as the reward. When the action taken is *"stay,"* a theoretical reward of 0 is designated, indicating the anticipation that block receive times before and after the action will ideally remain the same, even though slight deviations will be observed in practical experiments. This design enables our model to find out the most efficient actions for reducing block receive time.

These components comprise the operational basis for our DQN model, aimed at boosting the efficiency and functionality of Bitcoin networks, primarily targeting the reduction of block propagation time.

B. Architecture

Our reinforcement learning approach is anchored by the core framework of our DQN model, illustrated in Fig 1. SimBlock acts as the interface facilitating the acquisition of states from each node within the Bitcoin network for the agent. Within its deep learning segment, the DQN model includes three Fully Connected (FC) layers. The input dimension of the model is set at 8, aligning with the 8 latency values of outbound nodes. Similarly, the output dimension includes 9 distinct elements, covering 8 possible actions to remove an outbound node, including the *"stay"* option. The agent utilizes the states of each node acquired from SimBlock to determine the actions of individual nodes. These actions are

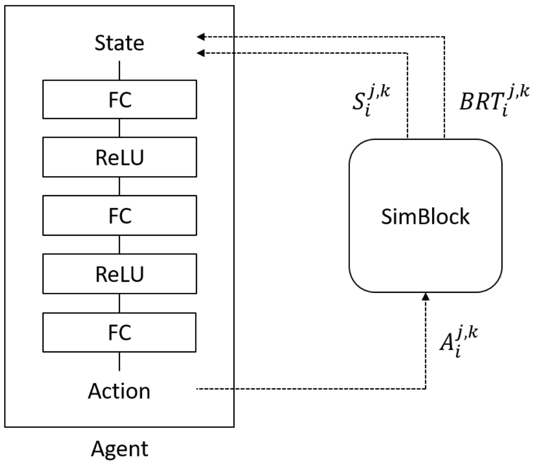

Fig. 1. Architecture of DQN Model

relayed to SimBlock, where modifications to the selected nodes' outbound connections are implemented as per the agent's decisions. The *"stay"* option maintains the node's outbound connections without any alterations. After executing actions and making adjustments to node connections, SimBlock returns the block receive time of each node. The reward $R_i^{j,k}$ is then computed as follows.

$$\begin{aligned} R_i^{j,k} &= BRT_i^{j,k} - BRT_i^{j,k-1} \\ if\ A_i^{j,k} &== "Stay",\ R_i^{j,k} = 0 \end{aligned} \quad (1)$$

Throughout the training phase, we ignored the future reward since we wanted to focus on immediate change in block receive time. Therefore, the update rule for Q-values with learning rate α is as follows.

$$\begin{aligned} Q(S_i^{j,k}, A_i^{j,k}) \Leftarrow Q(S_i^{j,k}, A_i^{j,k}) \\ + \alpha[R_i^{j,k} - Q(S_i^{j,k}, A_i^{j,k})] \end{aligned} \quad (2)$$

C. Workflow

Our detailed training process is shown in Fig 2.

1) *Network Initialization:* The training begins with SimBlock's execution, generating a foundational Bitcoin network. Its initial run generates a specific number of nodes and their connections, following the configuration and Bitcoin routing protocol. This network forms the dynamic environment within which the RL agent operates and learns.
2) *Simulation:* During each simulation, the Bitcoin network operates dynamically, with nodes randomly mining blocks. The simulation continues until the block height reaches the predefined threshold. At the end of the simulation, SimBlock offers the RL agent two vital pieces of information. It provides the current network state, including latency values of outbound connections for each node. It also reports the mean block receive

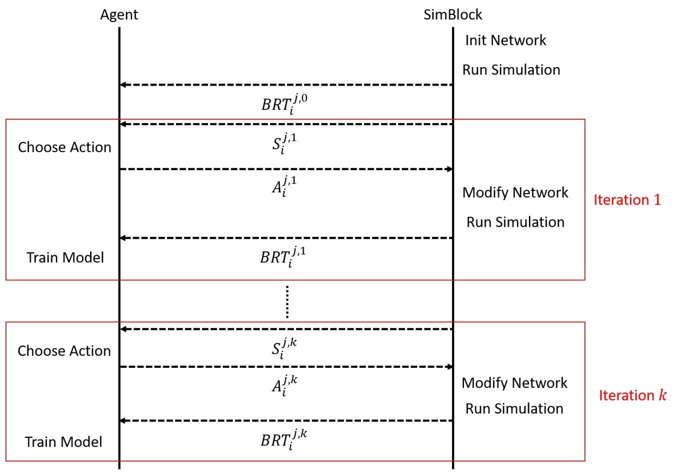

Fig. 2. Workflow in Epoch j

time for each node, a crucial metric for evaluating the network's performance.

