UL-blockDAG : Unsupervised Learning based Consensus Protocol for Blockchain*

1st Swaroopa Reddy B
Electrical Engineering Department
Indian Institute of Technology Hyderabad
Hyderabad, India
ee17resch11004@iith.ac.in

2nd Sharma G V V
Electrical Engineering Department
Indian Institute of Technology Hyderabad
Hyderabad, India
gadepall@iith.ac.in

***Abstract*—In this paper, we propose a consensus protocol by considering the ledger as Directed Acyclic Graph (DAG) called *blockDAG* instead of chain of blocks. We propose a two step strategy for making the system robust to double-spend attacks. The first step is graph clustering algorithm based on spectral graph theory for separating the blocks created by the non-cooperating miners (attacker) in the blockchain network followed by the ordering algorithm based on the topological ordering of the blockDAG using the references included in block header. The first step is an unsupervised learning classification of the vertices of a graph into two classes. The simulation results show that the proposed clustering Algorithm based consensus protocol counter-attack the attacker's double-spending strategy by eliminating the attacker blocks created during attacking phase from the confirmed list of the blocks. In bitcoin's longest chain rule protocol, the ledger takes the chain of blocks and it operates with the overestimation of the network's end-to-end propagation delay which results in a low transaction throughput. Bitcoin protocol guarantees the security through *longest chain rule* but it suffers from the limited transaction scalability. The proposed consensus protocol works better for higher block creation rates which inturn improves the transaction throughput without compromising the security of the blocks from double-spending attack.**

***Index Terms*—Directed Acyclic Graph, blockDAG, Unsupervised Learning, Graph Clustering, Transaction Throughput**

I. Introduction

The bitcoin protocol in [1] consists of chain of blocks in which each block header includes a reference to the previous block in the form of block hash. While creating a new block, this protocol restricts the miner to refer the tip of the longest chain in the network to maintain consensus among all the nodes in the network. This protocol also involves the Proof-of-work (PoW) puzzle for creating a new block by the miners. To make the blockchain system robust to double-spend attacks, this PoW puzzle was made computationally hard by assigning a block creation interval of 10 minutes by setting a very difficult target value for solving PoW puzzle.

As per the measurement study conducted in [2], 10 minutes block interval was very high compared to network's end-to-end propagation delay. Because of this overestimation of the network's propagation delay, bitcoin's transaction throughput i.e. the number of transactions processed per second (TPS) was highly restricted.

Decrease in the block creation interval (or increase in the block creation rate) increases the Transaction throughput, but also decreases the security from the double-spend attacks. Accelerating the block creation rate causes propagation of more number of blocks in the network which causes some honest nodes do not have all the blocks created in the network and does not extend the longest chain. On the other hand, attacker who doesn't follow the bitcoin protocol will increase his chain will gain from the double-spend attack with high probability [3].

In this paper, we propose a consensus protocol for blockDAG structure of the ledger instead of chain of blocks. Each miner, while creating a new block will include reference to all it's predecessor blocks which are not referenced previously results in a DAG structure similar to blockDAG structure in SPECTURE [4] and PHANTOM [5]. The protocol consists of two steps. In the first step, each client (node) applies Graph clustering algorithm [6] for two clusters for seperating the blocks with less inter-connectivity with other blocks in the DAG. The blocks created by the attacker (miners who doesn't follow the protocol) do not have the well-connected blocks and the client can easily identify those blocks by applying the Graph clustering algorithms in [6]. In the second step, the ordering algorithm exclude the attacker blocks which include the double-spend transactions and order the honest blocks such that making the system robust to double-spend attacks. Our model ensures high transaction throughput as the block creation rate can be increased without compromising the security against double-spend attacks.

The simulation results show that, the attacker's double-spend strategy was counter-attack by the unsupervised learning based clustering and also show that throughput has increased drastically to thousands of transactions per second due to blockDAG structure of the ledger without compromising the fairness of the distributed system.

