Proof of Game (PoG): A Game Theory Based Consensus Model

Adarsh Kumar and Saurabh Jain(✉)

School of Computer Science, University of Petroleum and Energy Studies, Bidholi, Dehradun, India
{adarsh.kumar, saurabh.jain}@ddn.upes.ac.in

Abstract. Now-a-days, Blockchain networks are widely accepted in various applications for its enhanced security levels. Blockchain characteristics like: peer-to-peer, decentralized and immutable distributed ledger makes this technology acceptable to academia, research and industry communities. This work proposes 'proof-of-game (PoG)' consensus algorithm suitable for resourceful and resource-constrained devices. Heavy computational challenge in block structure protects the blockchain network from selfish miners and majority attacks. PoG consensus algorithm is suitable for both single and multi-player challenges. It is observed that single and multi-bit challenges increases the resource consumption and makes it difficult for resource constrained device to confirm block in stipulated time period. However, a multi-round multi-bits challenge makes it feasible for resource constrained devices to provide high security within specified time period. In implementation, it is observed that mined blocks indicates the chances of attacks. Large number of blocks are mined if block miner is honest, computational challenge is high and number of participants associated with block is large. Similar scenario is possible with transactions. In results, it is observed that presence of large selfish miners decreases the blocks mined exponentially with increase in computational challenge.

Keywords: Blockchain · Game theory · Proof of concepts · Selfish miner · Miner algorithm · Cryptocurrency

1 Introduction

In 2008, Satoshi Nakamoto introduces the Blockchain concepts an immutable timestamp ledger of blocks [1]. These blocks contains transactions records in a distributed manner. These transaction recors includes payment history, personal data, challenges, information about consensus mechanisms etc. In recent years, blockchain has attracted tremendous importance in various industry domains. This importance can be realized from the number of cryptocurrencies adopted day-by-day. Presently, there are more than 2200 cryptocurrencies [1, 2]. In Blockchain, a distributed digital ledger of transactions is shared with all participants which in-turn eliminates the chances of centralized authority. New blocks can be added to existing Blockchain if user solves a computationally hard and easily verifiable challenge. This challenge is backbone of consensus algorithm used in Blockchain network, which restricts the users from malicious activities. This restriction is resource dependent and existing Blockchain

P. Karrupusamy et al. (Eds.): ICSCN 2019, LNDECT 39, pp. 755–764, 2020.
https://doi.org/10.1007/978-3-030-34515-0_80

implementation uses various consensus algorithms: Proof of Stake Model (PSM), Proof of Elapsed Time (PET), Byzantine Fault Tolerance Algorithm (BFTA), Practical Byzantine Fault Tolerance Algorithm (PBFTA), SIEVE Consensus Protocol (SCP), Cross Fault Tolerance (XFT), Federated Byzantine Agreement (FBA), Ripple Consensus Protocol (RCP), Stellar Consensus Protocol (SCP) etc. Resource-intensive consensus algorithm makes it impossible for a resource constrained block generator in selecting nodes randomly and/or limiting a number of blocks it can generate. Presence of multiple block generators can create duplicate blocks with a pool of transactions. To protect the Blockchain network from a large number of malicious block generator, block and fake transactions, trustworthy consensus algorithm is required such that once block can be generated for a Blockchain network within specified-period.

In this work, "Proof-of-Game (PoG)" for resource dependent and computational independent consensus algorithm is introduced. PoG restricts the block generator in generating the number of blocks based on the level of game it can play and score it can earn. PoG concept can be applied for single or a group of block generators. Single block generator uses one-player and group block generator uses multi-player PoG concept. Single player PoG put challenge individually for users interested to add new blocks in existing Blockchain. Whereas, multi-player PoG allows multiple block generators to combine and play (put challenge) with user. This scheme is useful for both resourceful and resource constrained environments because game difficulty levels, number of game levels and scoring levels are proportionate to resource availability. High difficulty level and/or earning high game or scoring levels indicates trustworthiness which in-turns increases the importance of Blockchain network.

The rest of this work is organized as follows: State-of-the-art consensus algorithms are explained in Sect. 2. Section 3 proposes the single and multi-player PoG concept for Blockchain network. Experimental analysis of proposed consensus algorithm in Blockchain network is performed in Sect. 4. Finally, Sect. 5 concludes the work and outlines the future work.

