Cluster Computing
https://doi.org/10.1007/s10586-023-04108-5

The cellular goore game-based consensus protocol: a cognitive model for blockchain consensus

Reyhaneh Ameri[1] · Mohammad Reza Meybodi[1]

Received: 13 February 2023 / Revised: 5 July 2023 / Accepted: 13 July 2023

Abstract
A blockchain is a distributed general ledger that, after authentication, records cryptographic transactions. Consensus algorithms are used to record transactions in the general ledger. Consensus algorithms are designed to achieve reliability in a network involving untrusted nodes. Although several protocols have been in operation for several years, they still have drawbacks. Artificial intelligence (AI) and blockchain are two technologies that have received a lot of attention in the last decade and are rapidly growing. Fascinating results can be obtained by applying AI to solve existing blockchain challenges. Cognitive blockchain employs artificial intelligence to improve performance and solve its challenges. Cognitive blockchain can perceive network conditions, analyze gained knowledge, effectively decide, and adapt to improve network performance. CGG, which models distributed systems with many simple objects that interact with each other locally, could be a suitable learning model for the cognitive blockchain because of the distributed nature of the blockchain network. We presented a new consensus protocol based on the Cellular Goore Game (CGG) as a cognitive model for blockchain consensus that could be used as intelligence consensus in the cognitive blockchain. The proposed consensus protocol partitions the nodes that take part in the consensus, organizes the CGG model to verify a new block after publication, and finally, based on the CGG's result, confirms or rejects the block. A new intelligent consensus approach improves fault tolerance, scalability, and performance and reduces participation costs in consensus for private and consortium blockchains. We conduct a qualitative study of the proposed protocol and analyze its experimental performance. The results show that the proposed schema performs well in the presence of numerous faulty nodes, as well as in terms of scalability and throughput.

Keywords Blockchain · Consensus · Cellular goore game · Learning automata · Cognitive

1 Introduction

Blockchain is a decentralized, distributed, peer-to-peer ledger that keeps a permanent record of interactions and transactions between users who have access to the decentralized blockchain network [1–3]. The core of a blockchain is a distributed ledger, and despite arbitrary behavior by malicious participants, the distributed collection of individuals maintains the consistency of the distributed ledger [4]. Consensus mechanisms are used to verify each block before being added to the chain. Blockchain offers a new way of coordinating multiple dishonest parties and making governance in existing networks more decentralized [5]. Decentralization, transparency, immutability, security, auditability, anonymity, and autonomy are the main characteristics of blockchain [6].

Blockchain could build new Internet infrastructure [7], but its use is limited by issues with scalability, environmental cost, security, and privacy. Scalability is the main reason blockchain can't be used as a universal platform for many different services and applications [8, 9]. Furthermore, most blockchains on the market presently consume a lot of energy. Most new blockchain platforms prioritize transactional throughput at the expense of other critical

✉ Reyhaneh Ameri
r.ameri@aut.ac.ir

Mohammad Reza Meybodi
mmeybodi@aut.ac.ir

[1] Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran

Published online: 03 August 2023

performance characteristics, such as decentralization, security, and latency.

A cognitive blockchain, introduced in [10], combines intelligent cognition and blockchain to improve performance and reach intelligence. A cognitive blockchain is a blockchain in which a cognitive process is embedded to optimize performance, making intelligent decisions based on the state of the network and using the consequences of its actions to improve future decisions. It can perceive the current network condition, assess network behavior and traits using sensed knowledge, and take appropriate actions in response to observations. The primary motivation behind developing the cognitive blockchain is to minimize human intervention by incorporating the cognition process. Due to the distributed nature of the blockchain network, in the cognitive blockchain, a cognitive engine set is embedded in the network nodes, which helps them in making a series of intelligent decisions. The use of this cognitive engine by nodes can be optional. Therefore, according to Fig. 1, the cognitive blockchain consists of several cognitive and simple nodes. A cognitive node is a node with the ability to optimize blockchain performance based on the situation at hand. Furthermore, compared to the cognitive node, a simple node lacks intelligence.

All nodes must agree on the protocol for updating the ledger's content to ensure the stability of the blockchain, and blocks cannot be accepted unless most nodes agree. The consensus technique is the only safe update solution in a decentralized distributed network with faulty and malicious nodes. CGG [11] can provide as a distributed model for many simple components that interact locally. This paper presents a novel intelligent CGG-based consensus method that enhances fault tolerance, improves scalability and performance, and is suitable for private and consortium blockchains. Every cognitive peer in the blockchain network can agree on the distributed ledger's current state using its embedded cognitive engine and the proposed consensus protocol's CGG-based strategy in a partially synchronous timing model. By using cognitive engines to automate consensus-related tasks, the proposed algorithm reduces participation costs, such as the nodes' time spent, and enhances security by minimizing message manipulation. Additionally, the suggested protocol uses public-key cryptography (PKC) [9] to asymmetrically encrypt data and verify its validity and the proper incentive mechanism to encourage participants. This paper provides the following contributions:

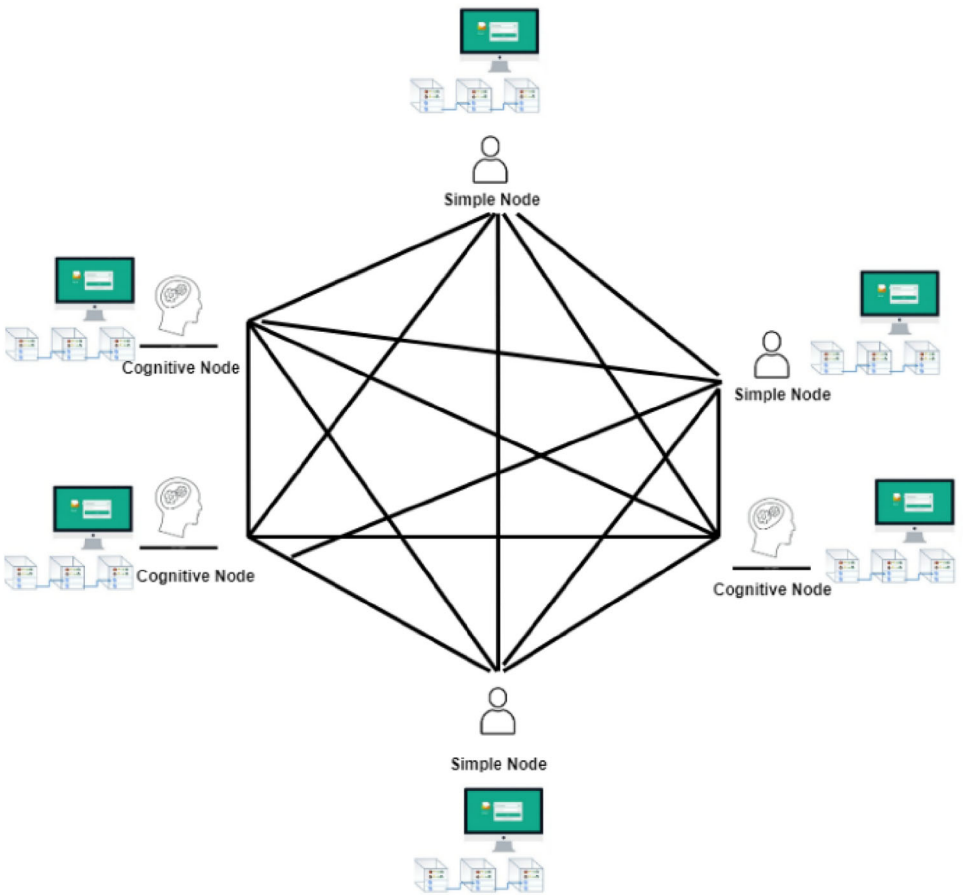


Fig. 1 Cognitive Blockchain Network

- We propose CGG-CP, a novel CGG-based consensus protocol, in this paper. The presented protocol can be used as intelligence consensus in the cognitive blockchain, which reduces the influence of failure nodes, and increases the performance and scalability of consortium blockchain systems.
- Because the presented consensus approach employs the cellular Goore model, the convergence speed of the CGG model has a significant impact on the efficiency of the proposed algorithm. We introduced strategies to speed up the CGG's convergence, for example, by adjusting the initial players' action probabilities.
- Finally, we performed a qualitative analysis of the CGG-CP and thoroughly assessed its performance in our blockchain implementation. We evaluated CGG-CP in terms of the defined evaluation metrics, such as success percentage, throughput, average latency, average block time, the average number of communications, and average reward, and compared CGG-CP to PBFT and RB-BFT [12]. The results reveal that CGG-CP improves performance in the presence of a high number of faulty nodes and reduces the average block time.

In the remainder of this article, we give some preliminary materials used in this paper in Sect. 2. We review the relevant literature in Sect. 3. In Sect. 4, we propose the new CGG-based consensus protocol. Then, we analyze the proposed consensus protocol in Sect. 5. The simulation results used to assess the suggested approach are shown and discussed in Sect. 6. The paper is concluded in Sect. 7.

2 Preliminaries

In this section, to give context for the remainder of the paper, we briefly overview Consensus Mechanism, Learning Automata, Goore Game, and the Cellular Goore Game.

2.1 Consensus mechanism

Scientists and articles have defined different consensus and consensus procedures [13–15]. Consensus is a collective decision-making procedure in which individuals support a decision in the group's best interest, even if it's not their preference. Consensus mechanisms control how participants reach an agreement and who can add new blocks to the chain. A consensus algorithm must meet the following four formal constraints [14–17].

- Agreement: No two correct nodes may decide on different values; rather, all correct nodes must agree on the same value.
- Validity: The chosen value is one that was proposed by one or more nodes.
- Integrity: If every individual correct node proposed the value v, then every correct process would have to decide on the value v.
- Termination: Eventually, all correct nodes must agree on the same value.

The safety, liveness, and fault-tolerance properties show a consensus mechanism effectiveness and applicability [10, 18]. Safety guarantees that the generated outputs are valid in accordance with the rules of the mechanism and that all nodes produce identical outputs for the same inputs they receive. The term "liveness" refers to the fact that all participating, non-faulty nodes in the consensus mechanism finally receive the submitted transaction and generate a value. Fault tolerance is provided if the consensus mechanism can recover from the failure of participating nodes. It should be mentioned that fault tolerance in distributed environments is classified into two types of faults: (1) stopping failure and (2) Byzantine failure. The stopping failure happens when hardware or software fails, and as a result, the node in the consensus mechanism stops responding. In the Byzantine failure model, faulty processes may exhibit entirely unconstrained behavior. Fault tolerance is crucial and critical for distributed ledger technology (DLT); hence, DLT-based systems typically select between liveness and safety according to their system's needs.