The rest of the paper is organized as follows. Section II describes the related work. Section III gives the system model and notations used in the protocol. Section IV describes the preliminaries related to graph clustering algorithm. Section V discuss the block partitioning algorithm followed by ordering

This research was funded by 5G Research and Building Next Generation Solutions for Indian Market Project, Department of Information Technology, Government of India.

algorithm. In section VI, we present the simulation results and discussion. In section VII, we conclude the paper and gave future directions of resarch.

II. Related Work

A GHOST(Greedy Heaviest-Observed Sub-tree) protocol, a variant of GHOST was used in Ethereum blockchain [7], was proposed in [8], where, instead of longest chain consensus rule, the path of heaviest subtree is chosen. However, the TPS for this protocol is still quite less [9] and susceptible to attacks as described in [10].

A mathematical model for optimal TPS in terms of block creation rate is shown in [11]. While the TPS is $100\times$ more than in bitcoin and Ethereum, However, the TPS is constrained by end-to-end propagation delay.

A SPECTRE protocol is proposed in [4] which builds with the concept of the blockDAG structure of the ledger. This protocol descibes the ordering of the blocks in the DAG based on the pair-wise voting procedure, where, it doesn't give the complete ordering of the blocks due to the Condorcet paradox [12]. While the TPS is much higher compared to [1] and [8], SPECTRE is not suitable for smart contract applications.

A PHANTOM protocol is proposed in [5] based on similar lines of SPECTRE , where , instead of pairwise ordering in SPECTRE, maximum k-cluster subDAG algorithm followed by ordering algorithm were proposed in PHANTOM. While, PHANTOM gives high transaction throughput and total ordering of the blocks, there is a trade-off between parameter *k* and confirmation times and also susceptible to liveness attack described in [13].

III. Model and Notations

In this work, we stick to the Bitcoin's model [1] in every aspect - transactions, blocks, PoW, information propagation in the P2P network [2]. The only difference is structure of the ledger. Where, instead of reference to a single block in bitcoin, newly created block references to more than one block in it's header and generates the DAG structure of the ledger.

We refer to the model specified in [4] and [5], where the parameters used in our framework are given in Table I and also we used the same DAG topology and mining protocol described in [4] and [5], where, the miners are instructed as follows

- Blocks created/recieved by the miners/nodes should be broadcast to all its neighbouring peers.
- while creating a new block, the miner should include references to all the leaf-blocks/tips (observed in it's local copy of the DAG) in it's block header.

IV. Preliminaries

The following definitions of spectral graph theory are used in our work.

Let $G = (C, E)$ be an directed graph with vertex set C and edge set E. We assumed that graph is unweighted.

TABLE I: Parameters used in UL-blockDAG protocol

Symbols	Description
G	Block DAG
G_U	Undirected version of G
C	Set of the blocks in block DAG
E	References to blocks in block DAG
C_1	Blocks created by honest nodes ($C_1 \subseteq C$)
C_2	Blocks created by attacker (malicious) nodes ($C_2 \subset C$)
$\mathbf{A_d}$	Adjacency matrix of block DAG
$\mathbf{A}$	Symmetrized Adjacency matrix for G_U
$\mathbf{D}$	Degree matrix
d_i	i^{th} diagonal entry in $\mathbf{D}$
$\mathbf{L}$	Laplacian matrix
$\mathbf{x}$	Cluster indicator vector for C_1 and C_2
λ_i	i^{th} eigen value of $\mathbf{L}$
λ	Block creation rate
D	End-to-end delay in the network
b	Block size in **MB**
c	Client/Node in the network
G_t^c	Locally bserved blockDAG at client c at time t
n	Number of nodes in the network
q	Fraction of the attacker's hashrate
n_p	NUmber of peers to each node
T_p	Propagation delay
R	Upload bandwidth
T_p	Prpogation delay beteen peers
k	Number of confirmations

