GAME THEORY APPLICATION RESOURCES MANAGEMENT AND DISTRIBUTION IN BLOCKCHAIN NETWORK

Cong Hung Tran[1], Dien Tam Le[1, 2], Thanh Hieu Huynh[3]

[1]Posts and Telecommunications Institute of Technology
[2]Thu Duc Technology College
[3]Saigon University

ABSTRACT

The paper illustrated a basic blockchain system, applying game theory to simulate resource management in blockchain transactions. By the method of illustration, simulation, our team has demonstrated the effect of game theory transactions, transactions with specific value can demonstrate the benefits of game theory in co-life. time can be used to manage resources in blockchain. Based on the proposed algorithm model, we have built a test system with the maximum number of virtual machines to demonstrate the effectiveness in applying game theory in managing and distributing resources for transactions in the blockchain network.

1. INTRODUCTION

In recent times, game theory has been gradually exploited for research and application in Blockchain. It is basically the mathematical model that studies the strategic interactions between players to make reasonable decisions, if imposed on the Blockchain platform, game theory has analyzed the strategy of the consensus nodes. Through the interactions between them such as understanding and predicting mutually exploiting behaviors, then having optimal response strategy based on balance. Furthermore, game theory can be used to develop incentive mechanisms that prevent nodes from misconduct or initiate attacks. As such, referring to game theory is voluntary decision-making with all consensus of the nodes in the Blockchain network.

In a game, each player makes his or her own strategy to maximize its utility, based on the other player's strategy [1]. Referring to game theory, it is necessary to mention the following terms:

Game: Any situation where the outcome depends on the actions of two or more decision makers (players).

- Player: The player is the decision-maker in the game. Inside Blockchain, the player can be a miner, a mining pool or a blockchain user
- Convenience: Gadget, meaning payout, interest, or revenue that reflects a player's expected outcome

* Strategy: A complete plan of action that the player will use depending on the circumstances that arise in the game. In general, a player's utility is determined not only on the player's strategy, but also on the strategy of the other players. Due to the different strategies nodes implement to maximize their own strategy of utilities, nodes can attach new verification blocks to different blocks in the Blockchain

* Reasonableness: A reasonable player, means always watching, if the Player always maximizes his payout

Game theory pursues two basic assumptions. One is that an individual or group of individuals is a participant in the game and the goal is to gain benefits (be it to win or minimize his losses). Second, participating players always calculate in advance of their decisions according to the principle of reason to try to get more utility for themselves.

This article will present the problem of using game theory to simulate a resource optimization method for blockchain network to achieve the best efficiency. The content of the paper is divided into 5 sections: Part I – introduction; PartII - the principles of game theory; Part III - System model; Part IV - Simulation results; Part V - conclusions and recommendations.

2. The Principles of Game Theory

Applying the principles of game theory to the management and distribution of resources in the Blockchain network [2],[3], we will consider the type of static game with complete information. The problem is set out as follows:

In a certain village there is a resource of grassland, with n households, and all of them make a living by raising cows on this common pasture. The benefits of these households depend on the number of cattle grazed. In the case when the number of cattle for grazing is small, the benefits of households are not affected when one household raises 1-2 cows. However, once the number of cows for grazing is large and the grassland is limited, if there are too many cows, there will not be enough grass for the whole herd. Then the mean benefit from cattle grazing would be inversely related to the total number of cows.

The problem will be presented as follows:

- The number of households participating in cow raising is **n**
- Strategic space is the strategic set Si (or strategies that can be devised) condition is: **$0 \le gi \le Gmax$**
- Outcome (per household): **$vi = gi.v(g1 + g2 +\ldots+ gi\text{-}1 + gi + gi\text{+}1 +\ldots+gn) - c.gi = gi .v (gi +g\text{-}i) - c.gi$**

Nash equilibrium occurs when combining strategies:

gi (g*1, g*2 ,…., g*n) if with every household i have g*i The strategy chosen by the ith household is the ith best response to the strategies of the **(n-1)** remaining households **(g*-i)** Or can be interpreted: **$vi(g^*i, g^*\text{-}i) \ge vi(gi, g^*\text{-}i)$**

Nash equilibrium is created by the best response strategies of all players (corresponding to the optimal strategies of the remaining players), so it is strategically stable and strategically stable. At the same time it is self-enforcement, each player, once maximizing his interests (while others try to do so), will voluntarily comply with the equal to Nash, at the same time they had no incentive to move away from this equilibrium

Returning to the above problem from the initial data, it is possible to infer the optimal conditions for the ith household:

v(gi + g*-i) + giv'(gi + g*-i) – c = 0 (1)

Set **v(G) = v(gi + g-i)** is the additional revenue of household i when grazing one more cow

v'(gi + g-i): The ith household's revenue is reduced due to external influences because one final cow is grazed

v(gi + g-i) - gi.v'(gi + g-i) = marginal revenue of the ith household (marginal revenue is the additional revenue earned by producing and selling an additional unit of goods) c is the marginal cost of the ith household (marginal cost is the change in total cost with an increase of 1 unit of output)

Case 1: The result is determined by the individual's decision, the ith household decides the number of its herds but does not care about its interests affecting the cows of the households. Other

From (1) we add up the side by n and then divide the sides by n: **v(G*) + $\mathbf{1}\boldsymbol{n}$ G*v'(G*) – c = 0 (1')**

Case 2: The result will be decided by the specific group here will be the decision of the whole village. Then, the whole village will choose G to optimize v, with **v = G.v(G) – G.c**