2.2 Learning automata

An adaptive decision-making system known as a learning automaton [16] can enhance its performance by learning how to select the best action from a list of possible actions through repeated interactions with the random environment. A learning automaton has a limited number of actions, and each action has an undetermined probability of being rewarded by its environment. Through numerous interactions with the system, the goal is to learn to select the action that has the most significant chance of rewarding oneself. The best action to take can be determined by the iterative process of interacting with the environment if the learning algorithm is appropriately selected [17].

Fixed and variable structure LAs are the two categories into which the LAs can be divided [16]. Sextuple...$P.G.T$, is used in this paper to represent the variable structure LAs. is a set of inputs actions (called response or reinforcement signal), is a set of internal states, is a set of outputs, P denotes the state probability vector, G is the output mapping, and T is learning algorithm. The probability vector is changed using the learning process. Based on the characteristics of, environments might be divided into the P-, Q-,

and S-models. The learning algorithm is used to modify the probability vector. A P-model environment's output consists of two components: success and failure. While lies in the interval [0, 1] in S-model environments, it can take a finite number of values in Q-model environments.

$$p_i(t+1) = p_i(t) + a(1 - p_i(t)) \\ p_j(t+1) = p_j(t) - ap_j(t). \forall j \neq i \quad (1)$$

$$p_i(t+1) = (1-b)p_i(t) \\ p_j(t+1) = \frac{b}{r-1} + (1-b)p_j(t). \forall j \neq i \quad (2)$$

Suppose $_i$ is the action selected at step t as an example realization from distribution P(t). The equation for updating probability vector P(t) in a linear learning algorithm is defined by (1) for a positive response (β = 1) and (2) for a negative response (β = 0). Reward and penalty parameters are given by the two parameters a and b, respectively. The parameter a (b) controls how much the action probabilities increase (decrease). R represents the total number of possible actions the LA may do. The learning process described above is known as the linear reward penalty (L_{RP}) if a = b, the linear reward penalty (L_{RP}) if a $>>$ b, and the linear reward inaction (L_{RI}) method if b = 0. The linear reward-penalty (SL_{R-P}) scheme, the linear reward e-penalty (SL_{RP}) scheme, and the linear reward-inaction (SL_{RI}) scheme, respectively, are the linear reinforcement schemes for the S-three model's possible values of a: a = b, a $>$ b, and b = 0 [18].

2.3 The goore game

In this part, we will explain the Goore Game and the Goore Game with LA. Tsetlin [19] first presented it, and Narendra & Thathachar [20] and Thathachar & Arvind [21] also provided in-depth analyses. The Goore Game has fascinating characters that can be solved entirely distributedly without any player-to-player communication. It is a cooperative game where participants can choose between yes or no options. The referee assesses the players' chosen voting. A unimodal performance evaluation function G is present in the referee. Each time, each player chooses one of their options, and the referee calculates the ratio λ, which is the proportion of players that voted "yes" to all players. The referee then distributes a dollar with probability G(λ) and independently assigns a dollar with probability 1—G(λ) to each player. The players then separately determine how to vote for the next iteration based on their profits and losses. The maximum of G(λ) is associated with the number of players who will answer yes after enough iterations. In most circumstances, a player might not even be aware of the existence of the other players or the total number of participants. Only the results of each player's choice of action need to be known.

Cooperative, mobile robotics [22], and Quality of Service (QoS) management in wireless sensor networks [23] are two areas where the GG has found critical applications. Tsetlin used his so-called Tsetlin automaton to solve the Goore Game when it was initially studied. Later, more research was conducted in the LA region, and many families there were successful at solving the Goore Game. For instance, Thathachar et al. [21] suggested using Variable Structure Learning Automata (VSLA) [24] to solve the Goore Game. A Learning Automata (LA) that represents each player performs actions by the player's strategies. Each LA updates its action probability using the LR-I learning algorithm [25]. [11] displays the pseudo-code for using GG with learning automata.

2.4 The cellular goore game

The Cellular Goore Game (CGG), described in [11], can be used as a model for systems with many simple, identical parts that interact locally. Every node (cell) in a CGG network can act as a referee while playing a Goore Game with its neighboring nodes. In CGG, each cell can act as a player and a referee simultaneously. CGG maximizes the objective functions in the referee nodes. Similar to GG, each player separately chooses their most appropriate action from two options based on the gains and losses they received from the nearby referees. In CGG, participants have no idea how other players behave or even how or why they are rewarded or penalized. The neighborhood structure required for specific applications is present in GGG, as GGG and CGG's differences. Additionally, it is possible to simultaneously maximize multiple criteria by including multiple referees at once in the suggested model.

In the proposed CGG, each player is represented by a learning automaton with two actions (YES and NO), which updates their action probability vector using the LR-I learning algorithm. A CGG with learning automata has been defined as a structure of the form (N, P, LA, R, G), where P is a subset of V acting as the players, and N = (V, E) is an undirected network that specifies the structure of the CGG. Additionally, R is a subset of V acting as referees. Additionally, G = {G1, G2, … Gn}is a collection of unimodal performance criteria for the referees. LA = {LA_1, LA_2, …, LA_n} is a set of learning automata given to cells. For Quality-of-Service (QoS) control in WSNs when clusters overlap, one for clustered WSNs and one for WSNs with multiple sinks, [52] has employed the CGG model.

3 Related work

Consensus algorithms are classified into two types: proof-based and vote-based [13]. The proof-based consensus technique is based on the assumption that the node in the network that provides sufficient proof will add a new block to the chain and receive a reward. Whereas in vote-based consensus algorithms, miners are chosen through a voting process. Numerous different consensus algorithms based on votes and proofs have been presented. In the section below, we review the well-known mechanisms in these classes.

Many proof-based consensus techniques with various ways to select miners have been demonstrated in the literature. Miners, for example, are chosen based on their computing power, stake count, reputation, and storage capacity. The proof-of-work (PoW) protocol [26] was the initial decentralized consensus protocol proposed to maintain the stability and security of the Bitcoin network. Its main idea is that nodes compete for hashing power to decide who gets accounting rights and rewards [27]. Nodes calculate a mathematical problem's answer using prior block data. The first node to solve this math problem creates the next block and receives a reward. PoW is a widely used and popular consensus protocol, but it uses a huge amount of energy in networks and increases transaction confirmation delays. Proof of stake (PoS) is an alternate technique to PoW that selects the next mining node based on its possession of the blockchain network's digital currency [28].

PoS improvements include leased proof of stake (LPoS) [29], delegated proof of stake (DPoS) [30], and proof of stake velocity (PoSV) [31]. Regardless of the number of coins they possess, stakeholder participants in DPoS use the voting method to choose witnesses (as members of a delegation). They will validate the transactions and produce new blocks after choosing the witnesses. In LPoS, nodes with fewer coins can rent out their stakes to nodes with more coins. PoSV uses an exponentially increasing function rather than a linear function, like in PoS, to address the problem of cash flow. PoSV continues to favor nodes with more currency. The lower transaction flow in PoS is addressed by the proof of importance (PoI) [32]. The PoI selection procedure is based on the nodes' assigned significance scores.

Other types of proof-based consensus algorithms include proof of burn (PoB) [33] and proof of space (PoS) [34]. In a proof-of-burn system, miners must submit their coins to an address where they will be burned; once sent, these coins will be deleted from the network and rendered useless to other users. The ability to mine a new block will go to the miner who burns the most money at any given moment. In proof of space, miners have a better probability of mining a new block if they have more storage capacity (i.e., hard disks that are less expensive than PoW computing machines). Miners are chosen for the proof of elapsed time (PoET) [35] in a unique setting called the trusted execution environment (TEE) [36]. The node with the shortest waiting time can mine a new block in PoET, where each node has a variable waiting time duration.

When using voting-based consensus techniques, the network's nodes must jointly validate the transactions or blocks. The Byzantine consensus algorithm is significant in voting-based consensus. Lamport et al. [13] introduced the BFT algorithm that declared an allegory for the problems of achieving consensus in a decentralized system. BFT-based consensus algorithms are the set of consensus algorithms designed to solve the problem of Byzantine fault-tolerant and reach consensus on data in an environment with the possibility of malicious nodes. A distributed network's ability to reach consensus even when some nodes fail to react or provide inaccurate information is known as Byzantine Fault Tolerance (BFT). A BFT mechanism's goal is to prevent system failures by using collective decision-making (both from correct and faulty nodes), which attempts to minimize the impact of the faulty nodes. PBFT, DBFT, and Tendermint consensus algorithms are among these algorithms. PBFT [37] tolerates byzantine failures. Hyperledger Fabric [38] uses the PBFT consensus technique because it can handle about 1/3 of malicious byzantine replicas. Rounds determine new blocks. Rules will determine each round's primary [39]. The entire process can be separated into three phases: pre-prepared, prepared, and commit. PBFT is a consensus algorithm with proven security and liveness, but it cannot scale due to network overhead, reducing system throughput [40]. Another vote-based consensus is DBFT [42], or delegated Byzantine fault tolerance. The dBFT is the PBFT-based consensus mechanism that improves PBFT performance by choosing a relatively small number of backup nodes by voting as the number of consensus nodes grows. A transaction is deemed genuine if at least two-thirds of the nodes acting as witnesses to it agree. e-PBFT [45], in which the main node mechanism in the consensus process is removed and power is distributed evenly throughout the system's nodes. However, there is no effective constraint on the malicious activity of system nodes.

Ripple [41] and Stellar consensus protocol (SCP) [42] improve PBFT by achieving consensus in federates, which are smaller networks. Ripple, as voting-based algorithm, uses trust-based subnetworks to reach consensus and divides nodes into two types: servers to participate in the consensus process and clients only to transfer funds. This consensus protocol utilizes a raw ledger instead of a chain of blocks, and after a few confirmed periods, transactions

are added directly to the ledger. SCP, as an implementation of the Federated Byzantine Agreement (FBA) consensus method, contains an open membership network in which all nodes know each other and may value some more than others. Any node that has to verify a transaction waits for the great majority of its important nodes to agree with it. The transaction cannot be finilazed until the important participants agree. Tendermint [43] is considered a variant of PBFT, and its important difference from PBFT is the continual rotation of the leader. Each node in Tendermint must lock its coin to become a validator. These locked coins will then be used to reward or punish the validators who contribute.Tendermint reduces the message complexity of view change.