Definition 1. (Adjacency matrix). The adjacency matrix $\mathbf{A_d}$ of a directed graph G is an $\mid C \mid \times \mid C \mid$ matrix such that

$$(\mathbf{A_d})_{ij} = \begin{cases} 1, & \text{if } (i,j)\,\epsilon\, E \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Definition 2. (Symmetrization). The symmetric adjacency matrix $\mathbf{A}$ from $\mathbf{A_d}$ without changing the number of edges [14] is obtained as follows

$$\mathbf{A} = \mathbf{A_d} + \mathbf{A_d^T} \tag{2}$$

$\mathbf{A}$ represents the adjacency matrix of undirected graph (G_U) for the original directed graph G.

Definition 3. (Degree Matrix). The degree matrix $\mathbf{D}$ of the graph (G_U) is obtained from $\mathbf{A}$ as follows

$$\mathbf{D} = \mathbf{diag}\{d_1, d_2, \ldots, d_{|C|}\} \tag{3}$$

Where

$$d_i = \sum_{j=1}^{|C|} \mathbf{A}_{ij} \tag{4}$$

Definition 4. (Laplacian Matrix). The graph Laplacian matrix $\mathbf{L}$ is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{5}$$

Th important properties of $\mathbf{L}$ are defined in [6].

Definition 5. Rayleigh Ratio. The main tool in the optimizaion problem for graph clustering is Rayleigh ratio [15] defined as

$$R(L) = \frac{\mathbf{x^T L x}}{\mathbf{x^T x}} \tag{6}$$

and

$$\lambda_{min} < R(L) < \lambda_{max} \tag{7}$$

Where

- $x \epsilon \mathbb{R}^{|C|}$ is an orthonormal eigen vector of $\mathbf{L}$ also used as cluster indicator vector in graph clustering problem.
- λ_{min} and λ_{max} are minimum and maximum eigen values of $\mathbf{L}$.

V. UL-BLOCKDAG CONSENSUS PROTOCOL

In this section, we discuss the operation of the consensus protocol. The protocol consists of the following two steps-

- Seperating the blocks (with less inter-connectivity) created by an attacker from the well-connected blocks in the blockDAG using the spectral graph theory based Unsupervised Learning algorithm.
- Topological ordering of the blocks based on the directed edges in DAG structure.

A. Spectral graph clustering for DAG

The graph clustering is defined in two ways by maximizing the intra-cluster edge connections and minimizing the inter-cluster edge connections. The optimiztion problem for graph clustring based on spectral graph theory [16] is

$$\min_{\mathbf{x}\in\mathbb{R}^{|C|}} \quad \frac{\mathbf{x^T L x}}{\mathbf{x^T x}} \tag{8}$$

$$\text{s.t} \quad \mathbf{x^T 1} = 0 \tag{9}$$

$$\mathbf{x^T x} = 1 \tag{10}$$

By the Rayleigh-Ritz theorem [15], the solution for the above optimization problem is the eigen vector ($\mathbf{x}$) corresponding to the second smallest eigen value (λ_2) of $\mathbf{L}$. Finally, labelling the positive and negative components of $\mathbf{x}$ to corresponding vertices gives two clusters C_1 and C_2 of graph G . This entire graph clustering procedure is shown in Algorithm 1 .

The intution for using Algorithm 1 for separating the attacker blocks is explained through the following example.

An attacker model of blockDAG structure is shown in Fig. 1. The attacker keeps the blocks having the conflicting transactions with the original transactions in secret untill sufficient number of confirmations to original transactions attained and broadcast them all together.