3) *Action Selection:* Utilizing the obtained states, the RL agent makes decisions by selecting actions for each node, aiming to optimize the block receive time. These chosen actions are then translated into network-level adjustments within the SimBlock environment.
4) *Modifying Network:* Following the chosen actions, each node removes one of its neighbor nodes or maintains its neighbors. If a node removes one of its neighbors, a node from the entire network is randomly selected and added to its new neighbor node. Then, the simulation is conducted again with the modified network.
5) *Training Model:* By computing the reward of each node from given outputs, the model is trained. The process from *Action Selection* and *Traning Model* constitutes one iteration. This iteration is repeated to train the model with the given network. Furthermore, we defined the entire step as one epoch and repeated epochs multiple times to train the model with different networks.

IV. Experiment and Evaluation

A. Training Environment

Our experimental investigations took place on a computing platform powered by an Intel Core i7-9700 processor without a GPU. Within SimBlock, we configured the network to consist of 100, 250, 500, and 1,000 nodes, respectively. The block height was set to 500. For the training, we assumed that each node within the network employs a compact block relay strategy and is a churn node for convenience. Our experimental setup mirrored the region distribution observed in the Bitcoin network and latency among regions during the year 2019, provided by SimBlock. The region distribution is shown in Table II and an analysis for the latency can be found in [5].

For model training, We trained models with 100, 250, 500, and 1,000 nodes, respectively. To prevent overfitting, we stopped an epoch when more than 80% of the nodes chose

TABLE II
Region Distribution in Bitcoin 2019

Region	Distribution
North America	0.3316
Europe	0.4998
South America	0.009
Asia Pacific	0.1177
Japan	0.0224
Australia	0.0195

the *"stay,"* option in an iteration. Hyperparameters used for training are listed in Table II.

TABLE III
DQN Hyperparameters

Hyperparameter	Values
Learning rate	0.001
Optimizer	Adam
Batch size	256
Experience replay size	10,000
ϵ	$\max(ExploreRate^{Tries}, 0.01)$
ExploreRate	0.995
Hidden dimension	64
Epoch	100
Iteration	100

B. Evaluation for Block Receive Time

After training 4 models, we evaluated their performance by observing the block receive time of a node. Similar to the training process, we ran SimBlock over 100 epochs to evaluate them in different networks and get stable results. For each epoch, 100 iterations were used to estimate the convergence of the result. Since 500 blocks are generated for an iteration, we measured the mean of the block receive time of a node that was generated at first for an iteration. While we assumed that every node uses a compact block relay and are churn node in training, we changed them to the default setting defined in Simblock to evaluate the model in a more realistic environment.

We computed the mean of the block receive time of the first node for 100 epochs and it is shown in Fig 3. Our model demonstrated a distinct decrease in block receive time successfully. We could observe that the block receive time converged over iteration. The maximum and minimum block receive time over iterations are shown in Table IV. For the 100 nodes, block receive time was reduced by about 20%, and the rate was decreased for more number of nodes. For the 1,000 nodes, the rate of decrease reached about 10%. In other words, our model was effective for fewer nodes in improving block receive time.

TABLE IV
Comparison of Block Receive Time

Number of Nodes	Max(ms)	Min(ms)
100	1,060	854
250	1,086	931
500	1,133	987
1,000	1,170	1,045