2 Literature Survey

As discussed earlier, extensive work has been done [3, 4] to design a consensus algorithm for Blockchain network. Bach *et al.* [5] performed a comparative analysis of blockchain consensus algorithms. This comparative analysis is performed on parameters like: scalability, method of rewarding validators, security risks etc. In observations, it is found that Proof of Work (PoW) is well accepted consensus algorithm now a days. In another observation, it is found that there are blockchain networks, which uses other blockchain networks like: Proof of Stake (PoS), Proof of Luck (PoL), Proof of eXercise (PoX) etc. According to [3], a blockchain network should include incentive-compatible sub-stages with tolerance to Byzantine and unfaithful faults, attackers with cumulated computational power should be identifiable and isolated a balance between processing throughput and network scalability etc. Game theory is also widely acceptable in consensus mechanisms. Various game theory based consensus algorithms are explored as follows:

Jiang and Wu [6] uses game-theoretic model to study impact of block size in miner's payoff. Here, miner's strategy in varying block sizes for earning profits is

analyzed. Miner's varies block sizes to optimize payoff. An experimental analysis to identify equilibrium where miners have no incentive to misbehave provide the guidelines on the default block size. Results show that a block size of 4 MB should be the default block size.

Swanson [7] argues that a permissionless set of nodes in blockchain network must require some game-theoretic approach for security. Blockchain network protocols versus trust tradeoff is a major challenge and various attempts are made to advance different sections of both sides. Cryptoeconomic game theory such as block reward halving i.e. rewards of miners are split into half after every four years is not fruitful for participants because they want to depend on a network for more than a decade. This would either be possible with significantly higher transaction fees or lower security thresholds from miner (or both). Another possible solution is to design an efficient and advanced game-theory model with less fee, high security and long participant dependency.

Dey [8] proposes a methodology with software agents to monitor blockchain network's stakeholders. This monitoring is allowed to detect anomalies using supervised machine learning algorithm and algorithm game theory. Supervised machine learning for anomaly detection is helpful in identifying various attacks like: majority population size attack, DoS, DDoS etc.

Liu *et al.* [9] performed a comprehensive survey over applications of game theory in blockchain. According to [9], various options available for game theory models are: non-cooperative game, extensive-form game, stackelberg game and stochastic game. Game theory can be applied in detecting various attacks like: selfish mining attack, majority attack, Denial of Service attack, false data sharing, distrustful goods trading and cyber-insurance. According to Nakamoto protocol [4], anyone in blockchain network can use token supply and transaction tipping while playing the role of blockchain mining. This process inculcates the profit process and maximize the payoff with effective application of game theory. Additionally, game theory can be integrated with two ways of mining management: individual mining and pool mining. In individual mining process, dominant strategy of each miner is either computational power or fork a longest chain. In addition to this, block size setting, pool selection and reward allocation are other action in pool mining based game theory model for payoff.

Dimitri [10] proposes a mining game theory based on block size. A large block size indirectly increases the power computation cost for a miner in generating a hash value. Further, author identifies that bitcoin mining process is profitable if fewer independent entities are actively mining, and the rationality in miners is profitable for all as large payoffs provide support to earn additional computational power.

Stone [11] proposes a game theoretical mode with payoff strategy related to block size. Large block size means large payoff price. Here, a threshold block size is analyzed for real scenario. Larger block size adds additional delay, which in-turns put the block into isolation. Isolation is a major cause for Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack. It is also bserved that physical structure of blockchain network does not allow a large gas price for large block size.

Various other game theory based models are adopted for blockchain mining [12–14]. Among those evolutionary game models, hierarchical game and auctions are widely explored. However, single game model for both resourceful and resource constrained device's network is not explored much. Major challenge in designing this model is resource dependent and challenge variated rewarding point based game theory approach.

3 Proof of Game

In this work, PoG incorporating game theoretical model and mining process is proposed for blockchain network. Algorithm 1 explains the regular blockchain construction process and specific block entities. Various functions are written to generate genesis block and blockchain's subsequent blocks. Each of these blocks are linked with predecessor and successor blocks through a calculated hash value. As desired, this model of blockchain network construction has the provision of adding blocks of any size at any location.