Fig. 2 shows the output of the Algorithm 1 for the blockDAG shown in Fig. 1. The elements of the eigen vector corresponding to the second smallest eigen value of the Laplacian matrix $\mathbf{L}$ for the blockDAG in Fig. 1 is mapped to discrete values in the set $\{1, -1\}$ which represents the clusters C_1 and C_2. The blocks from $\mathbf{0-13}$ are added to cluster C_1 and the attacker blocks $\mathbf{15-19}$ along with block $\mathbf{14}$ which reference to $\mathbf{14}$ (an attacker block) are added to cluster C_2.

B. The client protocol

A client is a node in the network which is either a miner having a high computional power to solve PoW mining problem or a simple node with no mining power. The client protocol is a two step consensus protocol described through the following two algorithms.

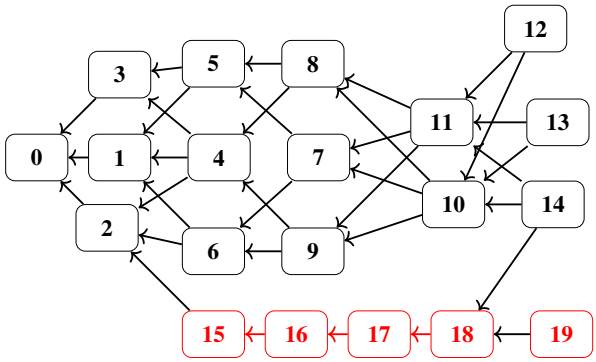


Fig. 1: An example of a blockDAG attacker model. Block **15** created by an attacker has a conflicting transaction with a transaction in block **3** shown by rounded text.The blocks from $\mathbf{0-14}$ are generated by honest miners and an attacker keeps the blocks from $\mathbf{15-17}$ in secret untill **3** attained sufficient confirmations (In this case 3 confirmations) and broadcast all the blocks created by the attacker. Observe that, there are no references to attacker blocks $\mathbf{15-17}$ from the blocks created by honest blocks which indicates blocks $\mathbf{15-17}$ are in secret, but both the honest block (**14**) and attacker block (**19**) has a reference to block **18** which indicates the attacker broadcasted the blocks after creation of block **18**.

1) Separating the attacker blocks and honest blocks: For all $c \in V$, the input G_t^c to Algorithm 2 at each time instant t are shown in Fig. 3. For simplicity, at each instant of time a hypothetical block H added to blockDAG.

Algorithm 2 operates as follows :

- Given a blockDAG G_t^c observed locally at each client c at a paricular time t , the algorithm recursively finds the clusters C_1 and C_2 each time after adding new blocks at each block height.

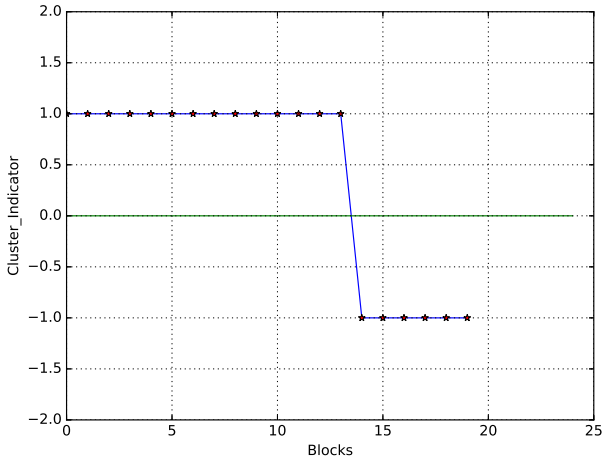


Fig. 2: Grouping the blocks in blockDAG shown Fig. 1 into two clusters. Blocks represented above the green line are added to C_1 and blocks below the green line are added to C_2.