The PBFT consensus algorithm and its variants are subject to several attacks on the primary node and are barely able to detect and eliminate faulty nodes in the blockchain system. To solve these issues, Lei et al. [46] suggested a Reputation-Based Byzantine Fault-Tolerance (RBFT), which assesses each node's contribution as well as its reputation during the voting process. If any malicious behavior is found, a malfunctioning node will receive a lower reputation and, hence, less right in the consensus process. When verifying new blocks, the RBFT algorithm prioritizes nodes with a higher reputation. The Weighted Byzantine Fault Tolerance (WBFT) [44] consensus algorithm, which improves system throughput and consensus delay, added a dynamic weighting mechanism for consensus nodes. This improves the security of the blockchain system by reducing the influence of malicious nodes and the probability of malicious behavior. [45] presents an improved algorithm, tPBFT (trust-based practical Byzantine algorithm), for consortium chain high-frequency trading. A trust equity scoring technique can dynamically adjust the consensus node list. tPBFT simplifies the pre-prepare stage and verifies the hashed transaction list at the reply stage, minimizing network node interaction overhead. [12] present an improved Byzantine fault-tolerant consensus called RB-BFT by developing a node reputation model to improve the reliability of PBFT's primary node and reduce the high communication complexity of PBFT. The proposed approach statistically evaluates node behavior in consensus to choose high-trust nodes as primary nodes to improve system security.

Blockchain technology relies on consensus methods to secure, reliable, and resilient the network. Different consensus algorithms each have their advantages and drawbacks, and use cases and needs determine which consensus algorithm is appropriate to use. Table 1 provides a comparison between the most popular consensus algorithms. Proof-based consensus algorithms such as PoW and PoS are mostly decentralized, do not require trust between people, and are more suitable for public blockchains. Low efficiency and probabilistic transaction confirmation make them inappropriate for fast transaction processing applications. Transactions with voting-based consensus algorithms are finalized definitively and faster and use less energy than proof-based algorithms. Voting-based algorithms, on the other hand, require a trusted set of validations and are less safe on open networks. Efficiency decreases with an increase in network nodes, and it cannot exceed a specific limit, so decentralization is limited. In recent years, artificial intelligence (AI)-based approaches have been proposed to solve issues related to blockchain consensus protocols, which will be discussed in the following.

In [46], Salimitari et al. presented a new framework for secure and robust consensus in blockchain-based IoT networks using machine learning. This paper proposes a two-stage consensus process involving an outlier detection algorithm to verify data consistency, discarding suspicious data, and PBFT as the consensus algorithm. Most of the existing research increases the performance of BFT algorithms in two ways: one is to optimize the architecture, and the other is to reduce the number of nodes involved in consensus by utilizing trusted institutions [47, 48]. A decision-theoretic online learning-based methodology for choosing a subset of nodes to take part in the consensus process in BFT-based consensus methods is proposed by Bogdi et al. in [49]. The suggested model successfully chooses the consensus committee's nodes with a higher reputation. The reputation value is determined for the nodes that want to participate in the consensus committee to decrease the probability that the nodes in the consensus committee will be damaged. Nodes with high reputation values are chosen for the consensus committee. Response time, stock quantity, and response type have been used as features to measure the validity of nodes.

Liu et al.'s proposal [50] for deep reinforcement learning-based performance optimization in blockchain-enabled Internet of Vehicles maximizes transactional throughput while maintaining the underlying blockchain system's decentralization, latency, and security. This study also measures blockchain systems' scalability, decentralization, latency, and security performance. Additionally, Liu et al. [51] present a method for increasing the transactional throughput of the Industrial Internet of Things powered by blockchain by choosing the block producers and consensus algorithms (from the Quorum [52], Zyzzyva [53], and PBFT [54] consensus algorithms) and modifying the block size and block interval using the DRL technique. This strategy tries to increase the underlying blockchain's scalability without compromising the system's decentralization, latency, or security. The Credit Reinforcement Byzantine Fault Tolerance Consensus (CRBFT) algorithm, proposed in [55], assigns the credit attribute to the node by

Table 1 A comparative review of blockchain consensus algorithms

Consensus ALGORITHM	Blockchain type	Transaction finality	Throughput (TPS)	Tolerated power of advisory	Cost of participation	Scalability of network	Trust model	Energy saving	Example
PoW [26]	Permissionless	Probabilistic	7–30	< 25% computing power	Yes	High	Untrusted	No	Bitcoin [26]
PoS [28]	Both	Probabilistic	30–173	Depended on specific algorithm used	Yes	High	Untrusted	Partial	Peercoin [59]
DPOS [30]	Both	Deterministic	25–2500	< 51% validators	No	High	Untrusted	Partial	Bitshares [30]
PoI [32]	Both	Probabilistic	10,000	< 50% importance	No	High	Untrusted	Yes	NEM [60]
PoB [33]	Permissionless	Probabilistic	–	< 50% of stack	Yes	Medium	Untrusted	Yes	Slimcoin [61]
proof of space [34]	Permissionless	Probabilistic	80	< 25% computing power	Yes	High	Semi-trusted	Yes	SpaceMint [62]
PoET [35]	Both	Probabilistic	–	< 50% trusted execution environment nodes	Yes	High	Untrusted	Yes	Sawtooth Lake [35]
PBFT [37]	Permissioned	Deterministic	100–2500	< 33% replicas	No	Low	Semi-trusted	Yes	Hyperledger Fabric [38]
Ripple [41]	Both	Deterministic	50–1500	< 20% faulty nodes in UNL	No	High	Semi-trusted	Yes	Ripple [41]
SCP [42]	Both	Deterministic	50–1000	-	No	High	Semi-trusted	Yes	Stellar [42]
Tendermint [43]	Permissioned	Deterministic	> 1000	< 33% byzantine voting power	No	High	Semi-trusted	Yes	Tendermint [43]

utilizing reinforcement learning that adaptively alters the node credit. The suggested method can automatically identify malicious nodes in the consensus network, enhancing consensus network security, decreasing consensus delay, and maximizing energy savings.

[56] improves scalability in PBFT by applying reinforcement learning to adjust the number of participating nodes based on their behavior and the existing network condition. The number of participating nodes is decreased without negatively affecting the security or liveness of the protocol. Goh et al. suggest in [57] a trust-based delegated consensus blockchain framework that combines the DRL and delegated consensus techniques to improve security performance and blockchain throughput in blockchain-enabled IoT networks. The proposed schema evaluates node reliability and selects reliable nodes for the consensus process to reduce malicious activity. For large-scale IoT services, the authors of [58] develop a deep Q-network-based sharded blockchain, where the shards increase system scalability and the deep network determines the optimum throughput configuration.

The majority of the studies presented to solve issues related to blockchain consensus protocols focused on optimization and scalability in a specific use case, with the solution provided depending on the application. Furthermore, most proposed solutions have only focused on improving scalability or performance, and their approach may weaken other aspects of blockchain, such as decentralization and security. For instance, most reputation-point-based strategies can cause a more centralized structure. Also, the proposed solutions in this context have not addressed the details of how to implement their algorithm in a distributed blockchain environment. For example, the suggested algorithm's executor in studies that select a committee of consensus nodes is unknown. To address the mentioned issues, we propose a novel intelligence consensus named CGG-CP, which promotes the decentralization and scalability of consortium blockchain systems while decreasing the impact of failed nodes. Our proposed consensus algorithm applies to consortium blockchains and isn't limited to any particular application. Furthermore, all of the details of how to execute it in a distributed environment are provided in the cognitive blockchain framework, which is discussed in the following sections.

4 The CGG based consensus protocol

In this section, a novel intelligent voting-based consensus algorithm for the consortium blockchain is presented. This proposed CGG-based Consensus Protocol, named CGG-CP, can be used as intelligence consensus in the cognitive blockchain, introduced in [10], which is a blockchain that has a cognitive process embedded into it to maximize performance. The suggested consensus protocol is a CGG-based approach that enables every cognitive peer in the blockchain network to agree on the distributed ledger's current state in a partially synchronous timing model. Also, the proposed protocol validates the authenticity of data using asymmetric encryption through public-key cryptography (PKC) [63]. In the CGG-CP, nodes with more credibility have more influence on confirming or disconfirming blocks. Also, it is required to include a suitable incentive mechanism for motivating the participants in the consensus process. The CGG-CP is explained in more detail in the following.

The block verification procedure of CGG-CP is described as a distributed optimization problem that will be solved using the CGG model. After publishing a new block, to confirm or reject its validity, the CGG is formed. Then the CGG's result is used based on a pre-specified policy to confirm or reject the block. In CGG-CP, the underlying graph of CGG that determines its structure can be formed based on the list of trusted people announced by the network nodes and predefined policies or organizational structures in the specific applications of this consensus algorithm. In graph development through trusted individuals, network nodes declare a list of trusted nodes as validators. Predefined policies can limit the number of trusted nodes that each node defines as its reliable nodes. At predetermined intervals, the underlying graph can be updated through nodes' re-announcements of their list of trusted people. We define four roles in the CGG-CP: referee, player, owner, and ordinary role. Nodes that are trusted by more than the defined threshold are selected as referees. Also, nodes with at least one referee on their list of trusted nodes will be in the player role. One of the referee nodes will take up the owner's role, rotating at random. The owner should create the block, decide whether to approve or reject it based on the CGG's results, distribute the reward to other nodes, and finally add the block to the blockchain. Nodes that cannot act as referees or players are ordinary nodes. Referees, players, and owners must utilize the cognitive engine, but ordinary nodes need not apply it. The organizational hierarchy or predetermined relationships between people in specific use cases can determine the underlying CGG graph. For instance, the blockchain network may be made up of members of numerous interconnected organizations, and the leaders of each organization serve as referees for the people in their organization who act as players. The proposed consensus algorithm can also involve stakeholders associated with these organizations as ordinary nodes.