In order to protect a blockchain network from attacks, computational challenges are put forward to node. If a node is able to solve the challenge within stipulated time period then node is allowed to add new block in existing blockchain network. A hash based computational challenge is implemented using algorithm 2. Computational challenges in Proof of Work (PoW) can be implemented using various ways [6, 7]. Today, more than seven hundred cryptocurrencies are using PoW concept in building consensus. Algorithm 2 is an extention of Algorithm 1 integrating SHA256 based

Algorithm 1 Regular Blockchain

Goal: To create a blockchain without proof of game

```
Index <- BlockCounter()
BBlock <- null
UData <- UserData()
PHash <- PreviousHash(BBlock)
BTime <- CurrentTimestamp()

BlockCounter <- function() {
 index <- index+1
}

UserData <- function() {
  # User entered data
}

GBlock <- function(block) {
  BBlock <- block
   BTime <- TTime()
   pHash <- null
  Hash <- SHA256(BBlock)
}

PreviousHash <- function(block) {
if {block=="Genesis" || Index=="0"}{
  return "0"
else if (block == "null") {
  return "null"
else{
  return Hash(PreviousBlock)
   }
 }
}

Hash <- function(block) {
  return SHA256(block)
}

Blockchain <- function(block) {
Index <- BlockCounter()
UData <- UserData()
PHash <- PreviousHash(BBlock)
BTime <- CurrentTimestamp()
}

AddBlock <- function(block) {
Index <- block.index
UData <- block.data
PHash <- block.PreviousHash(BBlock)
BTime <- CurrentTimestamp()
return Blockchain((Index, UData, PHash, BTime))
}

GBlock(0,CurrentTimestamp(),"Genesis","0",SHA256(0,CurrentTimestamp(),"Genesis","0")) #
    =>Creates Genesis Block
FirstBlock = new Blockchain(Index,UData,PHash,BTime,SHA256(Index,UData,PHash,BTim))
Blockchain(FirstBlock) # =>Creates First Block
SecondBlock = new Blockchain(Index,UData,PHash,BTime,SHA256(Index,UData,PHash,BTim))
Blockchain(SecondBlock) # =>Creates Second Block
ThirdBlock = AddBlock(Index,UData,PHash,BTime),SHA256(Index,UData,PHash,BTime))
```

computational challenges for blockchain network. In this implementation, hash computational challenger is given to node and it has to compute the hash within stipulated time period. If it is able to generate the hash then node is allowed to insert new block in blockchain network else it can be put under scrutiny. Line 35–39 shows the process of add new block if challenge is verified.

Algorithm 2 Extending a regular blockchain with 'Proof of Work'

Goal: To create a blockchain with 'Proof of Work' and without 'Proof of Game'

```
Index <- BlockCounter()
BBlock <- null
UData <- UserData()
Challenge <- GenerateChallenge()
PHash <- PreviousHash(BBlock)
BTime <- CurrentTimestamp()

GenerateChallenge <- function(Value, Operation) {
Value <- GenerateValue()
Operation <- GenerateOperation()
ChallengeTime <- CurrentTimestamp()
}

ValidateChallenge <- function(TimeThreshold) {
CurrentTime <- CurrentTimestamp()

if {parent != SHA256(previousBlock) ||  !ValidTransaction(transaction) || (CurrentTime -
    ChallengeTime >= TimeThreshold)}{
  return "False"
else{
  return "True"
   }
 }
}

GBlock(0,CurrentTimestamp(),"Genesis","0", SHA256(0,CurrentTimestamp(),"Genesis","0",)) #
     =>Creates Genesis Block
NewBlock = new Blockchain(Index,UData,PHash,BTime,SHA256(Index,UData,PHash,BTim))
GenerateChallenge(Value, Operation) # =>Creates a Challenge
ValidateChallenge(TimeThreshold) # =>Validate a Block
Blockchain(NewBlock) # =>Creates New Block
```

Algorithm 3 extendeds Algorithms 1 and 2 with Proof-of-Authority (PoA), Proof-of-Ownership (PoO) and Proof-of-Stake (PoS). In PoA, one or group of selected nodes in the blockchain network is given the authority to put challenge rather than anyone put a challenge. Thus, trusted nodes are given priority using this concept. In order to implement PoA for single node, highly trusted node is selected. Whereas, a common signature of group of nodes is taken as consent of authority for group based PoA. PoA is additional overhead over PoW. However, it is useful in avoiding various attacks and providing higher security. PoO is an extension of PoA. In PoA, one or set of active nodes can be selected for challenge generation whereas, PoO is node specific. One or group of Owners need to be active for challenge generation and addition of new blocks in blockchain network. Apart from challenge generation, node with ownership right can verify any node's credentials before allowing it to add new block. Now, if a group of nodes are interested in creating a blockchain network based on node's investment then PoS is the best option. In PoS, those nodes are allowed to participate in the network which are having stake greater than a specific amount, within a specific range or