The conditions for optimal results are: v**(G**) + G** v'(G**) – c = 0 (2)**

(2) happens when marginal revenue = marginal cost. Then, the results will be different in case 1 because it is decided by the collective when the interests of the collective will be neutralized, accounting for all n households in the village. Selected strategy **G**< G*** in case 1 (the number of cows grazed when the decision is made personal will be greater than the number of cows to graze when the decision is collective). In other words, the result will be more reasonable and optimize the village's resources for grazing. From **G**< G*** with the original assumption, we have **v(0) = 0, v'(G) >0** (in the case G is small), but if G exceeds the limit **v'(G)<0**. We can represent the function **v(G)**convex parasol graph as follows:

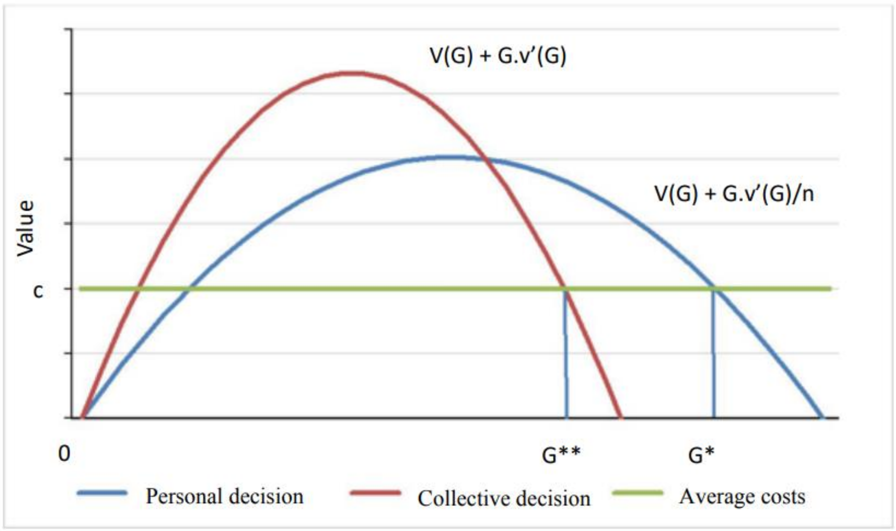


Figure 1: Graph showing the value of each strategy

From this graph, we can conclude that game theory principle when applied to resource management and distribution for an operating system, the results will be optimized when the decision is made to neutralize benefits. Utility, ensuring the system is not subject to resource conflicts and will improve performance and stability [4]. The calculation to optimize the interests of players participating in this problem is shown quite clearly if you choose to optimize the benefits for individual families, the benefits of the collective (other households will be reduced to the lowest level) and vice versa. From here will build simulations following this idea.

3. SYSTEM MODEL

From the actual problem above, we consider individuals and collectives to be 2 nodes A and B in the system, these nodes are verified by the village leader.

The idea of a simulation system built from Blockchain consists of nodes like computers linked together in peer-to-peer networks, all data is extracted from the ledger. A new node wants to join the system through the 0 bootnode button (the open door button of the system), to know the addresses of other nodes and notify other nodes of their participation. When receiving notification about the appearance of a new button, the system will update its data again.

First go through the basic steps to form a system:

Step 1: Initialize Blockchain chain

The initial string is empty, then create a profile with the button registration function, including: provide a new address => send a new node address to the whole system. Then the new node will be added to the list of nodes in the system and will update the data at the same time.

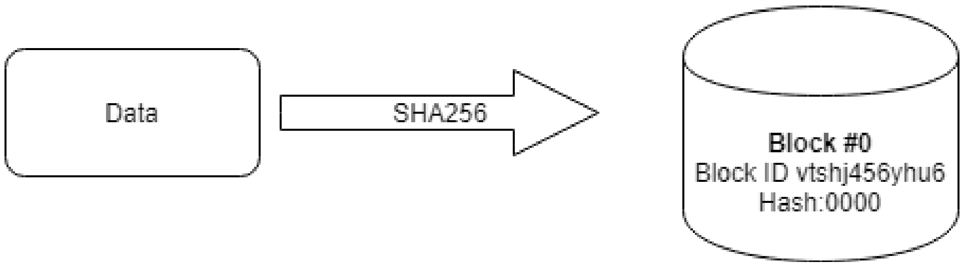


Figure 2: Data entered into the system

Step 2: add the transaction to the block - create new data

The add_new_transaction function adds a transaction to the Block's current list of transactions, in fact updates the information and needs of the two companies to the ledger, then assigns a timestamp for the transaction and saves the data (the data will be encrypted using the SHA256 algorithm that returns a hash value)

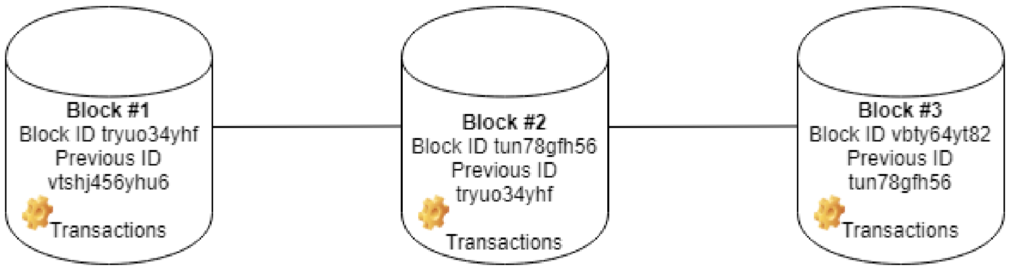


Figure 3: Links between blocks Step 3: Create a new block - authenticate the transaction - add a new block to the chain

The transactions will initially be stored as a group of transactions that have not been authenticated. The process of putting unauthenticated transactions into a block and calculating the POW is called Mining blocks. When the nonce satisfies the found constraints, we can say that a block has been mined and it can be put into the blockchain [5].