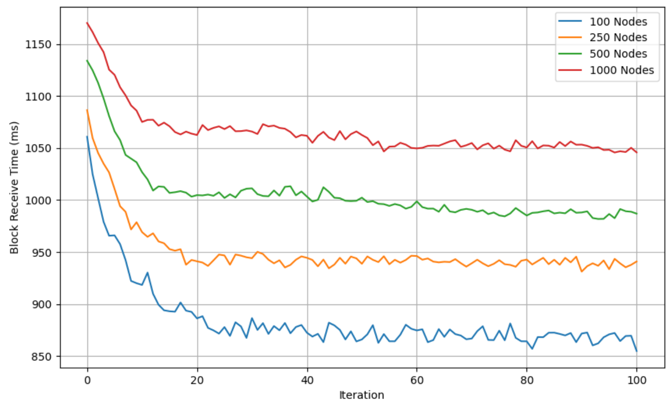

Fig. 3. Block Receive Time according to Number of Nodes

C. Evaluation for Block Propagation Time

Since our primary goal was to optimize the block propagation time, we measured the changes in the block propagation time with our models with the same method in the previous section. Fig 4 shows the results of our evaluation. We could observe that the overall patterns are similar to Fig 3, the block propagation time converges over iterations. This is reasonable because block propagation time is the same as the maximum block receive time for a block. The maximum and minimum block propagation times over iterations are shown in Table V. To sum up the results, the block propagation time was reduced by around 30%. It was 27% for the large network of 1,000 nodes, and 32% for the small network of 100 nodes.

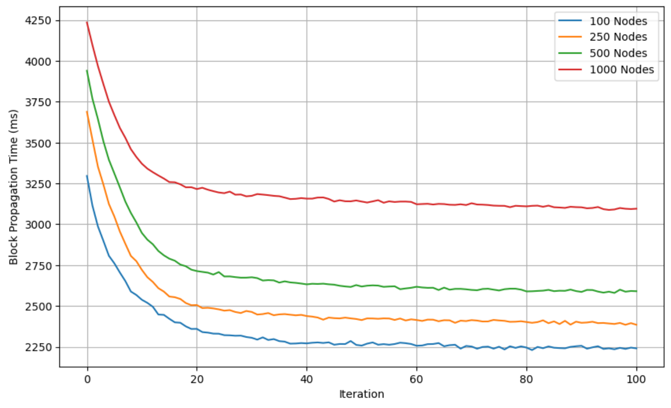

Fig. 4. Block Propagation Time according to Number of Nodes

TABLE V
COMPARISON OF BLOCK PROPAGATION TIME

Number of Nodes	Max(ms)	Min(ms)
100	3,295	2,230
250	3,688	2,385
500	3,939	2,580
1,000	4,235	3,088

D. Evaluation for Large Network

To measure the scalability of our model, we measured it on larger networks. We simulated a network with 10,000 nodes and applied the model trained with 1,000 nodes. The result is shown in Fig 5. The initial block propagation time was about 5,265ms while the minimum block propagation time was about 4,191ms. The block propagation time was reduced by about 20%. This result shows that our model improved the performance in a larger network, and it can be applied to the real Bitcoin network.

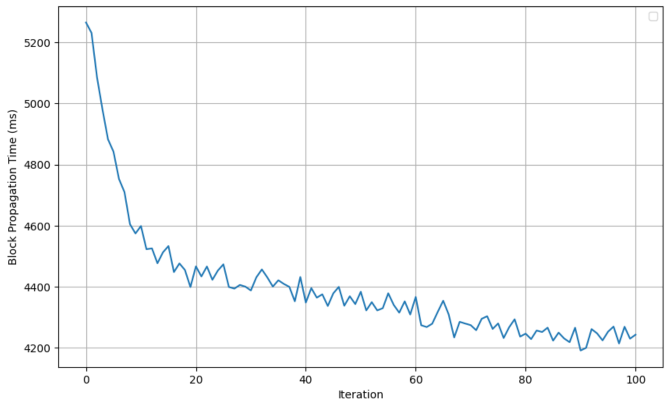

Fig. 5. Block Propagation Time for 10,000 Nodes

E. Analysis for Node Connections

While evaluating our model, we found that the model consistently selected the action labeled as *"stay"* for a significant proportion of nodes within the network during the later iterations. This observation suggests the presence of an optimal selection of outbound nodes. To analyze the node connections, we captured the snapshot of the node connections for the last iteration of the evaluation in the previous section. We summarized the result with the top three cases that have the largest percentages in Table VI. As we simulated a network with 10,000 nodes, there are 10,000 nodes in the table. It can be found that the distribution of the node's region perfectly matches with Table II.

For the nodes whose regions were set to North America, more than half of the nodes selected 5 nodes whose regions are also the same as their outbound nodes. The remaining 3 outbound nodes' regions were Europe. Including the top three results, it can be said that connecting with the same region is beneficial for the nodes in North America, avoiding too many connections. It is also interesting to see that the third result showed the North America nodes chose to connect to Asia Pacific node, unlike they chose the Europe node in the top two results. We could not find any examples that they chose 7 outbound nodes with the same region and 1 outbound node in Europe.