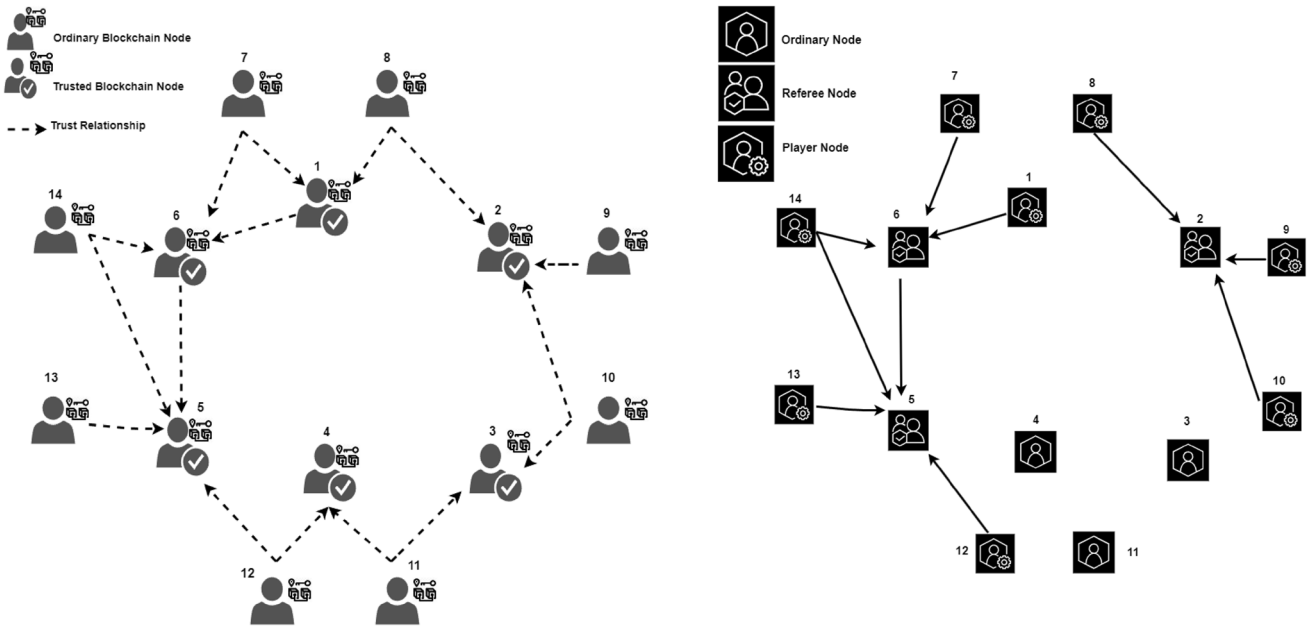
We assume that the cognitive blockchain consists of n cognitive nodes denoted by $CN = \{CN_1, CN_2, \ldots, CN_n\}$ in which the cognitive engine of CGG-CP is embedded.

Based on the list of approved nodes, the role of cognitive nodes is defined in the proposed method, as already mentioned. Player cognitive nodes are represented by CNP = {CNP_1, CNP_2,…, CNP_p} while referee cognitive nodes are represented by CNR = {CNR_1, CNR_2,…, CNR_r}. The CNP and CNR are subsets of the CN. Also, there is a possibility of overlap between CNP and CNR. It means that the cognitive node could play both the roles of player and referee at the same time. Other cognitive nodes not in the CNP and CNR have an ordinary role and aren't involved in CGG-CP block validation. A graph N = (V, E) can depict the CGG network, where V is the set of cognitive nodes with the roles of referee and player, and E is the edges that include trust relations between cognitive nodes, defined by Eq. 3. If there is an edge in the graph of the CGG network between two nodes, they are adjacent. Figure 2 depicts an example of the process of formation of the underlying graph in CGG-CP. As seen in Fig. 2. a, there are 14 nodes in this network, and nodes 1 through 6 are known as trusted nodes by other nodes. In the CGG network illustrated in Fig. 2. b, nodes 2, 5, and 6 serve as referees since they are trusted by more than two nodes, while the other nodes that voted for them are the players.

$$E = \{ \left(CN_i, CN_j\right) | CN_i \in CNP \& CN_j \\ \& \, CN_j are in the trusted nodes of CN_i \} \quad (3)$$

We classify nodes as either honest or malicious, and both categories can exhibit stopping failure by stopping somewhere in the middle of their execution. A malicious node may act arbitrarily, whereas an honest node strictly adheres to the consensus protocol. We assume that the node serving as owner are honest and unable to stop. Also, sending packets by nodes in the consensus process is done only through their cognitive engines, and it is not possible to manipulate this cognitive engine for malicious actions by digital signature. Figure 3 illustrates the presented CGG-based Consensus Protocol. These steps describe CGG-CP block validation:

- Construct the block: The owner generates the block from the queue of unconfirmed transactions received from nodes.
- Broadcast the block: The owner broadcasts the block to the referees, and then each referee broadcasts the block to the adjacent players.
- Determine the referees' performance criteria: Each referee evaluates the block and decides between the three alternatives of approval, non-approval, and the requirement to collaborate with the adjacent players. The performance criterion of the referee in CGG is determined based on the referee's decision, explained in the following. If the referee decides to collaborate with the players, the players should vote to determine the performance criteria for the referee based on their opinion about the block's validity.
- Set the players' action probability: Players get the block and verify its correctness. The action probability vector will be specified depending on their opinion of the acceptance or rejection of that block. However, if for any reason—including their absence—no answer is



a) The announcement of trustee nodes

b) The underlying graph of CGG-CP

Fig. 2 The process of formation of the underlying graph in CGG-CP

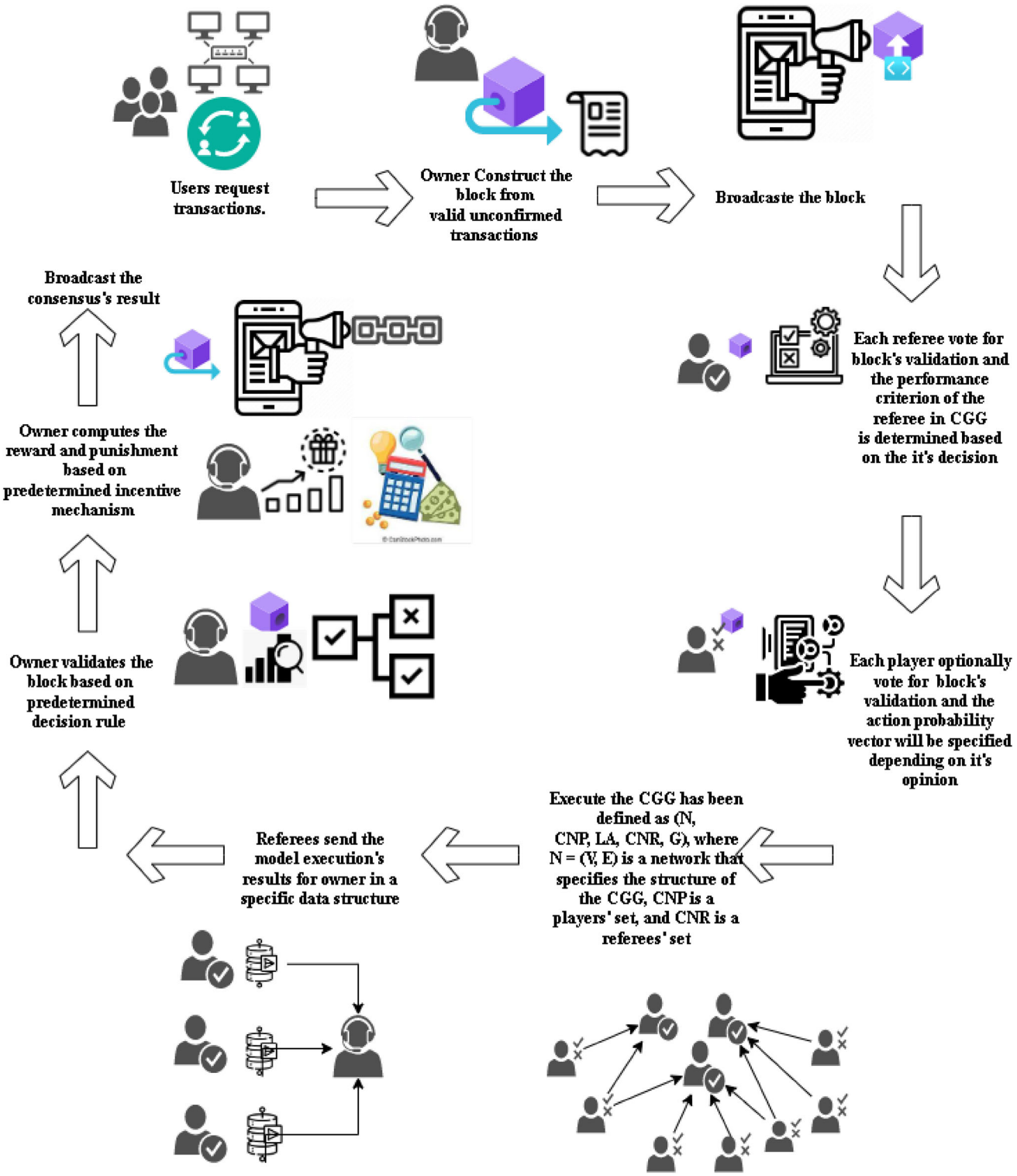


Fig. 3 The steps of CGG-based Consensus Protocol

received from them within the allowed time frame, the action probability vector with an equal probability will be considered. One of the goals of setting the initial probability vector value of the players' action choices is to speed up CGG convergence and, as a result, consensus.

- Execute the CGG: In the proposed CGG, each player is represented by a learning automaton with two actions

(yes and no), which updates their action probability vector with the initial value specified in the previous step using the L_{R-I} learning algorithm. The structure of a CGG with learning automata has been defined as (N, CNP, LA, CNR, G), where N = (V, E) is a network that specifies the structure of the CGG, CNP is a players' set, and CNR is a referees' set, as already described. In addition, LA = $\{LA_1, LA_2, \ldots, LA_n\}$ is a set of learning automata assigned to cells. Also, G = $\{G_1, G_2, \ldots, G_n\}$ is a set of referees' unimodal performance criteria determined in the previous stages. The defined CGG model should be executed. Every referee in each round of CGG plays a GG with their adjacent players. Based on predefined principles such as convergence and the maximum number of executive rounds, each referee declares the end of its GG to the neighboring players.

- Send the model execution's results: The referees must send the owner a series of statistical data regarding the result of the game they played with their neighboring players, such as their chosen criterion function and the acquired convergence value, in the form of a specific data structure.
- Determine the block validation: The owner then determines whether to approve or disapprove the block based on predetermined decision rule after receiving the data sent by the referees within a certain period. The number of adjacent players should determine how much weight the referees' opinions have in the decision rule. The majority opinion should be the most influential factor in making a decision.
- Compute the reward and punishment: The owner then computes the reward and punishment for players and referees based on a predefined incentive mechanism. The owner can temporarily exclude some nodes from participating in the consensus process as punishment. It is not always suitable to employ tokens or coins as an economic incentive in a blockchain consortium. So, in CGG-CP, the rewards given to nodes can help them gain the trust of other nodes in the network and be chosen as trusted nodes in later rounds. Furthermore, because referees incur a higher cost in order to participate in the consensus process, their incentives are typically significantly greater than those of the players. Furthermore, referees are rewarded based on the number of players in their group and the convergence speed of their GG.
- Broadcast the consensus's result: If the block is accepted, the owner completes the block with the necessary metadata and adds it to the blockchain. It sends the blockchain to the referees, who then transmit it to their neighboring players. The blockchain is then broadcast to ordinary nodes by the players.