The problem is that the hash value of all subsequent blocks can be recalculated quite easily to create another valid blockchain. To prevent this, we can exploit the asymmetry of the hash function we discussed above to make the task of calculating hash values more difficult and random. What this means: Instead of accepting any hash value for the block, we add some constraints to it. Let's add a constraint that our hash will start with n leading zeros where n is a positive integer.

Here we are going to add some fake data that we can change. We will add a new field, the nonce field. A nonce is a number we can keep changing until we have a hash function that satisfies the constraint. The nonce satisfying the constraints serves as proof that some computation has been performed. The number of zeros specified in the constraint determines the difficulty of the PoW algorithm (the larger the number of zeros, the harder it is to find the nonce). Also due to the asymmetry, POW is difficult to calculate but very easy to verify once you find the nonce

Step 4: Announce the new list of Blockchain chain

We need to develop a way for any node to inform the network that it has mined a block so that people can update their blockchain and move on to mining other blocks. Other nodes can just verify the PoW and add the newly mined block to their respective chain (verification is easy knowing the nonce)

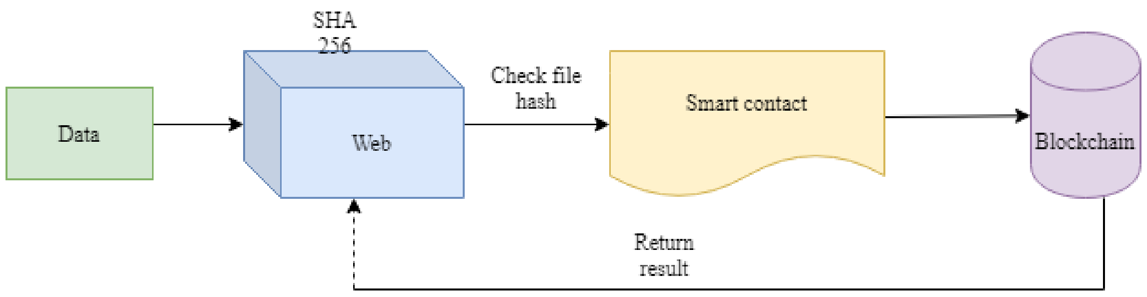


Figure 4: Basic operating mechanism of the system

Demonstration code:

```
@app.route('/add_block', methods=['POST'])

defverify_and_add_block():
block_data = request.get_json()

block = Block(block_data["index"],
block_data["transactions"],
block_data["timestamp"],
block_data["previous_hash"])

proof = block_data['hash']

added = blockchain.add_block(block, proof)

if not added:

return "The block was discarded by the node", 400
return "Block added to the chain", 201

defannounce_new_block(block):

for peer in peers: url = "{}add_block".format(peer)
requests.post(url, data=json.dumps(block.__dict__, sort_keys=True))
```

This step will take the chain information, calculate the current chain length, then update the list of nodes and return the current Blockchain chain information. At this point, the system will notify the results and update the information of the parties on the system.

To illustrate the nature of game theory [6], [9], this topic just stops at building a Blockchain system with 2 transaction nodes and connecting with the calculation of the opponent to get good values.

The system will be built on Ethereum, a Python programming language making the most of the resources from Flask. First, create Block classes with basic constructors to form blocks like:

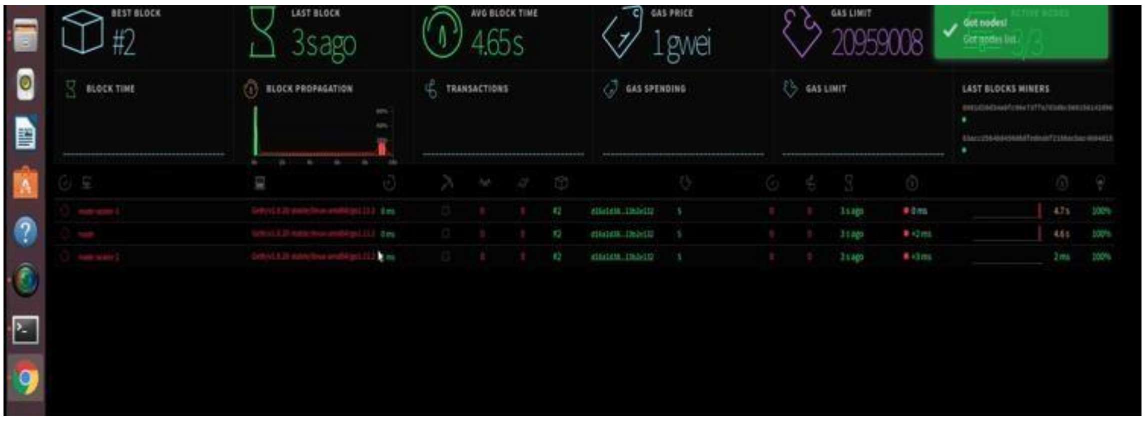
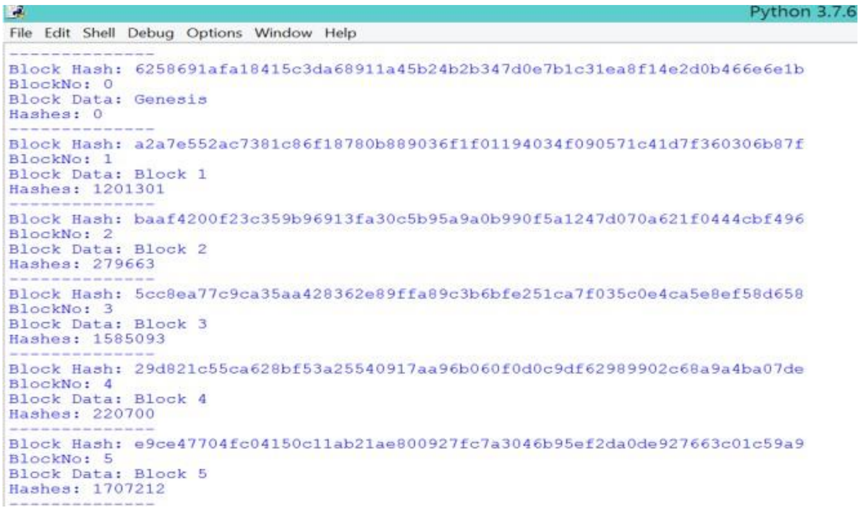