Algorithm 3 Proof of Game (PoG)

Goal: To identify and validate blockchain authority using 'Proof of Game'

```
PoW <- function(player) {
  if (player == 'single') {
    Lightweight_Hash_Computation(Time_
    Period, Hash_Algorithm, Data)
  }
  else if (player == 'multi') {
    While(Participant)) {
      group_sign=SIGNATURE(Identity,
    stakeholderList)
      Lightweight_Hash_Computation(Time_
    Period, Hash_Algorithm, Data, Group_
    Sign)
    }
  }
}

PoA <- function(player) {
  if (player == 'single') {
    Compute_Authority_Decision(Player_
    Signature, Player_Credentials)
  }
  else if (player == 'multi') {
    While(Participant)) {
      group_sign=SIGNATURE(Identity,
    stakeholderList)
      Compute_Authority_Decision(group_
    sign, Player_Credentials)
    }
  }
}

PoO <- function(player) {
  if (player == 'single') {
    Compute_Authority_Decision(Player_
    Signature, Owner_Creteria)
  }
  else if (player == 'multi') {
    While(Participant)) {
      group_sign=SIGNATURE(Identity,
    stakeholderList)
      Compute_Owner_Decision(group_sign,
    Owner_Creteria)
    }
  }
}
```

```
PoS <- function(player) {
  if (player == 'single') {
    Verify_Stake(Player_Stake)
  }
  else if (player == 'multi') {
    While(Participant)) {
      group_sign=SIGNATURE(Identity,
    stakeholderList)
      Compute_Owner_Decision(group_sign,
    Player_Stake)
    }
  }
}

PoG <- function(player) {
  if (player == 'single') {
    Execute PoW() || PoA() || PoO() || PoS
    ()
    Lightweight_Game_Single(bit_value,
    position_value)
  }
  else if (player == 'multi') {
    While(Participant)) {
      group_sign=SIGNATURE(Identity,
    stakeholderList)
      Lightweight_Game_multi(bit_value,
    position_value,participant_list)
    }
  }
}

Lightweight_Game_Single <- function(player
    ) {
bit <- Evaluate_Challenge(position)
if(verify(bit,position)=='valid') {
  return(bit,position)
  }
else {
  return 'invalid'
  }
}

Lightweight_Game_multi(bit_value, position
    _value,participant_list){
bit <- Evaluate_Challenge(position)
While(Participant <- participant_list)) {
bit_p <- Participant.bit
if(verify(bit,bit_p, position)=='valid') {
  return(bit,position)
  }
else {
  return 'invalid'
  }
}
```

interested to invest a minimum amount in business. Algorithm 3 explains the integration of PoW, PoA, PoO and PoS in blockchain network. This implementation is game theory based rewarding system. Initially, all participants are considered to be honest and share their resources. Computational resources are identified and verified by challenger before allowing any node to add new block. If a node is having sufficiently high computational resources then heavyweight computational challenge is put which in-turn ensures high security. Whereas, if node is resource constrained then lightweight computational challenges are used. Although lightweight computational challenges provide comparatively lighter security as compare to heavyweight computational challenges but security level can be enhanced using multi-rounds verification process explained in Algorithm 3. In multi-round verification process, bit value and bit position are verified at both challenge generation and challenge solver ends. If a threshold number of bit value and positions are verified then node is allowed to add new blocks. Challenger's trust over node increases with increase in number of new blocks. This trust is computed using rewarding points.

4 Analysis

This section explores the possibilities of various attacks in PoG algorithm for blockchain network.