In each block confirmation round, Each referee node has a specific uni-modal performance criterion which is maximized at the ratio of the predefined value, based on its decision to determine the referees' performance criteria step, to the total number of adjacent players. The following is a general definition of a uni-modal performance criterion:

$$G_i(\lambda_t) = c + ae^{\frac{-\left(\lambda_i^* - \lambda_t\right)^2}{b}} \quad (4)$$

λ_t is the proportion of adjacent player nodes with a "yes" action to all of the adjacent players for the i^{th} referee in round t of the CGG and c, a, and b are learning parameters. The optimal ratio, λ_i^*, has a predefined value near 1 if the block is approved by the i^{th} referee and a predefined value close to 0 if it is rejected. If the i^{th} referee chooses to collaborate with the players, they should vote to determine the i^{th} referee's optimal ratio based on their evaluation of the block's validity.

In the described steps for CGG-CP, it is necessary for the cognitive nodes to perform some operations, although most of these operations are performed automatically through the cognitive engine embedded in them, and it is only necessary that their access to the network is not disrupted. For example, the player's role must select its action and send it to the adjacent referee during several rounds of the CGG execution step, which is done automatically through its cognitive engine. So there is no need to take any action from the user's side with the player's role in each round in executing the CGG step. However, for a block evaluation by referees, the referee must be available to check that block at that time. In a few cases where the referee is not available to confirm the validity, the option of needing to participate with the players can be chosen automatically at the stage of determining the criterion function, but the penalty points for these situations should be taken into account when determining the reward. The use of cognitive engines to automate some consensus-related tasks reduces the participation cost for nodes and enhances security by minimizing the possibility of message manipulation.

In executing the CGG in the proposed consensus, the referees can detect the stopping failure of their adjacent players through the failure detection mechanism defined on their cognitive engines. If the referee's cognitive node does not receive any response from its player during several CGG rounds, it will temporarily remove this node because of a stop failure from the adjacent players in this CGG model. The efficiency of the proposed algorithm is significantly impacted by the convergence speed of the CGG model. We implemented strategies to speed CGG's convergence, including motivating players to validate blocks and adjusting initial action probabilities based on their

opinions of block confirmation or rejection. One of The objectives of CGG-CP is to overcome the issues of voting-based consensus algorithms, which can only achieve prompt consensus in networks with a limited number of peers in a somewhat centralized manner. The proposed consensus method doesn't require each consensus node to send their vote to all nodes, unlike some voting-based consensus algorithms like pbft. Instead, the CGG-CP involves the nodes with referee, player, owner role as validating nodes in consensus and nodes are connected in a neighborhood structure based on the CGG network graph. In order to reach a consensus, each referee exchanges messages with his neighboring players as well as the owner, and the players also communicate with their neighboring referees. In addition, the proposed protocol requires the owner to send and receive messages with all referees in order to reach a consensus. The CGG-CP guarantees high network scalability and allows for many validation nodes to reach consensus in the blockchain in a decentralized manner while still keeping transaction speed high.

Within the CGG-CP, the nodes have different roles and functions. The referee nodes, which have high trustworthiness, are accountable for confirming transactions and creating blocks. To obtain the correct blocks, nodes must choose the correct nodes as trusted nodes. Therefore, the use of the consensus algorithm on a public blockchain where the reliability of nodes remains unknown is not appropriate. The proposed consensus approach is suitable for consortium blockchains in which nodes are organized into access-controlled networks. The network is comprised of validator nodes that initiate, receive, and validate transactions, and member nodes that can only receive or initiate transactions in this type of blockchain. The CGG-CP consensus protocol is suitable for consortium blockchains as it involves organized nodes and distinct roles to achieve consensus. By creating the underlying network and grouping nodes with a defined organizational structure, the suggested consensus protocol can be adapted for private blockchains.

5 Analysis of the proposed consensus protocol (CGG-CP)

In this section, we perform an in-depth analysis of the CGG-CP in terms of fault tolerance, safety, liveness, participation cost, and complexity.

5.1 Fault tolerance

Byzantine and stopping failure models described in Sect. 2.1 can occur in the proposed consensus algorithm. Another process failure that can happen in the CGG-CP is the unavailability failure model. If nodes cannot perform their assigned tasks in the consensus process within the time limit, they exhibit an unavailability failure model. In CGG-CP, the task assigned to players is to verify the block and vote on its acceptance or rejection in the set players' action probability step. Also, the responsibility assigned to referees is to verify the block and vote on its acceptance or rejection in the determine the referees' performance criteria step. Referees and players may not confirm the block within the time limit for any reason. The proposed consensus algorithm can handle this failure models. CGG-CP's fault tolerance level depends on its decision rules and predetermined policies, such as the policies to create the graph and execute CGG and the decision rules for block validation. Our assumption is that the owner node won't experience any failure and the proposed method of consensus is not resilient to owner failure. Due to the proposed protocol's method of selecting the owner's role from referee nodes that are trusted by other nodes and rotating it alternately, the assumption seems plausible. Also, if a node with an ordinary role fails, it won't affect the consensus process because those nodes have nothing to do with block validation.

Players and referees could show a Byzantine failure by giving a false decision about block approval in the block validation of CGG-CP. If the Byzantine-faulty referee evaluates the block incorrectly, a sufficient number of its adjacent non-faulty players can prevent its GG from converging to the desired amount. Therefore, it won't be given any reward for taking part in consensus and may even be penalized, such as by a reduction in its credit as a reliable node. Assuming that enough non-faulty referees participate, the consensus result won't be significantly affected by a Byzantine-faulty referee. If a player wrongly validates the block, adjacent referees with enough non-faulty adjacent players will have a slower convergence, which will cause a decrease in rewards for this faulty player. One of the byzantine fault tolerance mechanisms is the cognitive engine embedded in the nodes and the impossibility of manipulating it via a digital signature in the proposed consensus. For example, each referee calculates and sends rewards through its embedded cognitive engine to adjacent players in the CGG step and cannot manipulate their rewards. In another instance, based on the criterion function of the referee and the actions of its adjacent players, the referee's embedded cognitive engine calculates the outcomes of the game played and reports them to the owner. Therefore, referees cannot alter the results of the model's execution. The proposed protocol reduces the motivation of consensus nodes to perform byzantine actions due to their known identities and the mechanisms used to reward and punish them.

In the proposed schema, if a node stops in the middle of its execution, this can cause a stopping failure. In the CGG-CP, when players have a stop failure, another way to improve fault tolerance is through the failure detection mechanisms of the referees. If the referee has a stop failure, the referee and the neighboring players will not affect the consensus process and will not receive a reward in that round. In the failure scenario where the referee is unavailable to validate a block, the option of participating with the players can be chosen automatically during the criterion function determination step. The referee will get a punishment, while the players will obtain significant rewards. When a player is unavailable to validate the block, an equal probability vector is used as its initial probability vector. If most of a referee's players are unavailable, the convergence speed of its GG decreases. In these cases, the referee can stop the GG before reaching convergence based on predetermined principles, such as the maximum number of executive rounds.

5.2 Safety

To demonstrate that consensus protocols are safe, it is enough to show that they have agreement, validity, and integrity properties. In the CGG-CP algorithm, finally, the owner determines the block validation based on the results of the CGG execution transmitted by the referees and the decision rule and then sends the result to others. So, the agreement is achieved, and all non-faulty nodes agree on a single decision about the block. The referees express opinions about the block's validity in the proposed consensus protocol. Their criterion function for rewarding the players is also set, and the CGG model will eventually converge. Then, the referees send the owner the convergence results, their criteria function, and other important information. The final decision must be the decision of other referees, and the validation property is also true in this protocol, given that the owner is non-faulty and that the rule for the owner's final decision is based on the opinion of the vast majority of non-faulty referees.

5.3 Liveness

"Liveness" means that all non-faulty nodes in the consensus method finally get the submitted transaction and generate a value. All valid clients' transactions will appear in the candidate block by the owner. If the steps in the proposed consensus protocol are followed to confirm the block, the block will be added to the blockchain, and the transactions in that block will be in the confirmed state. If not, the valid transactions will be added to the next block and finally approved.

5.4 Participation cost

The "cost of participation is defined as the amount of costs that each node should pay to participate in the consensus process. The cost can be the amount of energy, time, money, or anything else that blockchain nodes need to spend for consensus. CGG-CP uses the minimum amount of resources to validate blocks. Block generation rights aren't directly correlated with processing power [64], as they are in PoW, nor are they correlated with accumulated stake, as they are in PoS, so CGG-CP reduces the waste of computational resources. In the proposed algorithm, the nodes that others trust the most, including the referee node, validate the blocks with the cooperation of their players, using the cognitive engine embedded in these nodes and the CGG model. Nodes only pay for network communication costs in this approach. The nodes' embedded cognitive engines automatically perform defined consensus tasks. So the nodes' time spent participating in consensus has been reduced. Also, an incentive mechanism has been developed to encourage the nodes to participate in as much activity as possible.

5.5 Complexity

The two most usual approaches to quantifying complexity in consensus algorithms are time complexity and communication complexity. For all distributed algorithms, the time required to perform a task is one of the most important measures. We assume that the cognitive blockchain consists of N nodes, P player nodes, and R referee nodes, denoted by CN, CNP, and CNR, respectively. Also, K is the maximum number of executive rounds for each referee, and L_R and L_P are the average time it takes for referee and player nodes to respond, respectively. Furthermore, $|E|$ demonstrates the size of graph edges. According to the CPP-CP, first, the owner broadcasts the block to the referees, who then broadcast it to the adjacent players. It has a time complexity of $O(L_RR + L_PP)$ and a communication complexity of $O(R + P)$. Next, it takes $O(L_RR + L_PP)$ time and $O(R + P)$ communication complexity to determine the referees' performance criteria and set the players' action probability vector. According to the specified CGG's time complexity in [11], the time complexity of CGG execution is equal to $O(KL_R|E|/(P + R))$, and the communication complexity is $O(K|E|/(P + R))$.

Then, the owner must determine the block validation and compute the reward and punishment in $O(C_{validity-} + (P + R)C_{reward})$. Eventually, the time complexity of broadcasting the consensus's result is $O(L_RR + L_PP)$. Consequently, CGG-CP's time complexity is $O(L_R-R + L_PP) + O(L_RR + L_PP) + O(KL_R|E|/$

$(P + R)) + O(C_{validity} + (P + R)C_{reward}) + O(L_{R\text{-}}$ $R + L_PP) = O(KL_R|E|/(P + R)) + O(L_RR + L_PP)$. Furthermore, the communication complexity of broadcasting the consensus result is $O(R + P)$. CGG-CP communication complexity is $O(R + P) + O(K|E|/(P + R)) + O(R + P) = O(K|E|/(P + R)) + O(R + P)$. So, the time and communication complexity of CGG-CP depends on the number of edges in the underlying graph, K and the number of players and referees. The complexity grows linearly as the number of nodes in the consensus group increases.