Figure 5: Creating a node in the system

At the beginning of the system, we will initialize the nodes, the first node will be the bootnode because the bootnode is the gateway to connect to the remaining nodes, this node is also the basis for verifying other nodes when wanting to join the network

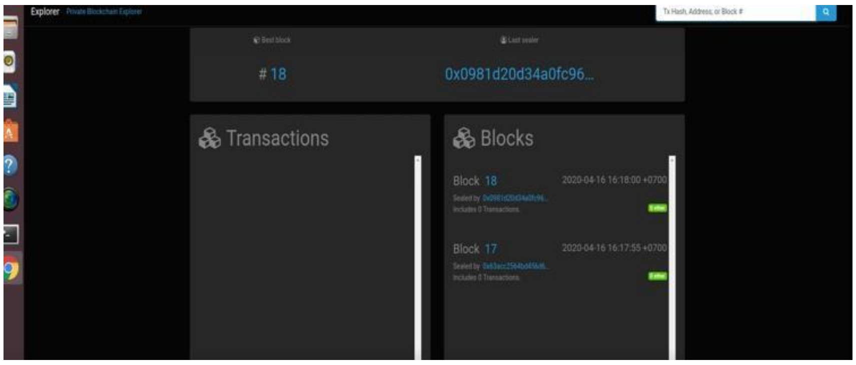
When the nodes are successfully created, we will connect the nodes together to generate transactions, this is an indispensable element in the system, the data (transactions) will be packed into blocks to be able to transaction (the first block is genesis with the value 0)



```
---------------
Block Hash: 6258691afa18415c3da68911a45b24b2b347d0e7b1c31ea8f14e2d0b466e6e1b
BlockNo: 0
Block Data: Genesis
Hashes: 0
---------------
Block Hash: a2a7e552ac7381c86f18780b889036f1f01194034f090571c41d7f360306b87f
BlockNo: 1
Block Data: Block 1
Hashes: 1201301
---------------
Block Hash: baaf4200f23c359b96913fa30c5b95a9a0b990f5a1247d070a621f0444cbf496
BlockNo: 2
Block Data: Block 2
Hashes: 279663
---------------
Block Hash: 5cc8ea77c9ca35aa428362e89ffa89c3b6bfe251ca7f035c0e4ca5e8ef58d658
BlockNo: 3
Block Data: Block 3
Hashes: 1585093
---------------
Block Hash: 29d821c55ca628bf53a25540917aa96b060f0d0c9df62989902c68a9a4ba07de
BlockNo: 4
Block Data: Block 4
Hashes: 220700
---------------
Block Hash: e9ce47704fc04150c11ab21ae800927fc7a3046b95ef2da0de927663c01c59a9
BlockNo: 5
Block Data: Block 5
Hashes: 1707212
---------------
```

Figure 6: Encrypted data

To demonstrate the nature of game theory we will consider two random nodes with simple transactions



18
0x0981d20d34a0fc96...

Block 18 2020-04-16 16:18:00 +0700
Includes 0 Transactions

Block 17 2020-04-16 16:17:55 +0700
Includes 0 Transactions

Figure 7: Transactions in Blockchain

The feature of Blockchain is to create new transactions and have new transactions to form the system, and this step will clarify the nature of game theory as follows: assuming Node A is malicious, want to promote a new block. to Node B, in this new block arising an invalid transaction (invalid signature), the calculation of the strategies of these two nodes will take place continuously in the system to eventually receive a value. in my best interests, namely the following two cases:

Case 1: Node B verifies to accept the broadcast of Node A's new block, and both will receive two different values corresponding to each strategy that the two nodes offer (Node A will benefit the most, Node B most damaged)

Case 2: Node B verifies and discovers that Node A is malicious, so refuses to accept the new block, now both of them also receive two values for themselves (Node A damages, Node B benefits)

This process can be illustrated with the following flowchart:

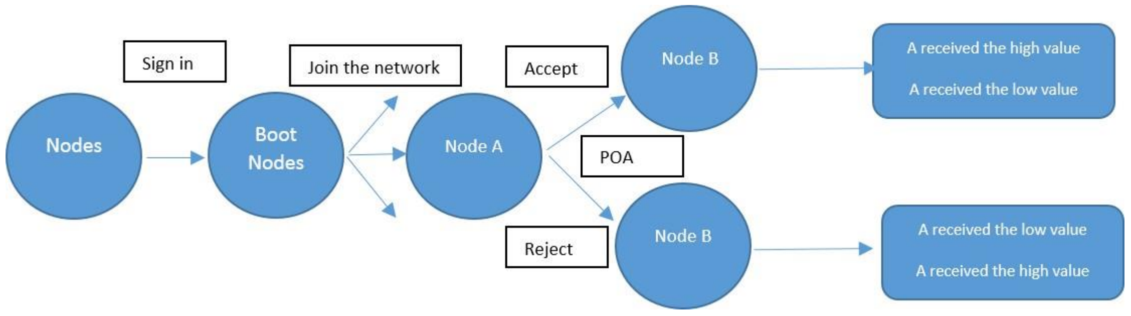


Figure 8: Diagram of verifying signatures between two nodes

To clarify the nature of game theory, we only consider Miner A to be malicious because if Miner A is honest, it is obvious to promote a new block.

In Blockchain to prevent tampering hidden inside blocks we use SHA-256 hash function

```
from hashlib import sha256

import json
defcompute_hash(block)

block_string = json.dumps(self.__dict__, sort_keys=True)
return sha256(block_string.encode()).hexdigest()
```

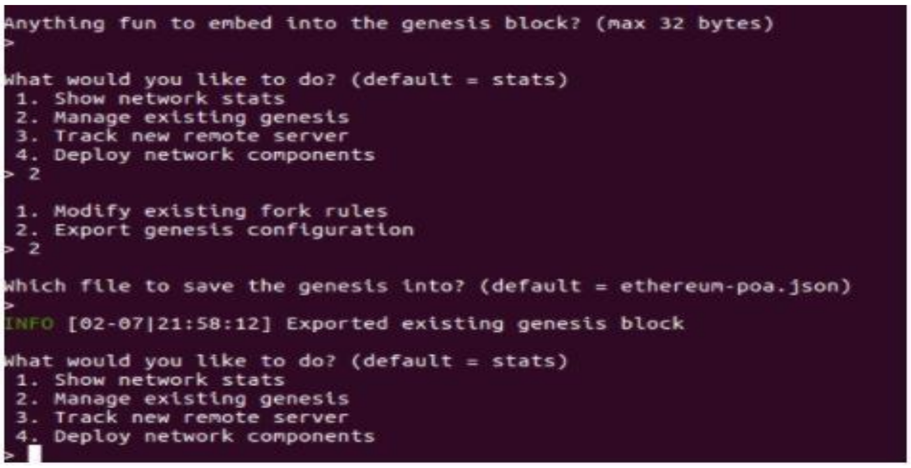
The descriptive code converts the block object into a JSON string and then returns the hash value that will be stored in the block as a digital signature. The blocks are linked together by a digital signature of the previous block generated by the hash function, so if any data changes, the signature changes.

In the case of Node A offering an unconfirmed transaction, PoA algorithm so that Node B can accept itself to get the most benefit, this process is called mining. At the same time, Node B also calculates whether to accept a transaction from Node A to receive different values (PoA is a smart verification algorithm, much faster than POW)

```
defadd_new_transaction(self, transaction):

self.unconfirmed_transactions.append(transaction)

def mine(self):

if not self.unconfirmed_transactions:

return False

last_block = self.last_block
```

This is an example for adding transactions pending to blockchain by adding them to block and issuing PoW. To illustrate more specifically, we will put this case of transactions in the world of cryptocurrencies, namely Ethereum, a fake Node A but approved by Node B, A will receive a greater value than B

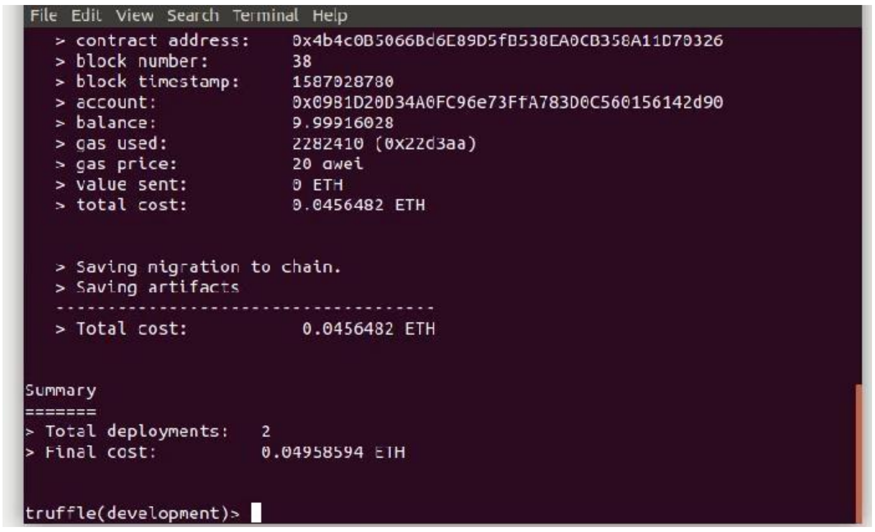
This word verification process will have a PoA consensus algorithm, this algorithm is developed on the PoS algorithm, but the difference is that instead of relying on the stake held in the system to delegate the verification authority. Instead PoA is based on the reputation of the user to designate that he or she has the authority to verify other users, specifically in this case bootnode.



```
Anything fun to embed into the genesis block? (max 32 bytes)
>

What would you like to do? (default = stats)
 1. Show network stats
 2. Manage existing genesis
 3. Track new remote server
 4. Deploy network components
> 2

 1. Modify existing fork rules
 2. Export genesis configuration
> 2

Which file to save the genesis into? (default = ethereum-poa.json)
>
INFO [02-07|21:58:12] Exported existing genesis block

What would you like to do? (default = stats)
 1. Show network stats
 2. Manage existing genesis
 3. Track new remote server
 4. Deploy network components
>
```