Algorithm 1 Spectral graph clustering for DAG

Input: G - DAG
output: C_1, C_2

procedure FIND-CLUSTERS(G)
  Construct $\mathbf{A_d}$ for G
  Compute $\mathbf{A} = \mathbf{A_d} + \mathbf{A_d^T}$
  Compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$
  Compute $\mathbf{x}$ ▷ eigen vector corresponding to 2^{nd} smallest eigen value of $\mathbf{L}$
  if $\mathbf{x_i} >= 0$ **then**
    add to C_1
  else
    add to C_2
  end if
  return C_1, C_2
end procedure

- Client should include all the blocks at each block height as a part of the input DAG till those blocks meet the required number of conformations.
- The confirmed blocks are added to $blueList$ for arranging them into topological order as described in Algorithm 3.

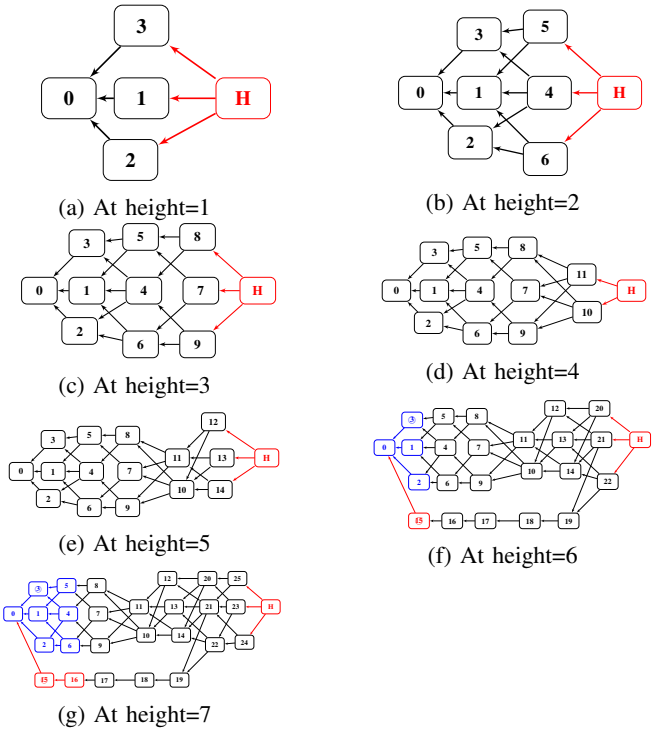
Algorithm 2 is illustrated with the following example shown in Figs. 3 and 4.

In this example, We assumed the number of confirmations ($k = 4$) is constant and We leave the analysis of required number of confirmations (k) to the future version. The attacker creates a conflicting transaction to a transaction in block $\mathbf{3}$ and included in block $\mathbf{15}$. The attacker keeps the blocks $\mathbf{15-18}$ secret till the blocks at $height = 1$ (blocks $\mathbf{1-3}$) confirmed by the other clients in the network with the required number of conformations and broadcast all the blocks from $\mathbf{15-19}$ at $height = 6$. Each client will confirm the blocks at a particular height h to either of the clusters C_1 or C_2 when it noticed the height increses to $h + k + 1$ in G_t^c. Here, at $height = 6$, the blocks $\mathbf{1-3}$ are confirmed to C_1 (shown in blue) and block $\mathbf{15}$ is added to C_2 (shown in red). Similarly at $height = 7$.

Fig. 4 shows the spectral properties (The eigen vector corresponding to 2^{nd} smallest eigen value) of the blockDAG structures shown in Fig. 3, which illustrates assigning the blocks to clusters C_1 (above the orange line) and C_2 (below the orange line). Fig. 4f and 4g shows the seperation of the attacker blocks ($\mathbf{15-19}$) from the blocks created by the honest nodes.

2) Ordering of confirmed blocks: Algorithm 3 operates as follows :

- After finding the $blueList$ from the given blockDAG G_t^c, each client should add the blocks in $blueList$ to $topo\_q$ starting from $genesis$ block.
- Arrange the $children$ of a block in ascending order of their hash indicates the block with lesser hash value (block with smaller hash than the target value in PoW puzzle [1]) added first among the blocks at same block height.