6 Experiments

This section conducts several experiments to evaluate the suggested consensus protocol. To thoroughly assess the effectiveness of the CGG-CP, we simulate a consortium blockchain as a decentralized peer-to-peer network using the Python programming language. We simulate the blockchain's multiple-node environment by using multiple ports. All results reported in this section are averages based on 30 tests to reduce statistical bias. We execute the experiments on a PC with a single Intel(R) Core i7-8550U 1.80 GHz CPU and 12 GB of RAM. We describe the general simulation parameters in Table 2. In the rest of this section, we first explain the evaluation metric used in our experiments and then describe a series of experiments.

6.1 Evaluation metrics

We presented a performance analysis of the suggested consensus protocol using the average entropy, average performance function, success percentage, throughput, average latency, average block time, the average number of communications, and average reward as presented in the paragraphs below. Entropy can analyze how the learning automata's states change. The equation below gives the average entropy at iteration t of the LA-based model and the total average entropy. The number of learning automata in the model is given by n, and each learning automaton's actions are given by r_i. The probability that the i^{th} learning automaton will select action α_j at iteration t is $p_j^i(t)$. H(t) value will be 0 if all learning automata in the model stabilize their chosen action. Higher entropy values denote higher rates of change in the learning automata's chosen actions. $\overline{H}$ denotes the total average entropy during the T episodes of block production in the blockchain. The scaled average performance function (G_{avg}), which was introduced in [11], can investigate the behavior of the CGG model. The total average G_{avg} ($\overline{G_{avg}}$) represents the scaled performance function's average during the CGG model executions in adding blocks to the blockchain, as formulated in the equation below.

$$
\begin{aligned}
H(t) &= -1/n \sum_{i=1}^{n} \sum_{j=1}^{r_i} p_j^i(t).\ln\left(p_j^i\right) \\
\overline{H} &= -\frac{1}{T}\sum_{t=1}^{T} H(t) \\
\overline{G_{avg}} &= -\frac{1}{T}\sum_{t=1}^{T} G_{avg}(t)
\end{aligned}
\tag{5}
$$

We represent the percentage of blocks that have been correctly decided using the consensus method as the success percentage. It should be mentioned that because it is

Table 2 General simulation parameters

Simulation parameter	Value
The number of total nodes, N	150
The number of referee nodes, R	5
The number of player nodes, P	100
Average transaction size, x	1000B
The block sizes, T^I	2 MB
The percentage of player faulty nodes, f_p	30%
The percentage of referee faulty nodes, f_r	20%
The maximum number of executive rounds for each referee, K	500
The entropy threshold, T_{min}	0.01
Initial probability for players to approve a block, $P_{approve}$	0.8
Initial probability for players to disapprove a block, $P_{disapprove}$	0.2
The optimal predefined ratio for referees to approve a block, $\lambda^*_{approve}$	0.9
The optimal predefined ratio for referees to disapprove a block, $\lambda^*_{disapprove}$	0.1
The unimodal performance criterion for each referee, $G(\lambda)$	$0.9e^{-\left[\frac{(\lambda^*-\lambda_i)^2}{0.0625}\right]}$
The reward learning parameter of $L_{R\text{-}I}$'s player node, a	0.15
The average rate of invalid blocks	0.1

often uncertain which block is correct, the success percentage measure cannot be used in the real environment. In distributed systems like blockchain, throughput and latency are crucial measures to assess how well consensus algorithms perform [65]. A blockchain's throughput is measured by the number of transactions it processes per second (TPS). It can be determined by the number of confirmed transactions during a particular test period and then calculated for a second. The average latency is computed as the average time between the creation of a transaction and its confirmation, while the average block time is defined as the average time it takes to add a new block to a blockchain. The average number of communications is the average number of non-null messages transmitted by each node to validate each block. The average reward is the average reward that each node in a consensus process gets to verify a block. We report some of these metrics according to nodes' roles in the results.

6.2 Experiment 1

In this experiment, we evaluate the efficiency and convergence performance of our proposed schema using the evaluation metrics discussed in the previous section, such as throughput, entropy, and the average performance function. Figure 4 depicts the average entropy and G_{avg} of the proposed schema versus the iteration number in a selected random block's execution CGG step. Entropy values are high in the beginning rounds and gradually decrease, indicating a decreasing rate of change, as seen in Fig. 4. Finally, the change rate of the suggested models' learning automata's chosen actions is very close to zero. G_{avg} gradually approaches its highest value, as seen in Fig. 4. Figure 5 depicts the total average entropy and G_{avg} versus block production episode. As we can see, the total average entropy for different episodes is nearly identical and around 0.01, and the average G_{avg} for different episodes is nearly identical and about 0.9. As a result, G_{avg} has converged to a desirable value in all episodes of block creation.

Table 3 shows the results of CGG-CP in 1000 episodes of block creation. The average number of communications by referees is much higher than by players, as seen in Table 3. Referees receive much better rewards than players; in fact, the cost that referees incur to engage in the consensus process is higher, and in return, they receive more benefits to compensate for the higher costs.

6.3 Experiment 2

We assessed the impact of the percentage of faulty nodes on the schema's performance in this experiment. For this purpose, we investigate the stopping, Byzantine, and unavailability failure models in CGG-CP and report the results in Figs. 6, 7, 8, 9 and 10. As can be seen in Fig. 6, even though 80% of the players have a stop failure, the success percentage is 100%, so the CGG-CP protocol is robust against the stop failure model of players. Figure 6 illustrates that when the proportion of faulty nodes increases in the players' stopping failure model, throughput increases and average block time decreases. Also, Fig. 6 shows that the average player and referee reward decrease as the proportion of players with stopping failures increases. The results can be justified as follows: when a player has a stop failure during a round of the CGG-CP, the nearby referees in the round detect the failure and remove the player from the consensus process. Therefore, in the players' stopping failure model, with the increase in the proportion of faulty players, the number of players in the CGG decreases, and its convergence is done faster. As a

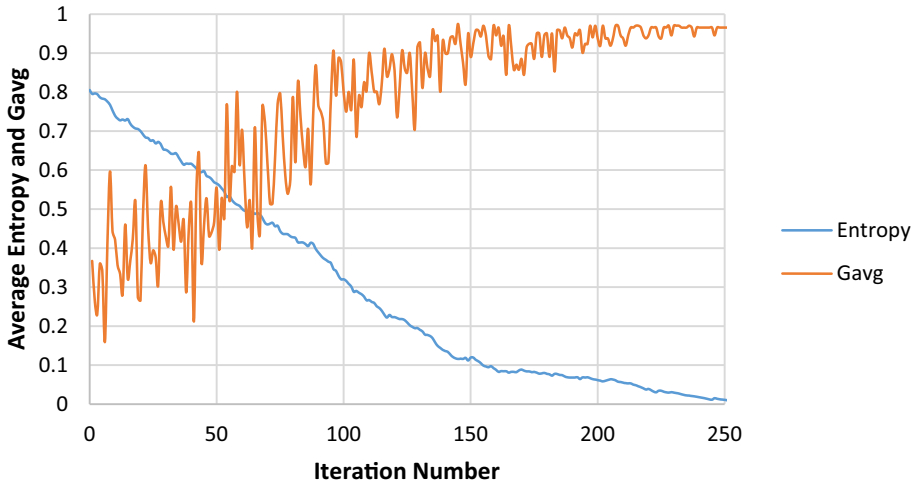
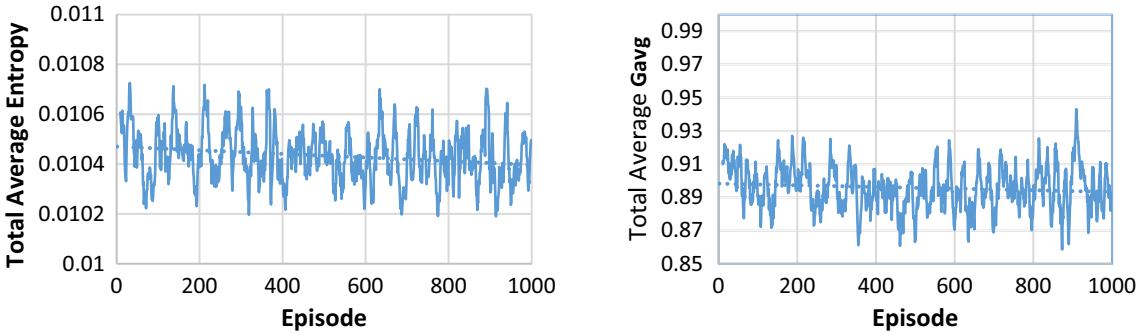


Fig. 4 The average entropy and G_{avg} versus iteration number in a selected random block's execute CGG step in experiment 1



a) The total average entropy. b) The total average G_{avg}.

Fig. 5 The total average entropy and G_{avg} versus episode in experiment 1

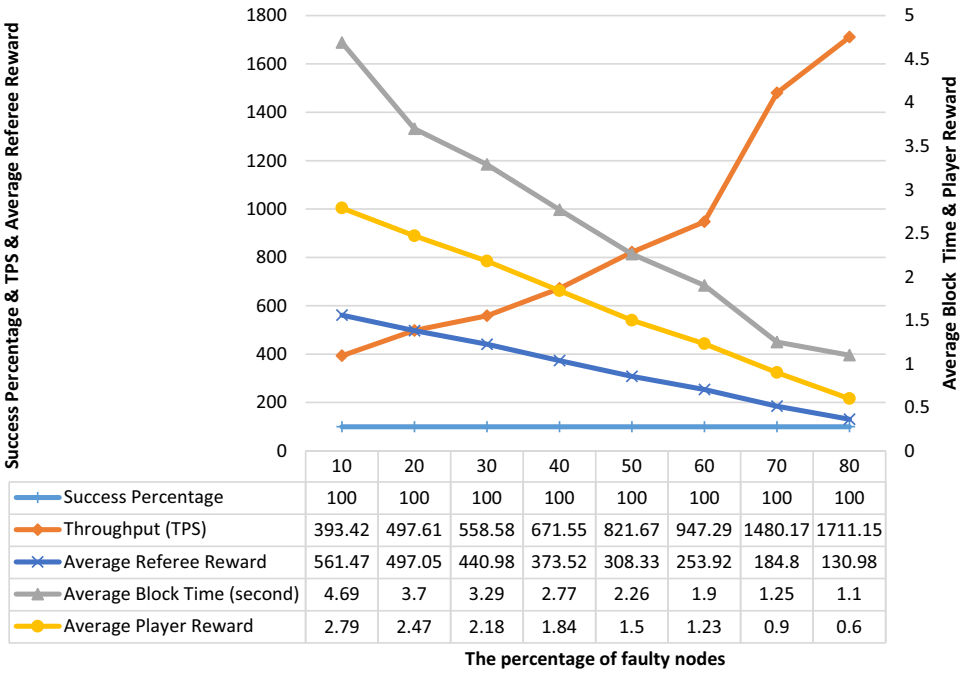
Table 3 The results of CGG-CP in experiment 1

Evaluation Parameter	Value
Success percentage	100%
Throughput (TPS)	585.85
Average latency (seconds)	8.83
Total average G_{avg}	0.961
Total average entropy	0.01
Average number of player communications	200.27
Average number of referee communications	4089.04
Average player reward	2.19
Average referee reward	444.31
Average block time (seconds)	3.38

result, CGG-CP reduces the consensus time and the average reward, which depend on the number of players participating in the consensus.