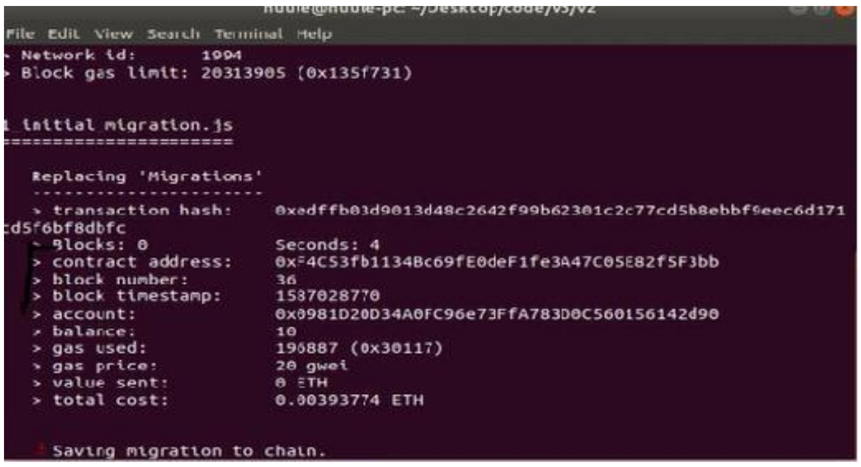
Figure 9: Blockchain network using PoA protocol

Case 1: B accepts A (false), both will return the corresponding value of A>B



```
   > contract address:    0x4b4c0B5066Bd6E89D5fB538EA0CB358A11D70326
   > block number:        38
   > block timestamp:     1587028780
   > account:             0x0981D20D34A0FC96e73FfA783D0C560156142d90
   > balance:             9.99916028
   > gas used:            2282410 (0x22d3aa)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0456482 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:           0.0456482 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.04958594 ETH


truffle(development)>
```

Figure 10: Value obtained for miner A



```
File  Edit  View  Search  Terminal  Help
- Network id:        1994
> Block gas limit: 20313905 (0x135f731)


1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0xedffb03d9013d48c2642f99b62301c2c77cd5b8ebbf9eec6d171
cd5f6bf8dbfc
   > Blocks: 0            Seconds: 4
   > contract address:    0xF4C53fb1134Bc69fE0deF1fe3A47C05E82f5F3bb
   > block number:        36
   > block timestamp:     1587028770
   > account:             0x0981D20D34A0FC96e73FfA783D0C560156142d90
   > balance:             10
   > gas used:            196887 (0x30117)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00393774 ETH


   Saving migration to chain.
```

Figure 11: The obtained value of miner B

Case 2: In contrast, when Miner A is malicious, when verifying, Miner B refused and did not accept the promotion of the new block to me, now B will receive a value greater than A will do the same

From the analysis of factors above, it shows that game theory always causes players to maximize their interests but always ensure a balance factor (A is the most profitable and vice versa), and at the same time when clearly identifying the nature of the game that can be applied in the Blockchain network, the resource management mechanism in the network will be very convenient and optimal when clearly defining each purpose and the needs of the objects in the system [7]. That will coordinate resources in the most reasonable way.

Usually, users have heterogeneous requests for different types of resources, needing to allocate resources to users according to their requirements. The main goal of equitable allocation of resources is to balance the percentage of the "shares" of each holder. In Blockchain infrastructure, thanks to virtualization technology, from the physical server, you can deploy the virtual machine on itself. All physical servers can accept virtual machine requests and create virtual machines in response to user requests. A request for a virtual machine includes (cpu, memory, disk) corresponding to the cpu, memory, and disk of the virtual machine CPU, RAM, DISK.

Process to allocate system resources:

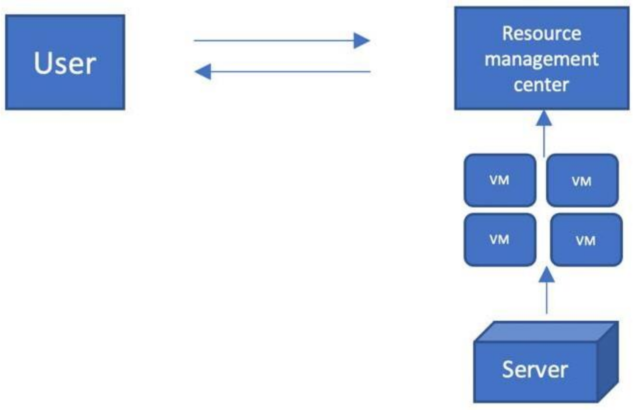


Figure 12: Diagram of resource allocation

The main purpose of this demo is to apply the principles of game theory in order to ensure that resources are allocated efficiently and stably, so a proper virtual machine resource allocation strategy is required. To concretize this we use the support algorithm, the metallurgical algorithm. In this paper, the metallurgical algorithm will have the following parameters:

- State-space: is the set of all strategies (in other words, sets of decisions) of the player.
- Energy function E( ): Here we use the benefit function of a decision as to the energy function for each state.
- Adjacent state: We find an adjacency of a state by randomly changing the player's decision.
- Temperature T: the initial temperature we set to 10000 (however can set arbitrarily, the larger T, the higherthe accuracy of the solution)