- **Selfish Mining and Majority-Attack Detection:** Selfish mining and majority attacks in blockchain network is explored by various authors [12]. Improper incentive-compatibility and selfish miners may compromise the game theory process and gain higher profits than their due shares. In nutshell, selfish miners act as attacks and produce a tampered chain. This can be detected as follows:

Let there are two sides of miners: honest miner (H_M) and selfish miner (S_M). Each of these sides have their own population sizes. Let M represents the population size of H_M playing PoG, and m represents the population size of S_M interested in breaking PoG and earn higher profit. A miner competition is played between both sides to earn maximum profit, and perform or avoid majority attacks.

$$Chance_M^{Success} = Maximize\left\{ Height\ of\ honest\ miner - \frac{Sum\ of\ transactions\ secuted\ successfully\ at\ respected\ height}{total\ transactions\ of\ honest\ miner} \right\} \quad \text{(i)}$$

$$Chance_m^{Success} = Maximize\left\{ Height\ of\ selfish\ miner - \frac{Sum\ of\ transactions\ secuted\ successfully\ at\ respected\ height}{total\ transactions\ of\ selfish\ miner} \right\} \quad \text{(ii)}$$

Now, success probability of PoG is calculated as:

$$P_{Success}^{h} = \sum_{i=1}^{h} Chance_{M}^{Success} - Chance_{m}^{Success} \quad \text{(iii)}$$

From Eqs. (i), (ii) and (iii), it can be easily predicted that if $Chance_{M}^{Success} < Chance_{m}^{Success}$ then new block is considered to be created by selfish miner and can be rejected. After this check, transaction and block sizes are also compared before accepting or rejecting the new block. Eqs. (iv), (v) and (vi) explains the block size comparison.

$$Block_{size}^{selfish\_miner} = \frac{Sum\ of\ the\ block\ sizes\ created\ by\ selfish\ miner}{total\ number\ of\ blocks\ created\ by\ selfish\ miner} \quad \text{(iv)}$$

$$Block_{size}^{selfish\_miner} = \frac{Sum\ of\ the\ block\ sizes\ created\ by\ honest\ miner}{total\ number\ of\ blocks\ created\ by\ honest\ miner} \quad \text{(v)}$$

If $Block_{size}^{honest\_miner} \leq Block_{size}^{selfish\_miner} \leq Block_{size}^{honest\_miner}$ then
Now, Eq. (vi) evaluates the block future state in ideal condition

$$Block^{future\_state} = Height\ of\ new\ block\ in\ Merkle\ Tree - \frac{Sum\ of\ Expected\ height\ of\ blocks\ in\ Merkle\ Root\ Tree}{total\ number\ of\ transaction\ expected\ to\ be\ executed\ by\ honest\ block\ miners} \quad \text{(vi)}$$

If $Block_{size}^{honest\_miner} \mp \delta < Block_{size}^{selfish\_miner} < Block_{size}^{honest\_miner} \mp \delta$ then block miner with new block is considered to be honest else miner is considered as selfish and new block is put in majority block attack category. Here, δ is acceptable minor variation in block size. δ varies with challenge. A larger challenge results in larger block size and a smaller challenge results in smaller block size. In this work, 1-bit challenge to Hash plus exponentiation challenge is analyzed for proposed work. In analysis, it is observed that the success probability of selfish miner and majority attack decreases with increase in honest blocks and heavy computational challenge.

An experimental analysis of blockchain network creation is performed over Intel Core i5-7200U CPU@2.5 GHz, 4 GB RAM and 64-bit operating system. In this experimental analysis, variation in blocks mined per second is observed with increase in challenge. A challenge variation from 1-bit to Hash with exponentiation is analyzed and results are shown in Fig. 1. Results shows that there is exponential decrease in number of blocks mined with increase in challenge. 1-bit challenge to 35-bit challenge verification process is 'match and miss' process. In this process single or multiple players has to predict, compute or guess bits at specific positions. Increase in number of bits verified increased the trust.