For the Europe nodes, 73.2% of the nodes chose all of its outbound nodes in the same region. This is reasonable because close to 50% of the nodes are in Europe. For past

TABLE VI
NODE CONNECTIONS WITH 10,000 NODES

Node's Region	**Outbound Node's Region**						**Count**	**Percentage**	**Total**
	North America	Europe	South America	Asia Pacific	Japan	Australia			
North America	5	3	0	0	0	0	1850	55.8%	3316
	6	2	0	0	0	0	907	27.4%	
	7	0	0	1	0	0	156	4.7%	
	...	...	...	...	...	...	403	12.1%	
Europe	0	8	0	0	0	0	3658	73.2%	4998
	2	6	0	0	0	0	1033	20.7%	
	3	5	0	0	0	0	214	4.3%	
	...	...	...	...	...	...	93	1.8%	
South America	3	3	2	0	0	0	5	5.6%	90
	2	6	0	0	0	0	5	5.6%	
	2	5	1	0	0	0	4	4.4%	
	...	...	...	...	...	...	76	84.4%	
Asia Pacific	2	2	0	3	1	2	196	16.7%	1177
	2	3	0	2	1	2	70	6.0%	
	3	1	0	2	2	3	67	5.7%	
	...	...	...	...	...	...	844	71.6%	
Japan	0	1	0	4	3	0	15	6.7%	224
	1	0	0	4	3	0	10	4.5%	
	0	1	0	6	1	0	9	4.0%	
	...	...	...	...	...	...	190	84.8%	
Australia	4	1	0	1	0	2	12	6.2%	195
	3	3	0	1	0	1	11	5.6%	
	3	4	0	0	0	1	10	5.1%	
	...	...	...	...	...	...	162	83.1%	

block propagation, propagating the block to close nodes is a straightforward strategy. This result also supports it. It is also important to point out that there was no example of a Europe node choosing 7 Europe nodes as outbound nodes. There might be some biased points while training.

For the nodes in South America, Japan, and Australia, there were too few samples to make meaningful observations. For the nodes in Asia Pacific, the top three results show that their outbound nodes are well-spread. However, similar to the three regions mentioned earlier, over 70% nodes were categorized in *"others"*, which means this can not be seen as a significant observation.

V. DISCUSSION

A. Comparison with Previous Work

Since [5] used a similar strategy that uses region information to reduce block propagation time and SimBlock for evaluation, we mainly compare this work to our work in this section. As mentioned in Section II, block propagation time was reduced by about 20$\sim$30% in that work. It reduced less when the network was large, and more when the network was small. This is the same with our results. In our work, we reduced block propagation time by 27% for the large network and 32% for the small network. It can be said that our work did not impressively improve by comparing the performance of previous work. However, by considering the environment where the simulator runs, just comparing the reduced rate could be not fair for evaluating overall performance.

Comparing the strategies, [5] used 6 inbound neighbors and 2 outbound neighbors to optimize the block propagation time. In our work, Table VI shows that our DQN model concluded that using different numbers of inbound and outbound neighbors following the node's region is optimal. In detail, the more the nodes are in the same region, it is beneficial to choose more inbound neighbors. Since we could not evaluate the nodes from the regions that have a small number of nodes accurately, we expect that there is room for improvement by choosing different numbers of inbound, and outbound neighbors in those regions.

B. Limitations

In this work, we assumed that nodes connect to a random node from the whole network. This indicates that when an action is executed and one of the outbound nodes is removed, it is likely to connect a new node from a region with a large number of nodes, in our work, it is Europe. However, in real Bitcoin protocol, nodes are not connected to a random node, they use DNS seeds to find a new node [20]. Moreover, we assumed that nodes have exactly 8 outbound neighbors but it can differ by node's configuration. This implies that our approach could not fit into a real Bitcoin network. To make an elaborate simulation of the Bitcoin network, more research on DNS seeds, and the degree of the nodes is needed.

We only used the latencies of neighbor nodes to train our DQN model and ignored other features such as the mining powers of nodes. There are two reasons why we used this approach. The first one is just to simplify the training process to boost the speed of training. The other one is that latency is the most straightforward feature and it is highly accessible by node to learn. We expect that using more features of the nodes as used in [17] can improve the performance of our model.