The Probability function is defined by the formula:

$$p = Exp\ (\frac{Current\ benefit\ value - New\ value\ of\ benefit}{Temperature})$$

The pseudo code of the algorithm:

```
Input: iterationsmax, tempmax

Output: Sbest// Best virtual machine allocation solution

Scurrent ← CreateInitialSolution();// Initialize the current solution
Sbest ← Scurrent; // Best solution initialization

for i = 1 to iterationsmax do

Si ← CreateNeighborSolution(Scurrent);// Generating a new possible solution - the adjacency state

tempcurr ← CalculateTemperature(i, tempmax); // Temperature calculation

if F(Si) ≤ F(Scurrent) then // Compare the benefit function of the solutions

Scurrent ← Si;
if F(Si) ≤ F(Sbest) then // Compare the benefit function of the solutions

Sbest ← Si;
end

else if Exp(F (Scurrent)− F (Si)/tempcurr ) > Rand() then

Scurrent ← Si;
end

end
return Sbest;
```

Algorithm Process Diagram:

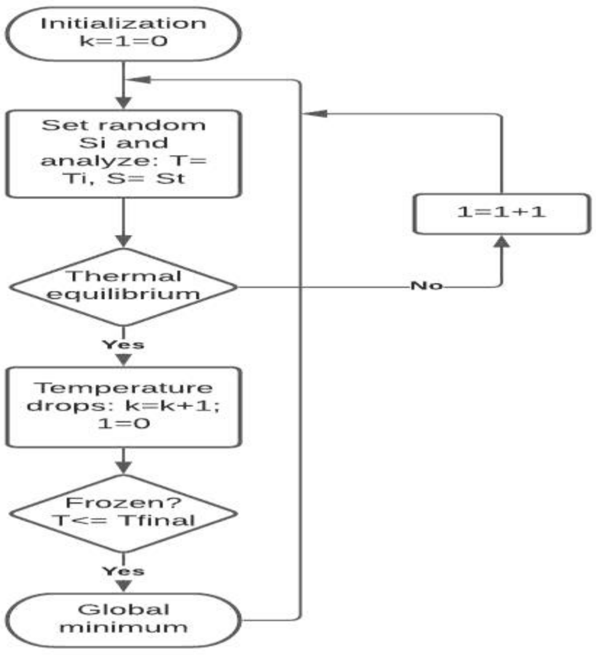


Figure 13: Metallurgical algorithm process

- K,1: Variable control loop
- 1: repeating at Ti temperature
- K: increases with thermal equilibrium at Ti temperature
- Ti& Si: random processing control

4. Simulation Results

The following emulator environment will run on NetBeans IDE software with the server configuration as follows:

- Operating system: win 10 Ultimate
- CPU: core I5
- Ram: 8G
- Hard disk: 500G

Required entry: Host host will be generated with a random configuration not exceeding maximum configuration

Required Output: Virtual machines require less than or equal to the number of available resources of the server Criteria to evaluate the resource allocation strategy of the system based on the Dominant share parameter (the maximum proportion of any given resource to the user) based on the DRF concept given by A Ghodsi [11]. Have studied in a multi-resource environment and fair levels of volatility (the smaller the level is guaranteed)

The operating principle of the metallurgical algorithm-based physical system is that resources will be allocated to virtual machines in each iteration step, virtual machines play the role of players, provided that resources are allocated. fairest and most effective.

Experimental results:

Table 1: Input parameters

	Number	MaxCPU	MinCPU	MaxRam	MinRam	MaxDisk	MinDisk
Host	100	10	4	32	8	32	16
VM	From 5 to 50	5	1	4	1	4	1

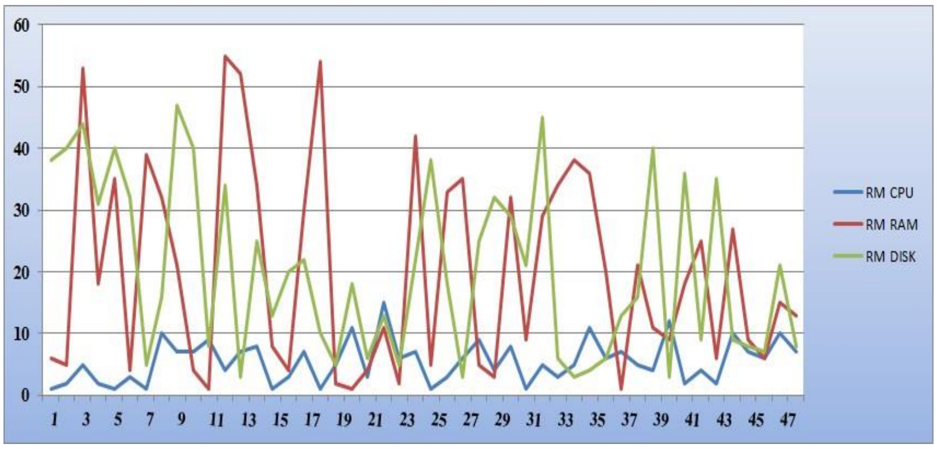


Figure 14: System initial resources

Table 2: Results when running from 5 to 50 virtual machines

The number of VMs	DOMINANT SHARE	FAIR MOVEMENT LEVEL	EXECUTION TIME (SECOND)	NUMBER OF Loops
5	0.111509	1.457	10	3047
10	0.053333	1.416	20	3046
20	0.033861	1.415	33	3093
30	0.025952	1.405	43	3099
40	0.020748	1.404	49	3105
50	0.017357	1.411	62	3119