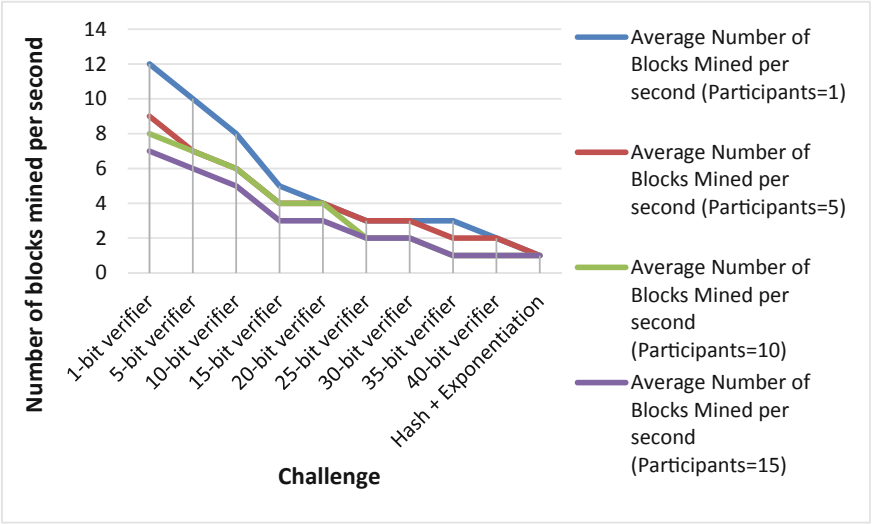

Fig. 1. Impact on number of blocks mined per second with variation in block challenge.

5 Conclusion

This work proposes a PoG consensus algorithm for blockchain network. It is observed that blockchain network can be protected from selfish miner and majority attack with increase in computational challenge. A heavyweight computational challenge consumes resources but provides higher security against selfish miner and majority attack. Whereas, lightweight computational challenge are suitable for resource constrained devices but provides comparatively lighter security against said attacks. However, a multi-round multi bit PoG concept for single or multiplayer game enhances the security for resource constrained devices as well. Thus, challenge variation from 1-bit to 35-bits is considered for analysis. Results shows that there is exponentiation decrease in blocks minded with increase in computational challenge but computational heavy challenge provides higher security. In single or multi-players PoG concept selfish and honest miners are identified through proper verification process. This verification process identifies the position of block in Merkle root tree and compares the block and transaction sizes. A large variation in block or transaction sizes indicates the chances of selfish miners. In future, single and multiplayer PoG will be extended for various lightweight cryptographic primitives and protocol based challenges in order to enhance the security levels.

References

1. All Cryptocurrencies. https://coinmarketcap.com/all/views/all/
2. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: Architecture, consensus, and future trends. In: 2017 IEEE International Congress on Big Data (BigData Congress), pp. 557–564. IEEE, June 2017

3. Wang, W., Hoang, D.T., Xiong, Z., Niyato, D., Wang, P., Hu, P., Wen, Y.: A survey on consensus mechanisms and mining management in blockchain networks. arXiv preprint arXiv:1805.02707, pp. 1–33 (2018)
4. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). http://bitcoin.org/bitcoin.pdf
5. Bach, L.M., Mihaljevic, B., Zagar, M.: Comparative analysis of blockchain consensus algorithms. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1545–1550. IEEE, May 2018
6. Jiang, S., Wu, J.: Bitcoin Mining with transaction fees: a game on the block size. In: Proceedings of the 2nd IEEE International Conference on Blockchain (Blockchain 2019), May 2019
7. Swanson, T.: Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. Report, April 2015
8. Dey, S.: Securing majority-attack in blockchain using machine learning and algorithmic game theory: a proof of work. In: 2018 10th Computer Science and Electronic Engineering (CEEC), pp. 7–10. IEEE, September 2018
9. Liu, Z., Luong, N.C., Wang, W., Niyato, D., Wang, P., Liang, Y.C., Kim, D.I.: A survey on applications of game theory in blockchain. arXiv preprint arXiv:1902.10865 (2019)
10. Dimitri, N.: Bitcoin mining as a contest. Ledger **2**, 31–37 (2017)
11. Stone, A.: An examination of single transaction blocks and their effect on network throughput and block size. Self-published Paper, June 2015. http://ensocoin.org/resources/1txn.pdf
12. Liu, X., Wang, W., Niyato, D., Zhao, N., Wang, P.: Evolutionary game for mining pool selection in blockchain networks. IEEE Wirel. Commun. Lett. 1 (2018)
13. Xiong, Z., Feng, S., Niyato, D., Wang, P., Han, Z.: Optimal pricingbased edge computing resource management in mobile blockchain. In: 2018 IEEE International Conference on Communications (ICC), Kansas City, Kansas, May 2018
14. Jiao, Y., Wang, P., Niyato, D., Xiong, Z.: Social welfare maximization auction in edge computing resource allocation for mobile blockchain. In: 2018 IEEE International Conference on Communications (ICC), Kansas City, Kansas, May 2018