(a) At height=1 (b) At height=2 (c) At height=3 (d) At height=4 (e) At height=5 (f) At height=6 (g) At height=7

Fig. 3: An example to show the operation of Algorithm 2. As time progresses, clustering and confirmation of blocks to C_1 (blue) and C_2 (red)

.

Algorithm 2 Finding the list of confirmed blocks

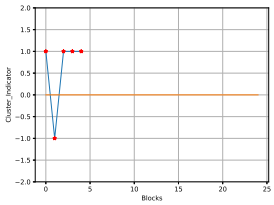
Input: G_t^c - A blockDAG at time t at client c, k - Number of confirmations required
output: *blueList* - A set of confirmed blocks

$i \leftarrow 0$, $x \leftarrow 0$
procedure FIND-LIST(G_t^c, k)
  $i \leftarrow i + 1$
  if $i > k$ **then**
    $x \leftarrow x + 1$
  end if
  $G_t^c \leftarrow G_t^c \backslash$ {All left most blocks till $height = x$}
  $C_1, C_2 \leftarrow FIND - CLUSTERS(G_t^c)$
  if $x > 0$ **then**
    for all $B \in C_1 \cap Blocks\ at\ height\ x$ **do**
      blueList.add(B)
    end for
  end if
  return $blueList$
end procedure

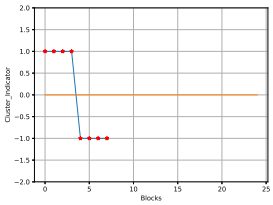
VI. Results and Discussion

Table II lists the values of the parameters used for generating the results in this section. See Table II for a description.

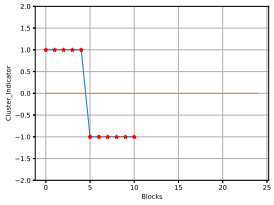
We have conducted an event-driven simulation using python by generating events for some duration ($\frac{1}{10}^{th}$ of a day $\approx$ 8640 sec) as per the blockDAG mining protocol in [4] and [5] with



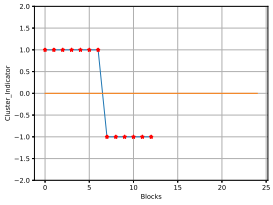
(a) At height = 1



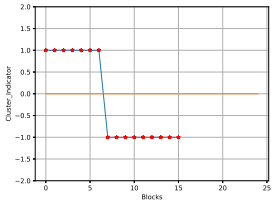
(b) At height = 2



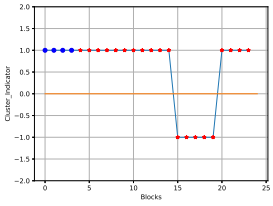
(c) At height = 3



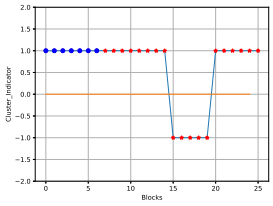
(d) At height = 4



(e) At height = 5



(f) At height = 6



(g) At height = 7

Fig. 4: Spectral properties for $2-clusters$ of blockDAGs shown in Fig. 3.

the proposed consensus protocol for a network with $n = 100$ nodes and 10 miners having the Hashrate distribution shown in [17] and also added a miner with a computational power of a fraction of $\approx \frac{1}{3}$ (and $\approx \frac{1}{2}$) of the total hashrate. The events are generated for creating a block, broadcasting the block to neighbours and adding the block after verifying the

TABLE II: Parameter values used for blockDAG

Parameter	value
n	100
n_p	8
T_p	30 msec
b	4 MB
R	10 Mbps
q	0.33 and 0.51
k	5
λ	1 blocks/sec