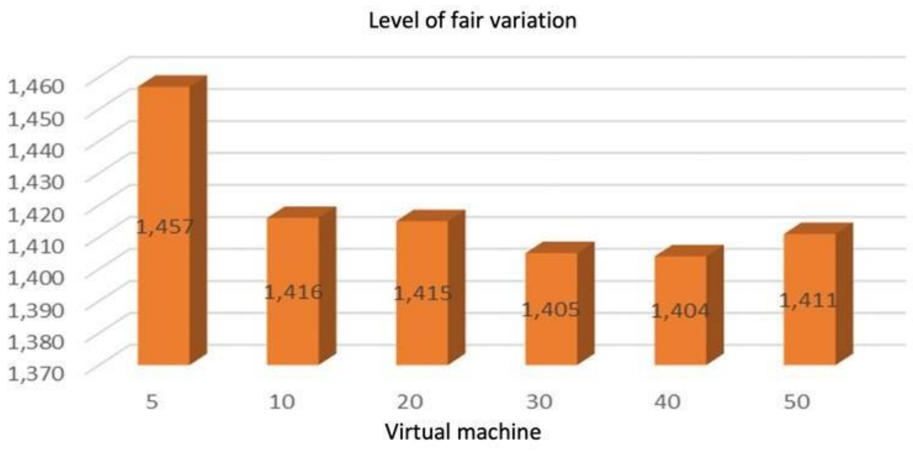


Figure 15: System equilibrium volatility

From the chart above it shows that when the number of virtual machines required is in the range of 10-50, the resource allocation strategy will be the best.

The above is just a local optimization simulation to realize the principle of game theory applied to the Blockchain network, thereby indicating the efficiency when using game theory in practice (if optimizing globally. will take enormous runtime). In the future, it is possible to research and build the most specific application with the development of the famous game model Stackelberg in parallel, creating web-based interfaces and web services to further improve performance. the system at the same time proves the superiority of the aforementioned game theory.

5. CONCLUSION

In this paper, we construct a basic blockchain resource distribution system simulation based on the properties of game theory. Thereby transactions witha specific value can demonstrate the benefits of game theory in life and can be applied to manage resources in Blockchain. The system is still very basic, the idea is only simulating, not an actual application, the management and distribution of resources towards balance, not really optimal in allocation. The number of virtual machines running is not much, it is difficult to accurately assess the results. In the coming time, our team will continue to research and develop the system to be able to apply the game theory model in resource optimization

REFERENCES

[1] José Moura, David Hutchison, "Game Theory for Multi-Access Edge Computing: Survey, Use Cases, and Future Trends", [online], January 2018

[2] Somdip, Dey, "A Proof of Work: Securing Majority-Attack in Blockchain Using Machine Learning and Algorithmic Game Theory", International Journal of Wireless and Microwave Technologies (IJWMT). pp. 1-9. ISSN 2076-1449, 2018

[3] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling Localized Peer-to-Peer Electricity Trading Among Plug-in Hybrid Electric Vehicles Using Consortium Blockchains", IEEE Transactions on Industrial Informatics, vol. PP, no. 99, pp.1-1,Preprints.

[4] Khiet Bui Thanh, Mai Hoang Xuan Lam, Nguyen Khac Chien, Ho Dac Hung, Pham Tran Vu, Cong Hung Tran, "An auto-scaling VM game approach for multi-tier application with Particle swarm optimization algorithm in Cloud computing" 2018 International Conference on Advanced Technologies for Communications (ATC) , ATC 2018, 18- 20 October 2018 in Ho Chi Minh City, Vietnam, IEEE catalog number: CFP18ATC-USB, ISBN: 978-1-5386-6541- 1, ISSN: 2162-1039, 978-1-5386-6542-8/18/$31.00 ©2018 IEEE, pp. 326-331

[5] Xiaojun Liu, Wenbo Wang, DusitNiyato, Narisa Zhao, Ping Wang, "Evolutionary Game for Mining Pool Selection in Blockchain Networks", IEEE Wireless Communications Letters, 2018

[6] Wenbo Wang, DusitNiyato, Ping Wang, Amir Leshem, " Decentralized Caching for Content Delivery Based on Blockchain: A Game Theoretic Perspective", IEEE International Conference on Communications (ICC), 2018

[7] ZehuiXiong, Shaohan Feng, DusitNiyato, Ping Wang, Zhu Han, "Optimal Pricing-Based Edge Computing Resource Management in Mobile Blockchain", IEEE International Conference on Communications (ICC), 2018

[8] K. Suankaewmanee, D. T. Hoang, D. Niyato, S. Sawadsitang, P. Wang and Z. Han, "Performance analysis and application of mobile blockchain," in International Conference on Computing, Networking and Communications (ICNC), Maui, Hawaii, USA, March 2018.

[9] Yue Wang, Changbing Tang, Feilong Lin, Zhonglong Zheng, Zhongyu Chen, " Pool Strategies Selection in PoWBasedBlockchain Networks: Game-Theoretic Analysis" , IEEE Access (Volume: 7), 2019

[10] Ziyao Liu, Nguyen Cong Luong, Wenbo Wang, DusitNiyato, Ping Wang, Ying-Chang Liang, Dong In Kim, "A Survey on Applications of Game Theory in Blockchain", 2019

[11] Ali Ghodsi, MateiZaharia,BenjaminHindman, Andy Konwinski, Scott Shenker, Ion Stoica, "Dominant Resource Fairness (DRF), Fair Allocation of Multiple Resource Types", University of California, Berkeley