Figure 7 depicts the effects of the players' Byzantine failure model on CGG-CP. As the percentage of faulty nodes increases, as shown in Fig. 7, success, throughput, and average reward decrease, and the average block time increases. The results can be explained by pointing out that when a byzantine player provides an incorrect vote regarding block approval, its action probability is not adjusted appropriately to maximize the performance criteria of its referees. Therefore, the CGG speed of convergence slows down in these conditions, and as a result, the time to reach a consensus increases. Where over 40 percent of the players are Byzantine, there is a possibility that they

Fig. 6 The impact of the players ' stopping failure model on the CGG-CP in experiment 2



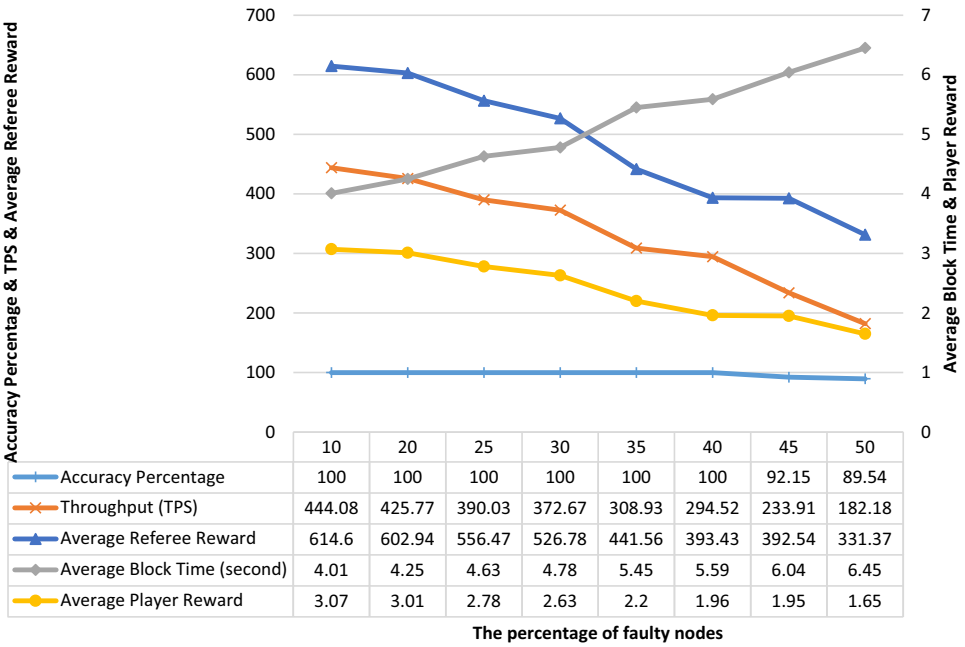


Fig. 7 The impact of the players' Byzantine failure model on the CGG-CP in experiment 2

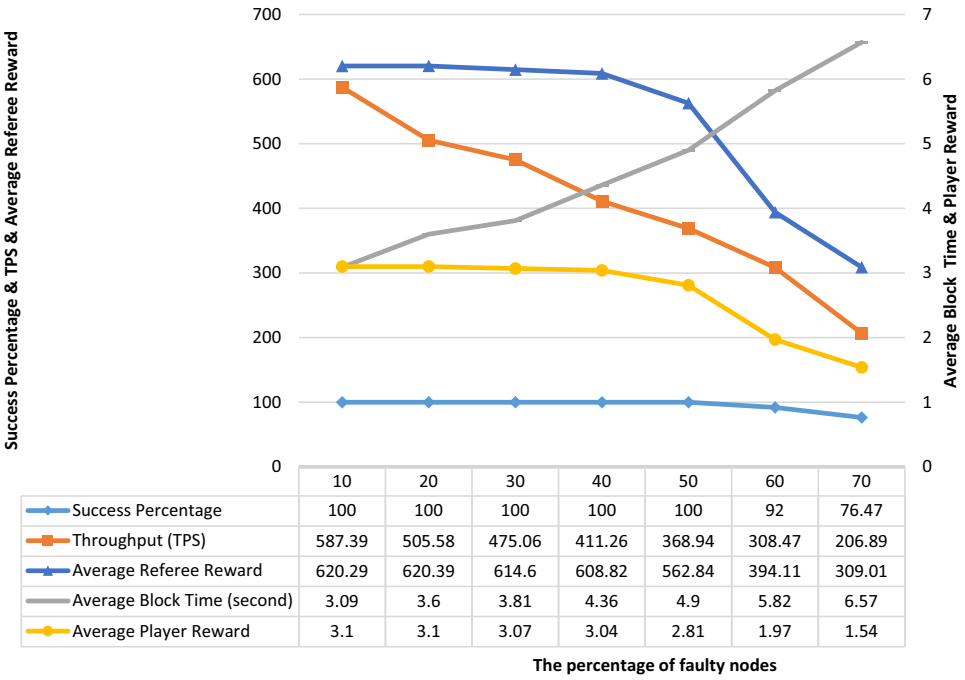
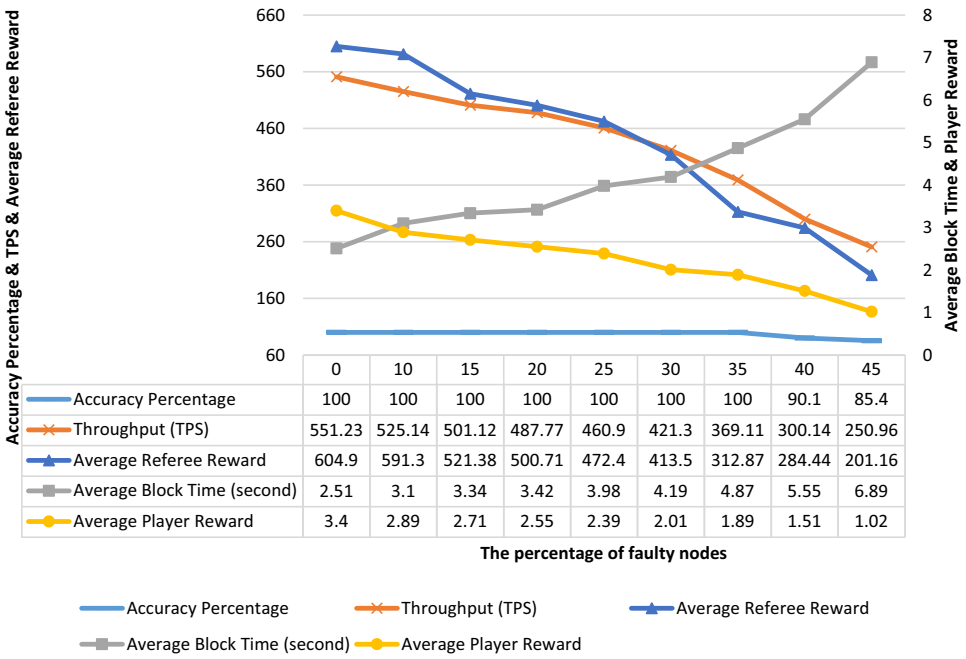


Fig. 8 The impact of the players' unavailability failure model on the CGG-CP in experiment 2

will achieve an incorrect consensus because the referees' criterion functions may not converge to the right value or may not converge at all. As a result, the success percentage in these situations is less than 100%.

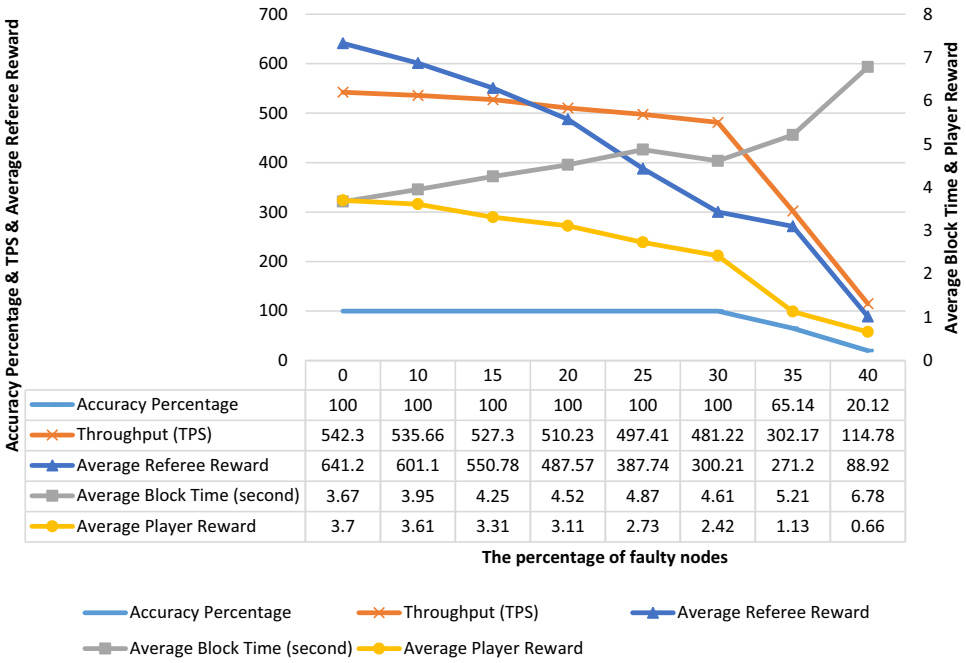
The impacts of the players' unavailability failure model on CGG-CP are shown in Fig. 8. Success percentage, throughput, and average reward all decrease when the proportion of faulty nodes increases, as depicted in Fig. 8, while average block time increases. When the number of players is not available to verify the block and vote on its acceptance or rejection, the action probability vector with an equal probability will be considered as their action probability. As a result, the CGG speed of convergence slows in these settings, and the time to achieve a consensus increases. Even if half of the player nodes are unavailability faulty, in all blocks, the CGG-CP correctly reaches consensus, and its success percentage is 100%.