Algorithm 3 Ordering of blocks

Input: G_t^c - A blockDAG at time t at client c
output: *ordList* - an ordered list of blocks

procedure ORD(G_t^c)
  Intialize empty queue *topo_ q*
  topo_ q.push(*genesis*)
  $blueList \leftarrow FIND-LIST(G_t^c)$
  while $topo\_q \neq \phi$ **do**
    $B \leftarrow topo\_q.pop()$
    $ordList.add(B)$
    $children(B) \leftarrow \{j : (j, B) \in E\}$
    Sort $children(B)$ in asceding order of their $hash$ value
    for all $C \in children(B) \cap blueList$ **do**
      $topo\_q.push(C)$
    end for
  end while
  return $ordList$
end procedure

block height and references to the previous blocks. The blocks are created with a rate (λ) of 1 block/sec (or block interval is 1 sec).

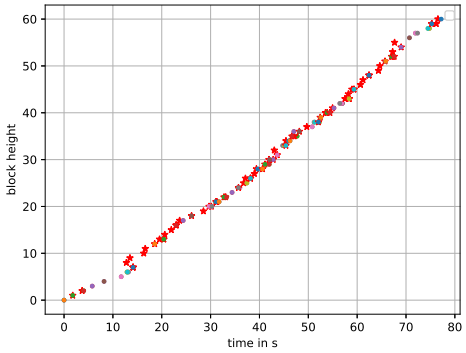
We have created the events such that the attacker with 33.33% (and 51%) of hashrate follows his double-spending strategy similar to an example shown in Fig. 3 from $height = 3$ to $height = 7$ by chosing $k = 5$. Fig. 5 show a part (for a duration of 86 sec) of total confirmed blocks ($blueList$) created by all miners in the network, where the attcker blocks (red in colour) from height $\mathbf{3-7}$ are not included in the $blueList$ and only honest blocks are present from height 3 to 7 as per Algorithm 2.

Fig. 6 shows the proportion of the rewards (shown by red dots and blue $+$) of each miner are nearly equal to their proportion of the hash rates (shown by a line) in the network, which indicates the fairness of the UL-blockDAG protocol.

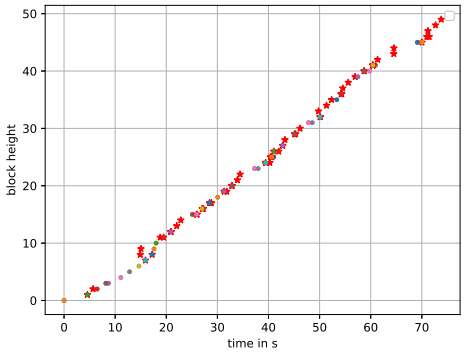
And also, the simulation results created ≈ 11000 blocks for a duration of 8640 sec ($\frac{1}{10}^{th}$ the day) with size 4 Mb each are included in the $ordList$ after executing the ordering Algorithm which proves the increase in TPS to thousands of transactions per second.

VII. Conclustions and Future Research

In this paper, we show the graph clustering based on the spectral graph theory concepts for DAG to seperate the blocks created by an attacker with double-spend strategy. The simulation results show that Algorithms 2 and 3 for separating the attacker blocks with different attacker hash rates (0.33 and 0.51) when attacker tries to attempt double-spend attack. We also demonstrate the fairness of the system and the increase in TPS to thousands of transactions per second as large number of blocks created due to DAG structure of the ledger and confirmed due to clustering and ordering algorithms. In future, we will analyse the number of required confirmations at client