VI. Conclusion

We studied a comprehensive exploration to optimize block propagation time within Bitcoin networks. Through reinforcement learning, we have devised an innovative approach focused on enhancing the performance of blockchain systems, specifically targeting the reduction of block propagation time in Bitcoin. Our study utilized SimBlock, a robust blockchain network simulator that has strength in simulating Bitcoin networks and evaluating block propagation time.

The DQN model was implemented in an environment where states represent the latencies of outbound nodes for each node. Actions allow the selection of an outbound node to remove connection and rewards are computed from the difference in block receive times before and after action. We planned to reduce block receive time for each node and eventually reduce block propagation time through the iterative learning process.

The results from our experiments demonstrated the effectiveness of our approach. Through training and evaluation, we observed a significant improvement in block receive times with a reduction in block propagation time. The improvements were evident across networks of various sizes, from 100 to 1,000 nodes. Our model also showed that it is scalable by evaluation for 10,000 nodes. This outcome suggests that our model has the potential to reduce block receive time for individual Bitcoin nodes, consequently decreasing block propagation time when applied across multiple nodes. We also analyzed node connections after applying our model, and it gave room for improvements with possible future research.

In conclusion, our research underlines the potential of reinforcement learning in optimizing block propagation time within Bitcoin networks. As mentioned in the discussion section, our future work will focus on optimizing the model by adding other features. Moreover, we are planning to make models that can be applied to other blockchain networks.

References

[1] Nakamoto, Satoshi. ”Bitcoin: A peer-to-peer electronic cash system.” Decentralized business review (2008).

[2] Yaga, Dylan, et al. ”Blockchain technology overview.” arXiv preprint arXiv:1906.11078 (2019).

[3] Zheng, Zibin, et al. ”Blockchain challenges and opportunities: A survey.” International journal of web and grid services 14.4 (2018): 352-375.

[4] Shahsavari, Yahya, Kaiwen Zhang, and Chamseddine Talhi. ”A theoretical model for block propagation analysis in bitcoin network.” IEEE Transactions on Engineering Management 69.4 (2020): 1459-1476.

[5] Matsuura, Hiroshi, Yoshinori Goto, and Hidehiro Sao. ”New Neighbor Selection Method for Blockchain Network With Multiple Regions.” IEEE Access 10 (2022): 105278-105291.

[6] Chawla, Nakul, et al. ”Velocity: Scalability improvements in block propagation through rateless erasure coding.” 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2019.

[7] Otsuki, Kai, et al. ”Effects of a simple relay network on the bitcoin network.” Proceedings of the 15th Asian Internet Engineering Conference. 2019.

[8] Aoki, Yusuke, and Kazuyuki Shudo. ”Proximity neighbor selection in blockchain networks.” 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019.

[9] Arulkumaran, Kai, et al. ”Deep reinforcement learning: A brief survey.” IEEE Signal Processing Magazine 34.6 (2017): 26-38.

[10] Aoki, Yusuke, et al. ”Simblock: A blockchain network simulator.” IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2019.

[11] SimBlock, [Online]. Available: https://github.com/dsg-titech/simblock (Accessed: 12.14.2023)

[12] Shudo, Kazuyuki, et al. ”Blockchain Network Studies Enabled by SimBlock.” 2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2023.

[13] Fadhil, Muntadher, Gareth Owenson, and Mo Adda. ”Locality based approach to improve propagation delay on the bitcoin peer-to-peer network.” 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2017.

[14] Owenson, Gareth, and Mo Adda. ”Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network.” 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017.

[15] Sudhan, Amool, and Manisha J. Nene. ”Peer selection techniques for enhanced transaction propagation in Bitcoin peer-to-peer network.” 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2018.

[16] Wang, Ke, and Hyong S. Kim. ”FastChain: Scaling blockchain system with informed neighbor selection.” 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019.

[17] Jiang, Suhan, and Jie Wu. ”Taming propagation delay and fork rate in bitcoin mining network.” 2021 IEEE International Conference on Blockchain (Blockchain). IEEE, 2021.

[18] Mnih, Volodymyr, et al. ”Playing atari with deep reinforcement learning.” arXiv preprint arXiv:1312.5602 (2013).

[19] Bitcoin Core, [Online]. Available: https://github.com/bitcoin/bitcoin (Accessed: 12.14.2023)

[20] Tapsell, James, Raja Naeem Akram, and Konstantinos Markantonakis. ”An evaluation of the security of the bitcoin peer-to-peer network.” 2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData). IEEE, 2018.