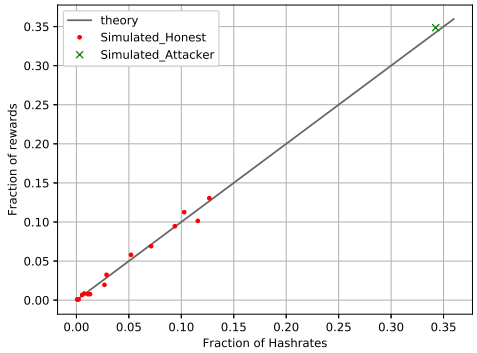


(a) $q = 0.33$

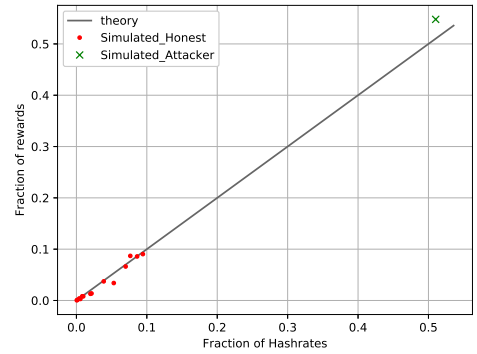


(b) q = 0.51

Fig. 5: Blocks created by all miners. Attacker blocks are shown in red clour and blocks created by other miners are shown with different colours. Blocks created by the attacker from height $\mathbf{3-7}$ are not included in $blueList$. Coincidence of the blocks created by different miners (with different coloured blocks) at each block height represents multible blocks at each height due to blockDAG structure of the ledger.



(a) $q = 0.33$



(b) q = 0.51

Fig. 6: Hash rate Vs Rewards

end in terms of the fraction of the attacker's hash rate and end-to-end propagation delay in the network. We also analyse the different clustering algorithms for identifying blocks created by an attacker with double-spend strategy.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[2] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*. Trento, Italy: IEEE, Sep. 2013, pp. 1–10.

[3] M. Rosenfeld, "Analysis of hashrate-based double spending," *CoRR*, vol. abs/1402.2009, pp. 1–13, February 2014. [Online]. Available: http://arxiv.org/abs/1402.2009

[4] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," Cryptology ePrint Archive, Report 2016/1159, 2016, https://eprint.iacr.org/2016/1159.

[5] Y. Sompolinsky and A. Zohar, "Phantom: A scalable blockdag protocol," Cryptology ePrint Archive, Report 2018/104, 2018, https://eprint.iacr.org/2018/104.

[6] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11222-007-9033-z

[7] ethereum/wiki, "A next-generation smart contract and decentralized application platform," 2015. [Online]. Available: https://github.com/ethereum/wiki/wiki/WhitePaper/

[8] Y. Sompolinsky and Z. Aviv, "Secure high-rate transaction processing in bitcoin," 01 2015, pp. 1–20.

[9] Etherscan. [Online]. Available: https://etherscan.io

[10] A. Kiayias and G. Panagiotakos, *On Trees, Chains and Fast Transactions in the Blockchain*, 07 2019, pp. 327–351.

[11] B. S. Reddy and G. V. V. Sharma, "Optimal transaction throughput in proof-of-work based blockchain networks," *Proceedings*, vol. 28, no. 1, 2019. [Online]. Available: https://www.mdpi.com/2504-3900/28/1/6

[12] W. V. Gehrlein, "Condorcet's paradox and the likelihood of its occurrence: different perspectives on balanced preferences*," *Theory and Decision*, vol. 52, no. 2, pp. 171–199, Mar 2002. [Online]. Available: https://doi.org/10.1023/A:1015551010381

[13] C. Li, P. Li, W. Xu, F. Long, and A. C. Yao, "Scaling nakamoto consensus to thousands of transactions per second," *CoRR*, vol. abs/1805.03870, 2018. [Online]. Available: http://arxiv.org/abs/1805.03870

[14] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, 2013.

[15] J. H. Gallier, "Notes on elementary spectral graph theory. applications to graph clustering using normalized cuts," *CoRR*, vol. abs/1311.2492, 2013. [Online]. Available: http://arxiv.org/abs/1311.2492

[16] D. Luo, C. Ding, F. Nie, and H. Huang, "Cauchy graph embedding," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ser. ICML '11, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, June 2011, pp. 553–560.

[17] Blockchain, "Hash Rate Distribution," 2020. [Online]. Available: https://www.blockchain.com/pools