Figures 9 and 10 show the effects of the referees' failure models on CGG-CP. Figure 9 illustrates that with an increasing number of Byzantine nodes, success,



	0	10	15	20	25	30	35	40	45
Accuracy Percentage	100	100	100	100	100	100	100	90.1	85.4
Throughput (TPS)	551.23	525.14	501.12	487.77	460.9	421.3	369.11	300.14	250.96
Average Referee Reward	604.9	591.3	521.38	500.71	472.4	413.5	312.87	284.44	201.16
Average Block Time (second)	2.51	3.1	3.34	3.42	3.98	4.19	4.87	5.55	6.89
Average Player Reward	3.4	2.89	2.71	2.55	2.39	2.01	1.89	1.51	1.02

Fig. 9 The impact of the referees ' Byzantine failure model on the CGG-CP in experiment 2



	0	10	15	20	25	30	35	40
Accuracy Percentage	100	100	100	100	100	100	65.14	20.12
Throughput (TPS)	542.3	535.66	527.3	510.23	497.41	481.22	302.17	114.78
Average Referee Reward	641.2	601.1	550.78	487.57	387.74	300.21	271.2	88.92
Average Block Time (second)	3.67	3.95	4.25	4.52	4.87	4.61	5.21	6.78
Average Player Reward	3.7	3.61	3.31	3.11	2.73	2.42	1.13	0.66

Fig. 10 The impact of the referees ' stopping failure model on the CGG-CP in experiment 2

throughput, and average reward decrease, while average block time increases. A referee's performance criterion is adjusted based on the referee's decision about block validation in CGG-CP. When a byzantine referee evaluates a block incorrectly, its adjacent non-faulty players will disagree with the referee's opinion. So the non-faulty players' action probability vector value opposes the referee's goal of maximizing performance in the beginning. This results in slower convergence of CGG speed and an increase in consensus time. In this experiment, the results show that the CGG-CP successfully reaches agreement in all blocks if 35% or fewer of the referees are Byzantine nodes. Figure 10 depicts the consequence of the referees' stopping failure model on CGG-CP. As shown in Fig. 10, an increase in stopping faulty referees results in a decrease in success percentage, throughput, and average reward, along with an increase in average block time. If the referee has a stop failure, the referee and the adjacent players will not participate in the consensus process and will not receive a reward in that round. If the number of faulty referees

exceeds 30%, the owner cannot decide to reject or approve some blocks without the presence of this number of referees.

6.4 Experiment 3

In this experiment, we compare the performance of the proposed consensus algorithm with the PBFT and RB-BFT protocols [12] in terms of fault tolerance, throughput, average block time, and communication complexity. The results of this experiment are given in Table 4 and Figs. 11–13. Based on the findings of this experiment, we can conclude the following:

- PBFT's fault tolerance rate is 33%, while RB-BFT's fault tolerance rate is 50%. CGG-CP's fault tolerance rate differs depending on the type of fault and participants' roles in the consensus process. Table 4 demonstrated the results of CGG-CP's fault tolerance based on the node role and the failure model. Referees are more reliable and have more responsibilities in the consensus process than players in CGG-CP, which justifies their lower fault tolerance rate. We can conclude from the obtained results that the fault tolerance rate of CGG-CP improved compared to PBFT. Based on the results and the fact that CGG-CP has much more players than referees, this protocol has more fault tolerance than RB-BFT in the stopping failure model. The CGG-CP improves fault tolerance by automating some consensus tasks using the cognitive engine embedded in the nodes and the mechanisms designed for fault detection.
- The proposed consensus algorithm outperforms the PBFT and RB-BFT in terms of throughput and average block time, as shown in Figs. 11 and 12. Our proposed strategy of grouping nodes and intelligently achieving consensus using the CGG model has reduced the time to reach consensus.
- As seen in Fig. 11, when the number of nodes increases, PBFT and RB-BFT algorithms experience a fast decrease in throughput, but the CGG-CP algorithm shows a slow linear decrease. Therefore, the CGG-CP performs better than PBFT and RB-PBFT in term of scalability.
- Fig. 13 illustrates comparative analysis of the communication complexity of CGG-CP with PBFT and RB-BFT. CGG-CP has a higher average number of communications than RB-PBFT. The reason for this is because nodes send too many exchange messages in order to converge CGG and, as a result, reach a consensus.
- When the number of nodes is over 80, the communication complexity of CGG-CP is lower than PBFT, as shown in Fig. 13. This result can be justified as follows: Unlike PBFT, the proposed consensus method doesn't require every consensus node to transmit their vote to all nodes. The CGG-CP employs nodes with the roles of referee, player, and owner as validating nodes for consensus. Nodes are linked in a neighborhood structure according to the CGG network graph. Each referee and player communicates with their neighbors and the owner.

Table 4 The Results of CGG-CP's fault tolerance based on the node role and the failure model in experiment 3

Node role	Failure model	Fault-tolerance rate (%)
Player	Stopping	80
	Unavailability	55
	Byzantine	40
Referee	Stopping	33
	Unavailability	50
	Byzantine	35

6.5 Experiment 4

This experiment examines the impact of the number of players and referees on the performance of the proposed schema. Figure 14 presents the throughput and average block time results for the CGG-CP with different node numbers, indicating that the ratio of referees to players is about 1/20. We can observe that increasing the number of nodes results in a little reduced transactional throughput and a slightly higher average block time for the proposed method. As the number of nodes increases, the suggested model contains more learning automata and requires more coordination between nodes, reducing the efficiency of the proposed method slightly.

Figure 15 shows the throughput versus the number of referees and players. As we can see from Fig. 15a, by increasing the number of players in the consensus mechanism, the throughput decreases and the average block time increases because of the decrease in the convergence rate of the CGG. We observe from Fig. 15b that the throughput of the CGG-CP increases with the increase of the number of referees and fixed number of players. This result justifies the fact that increasing the referees with fixed players causes a decrease in the degree of the referees' vertices and, as a result, improves the speed of convergence of the CGG algorithm.

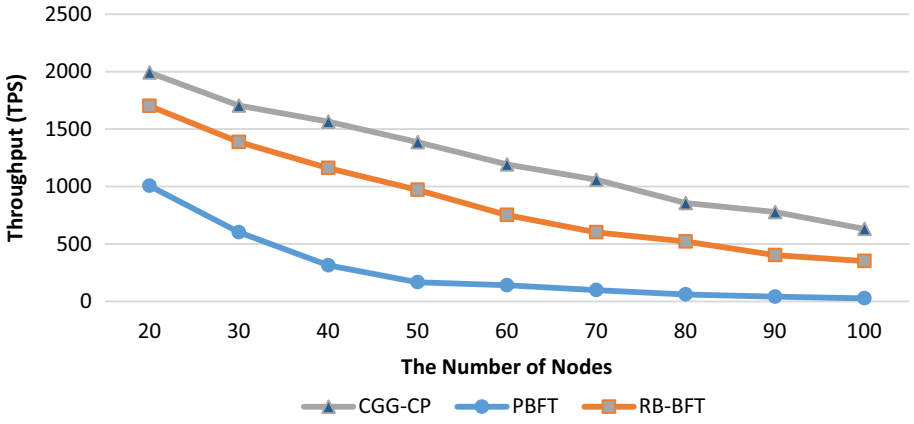


Fig. 11 Comparative analysis of throughput of CGG-CP with PBFT and RB-BFT

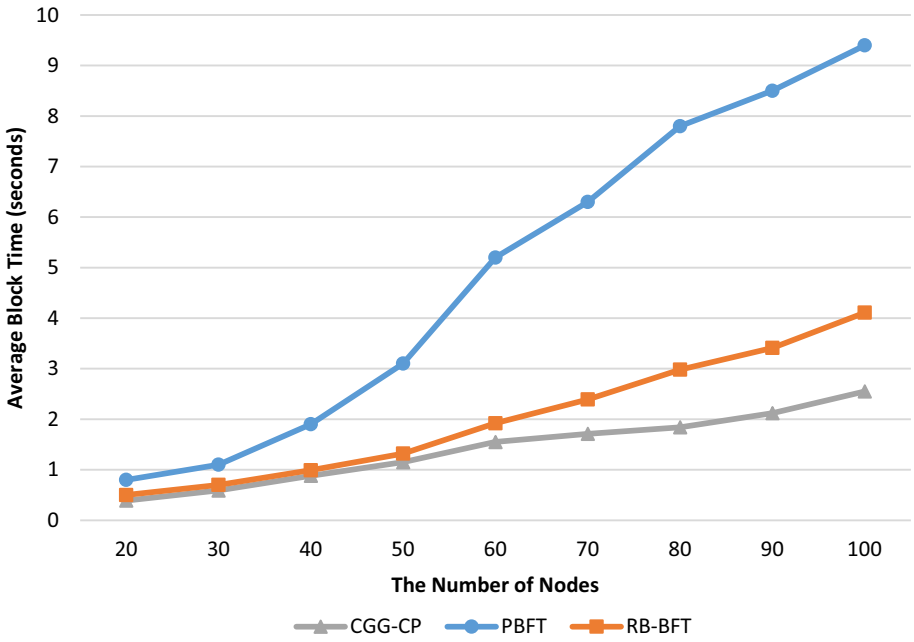


Fig. 12 Comparative analysis of average block time of CGG-CP with PBFT and RB-BFT

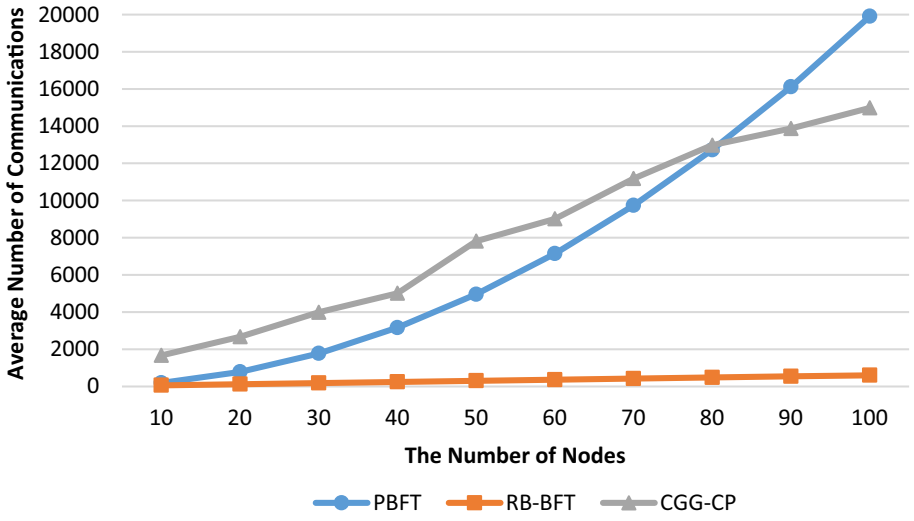
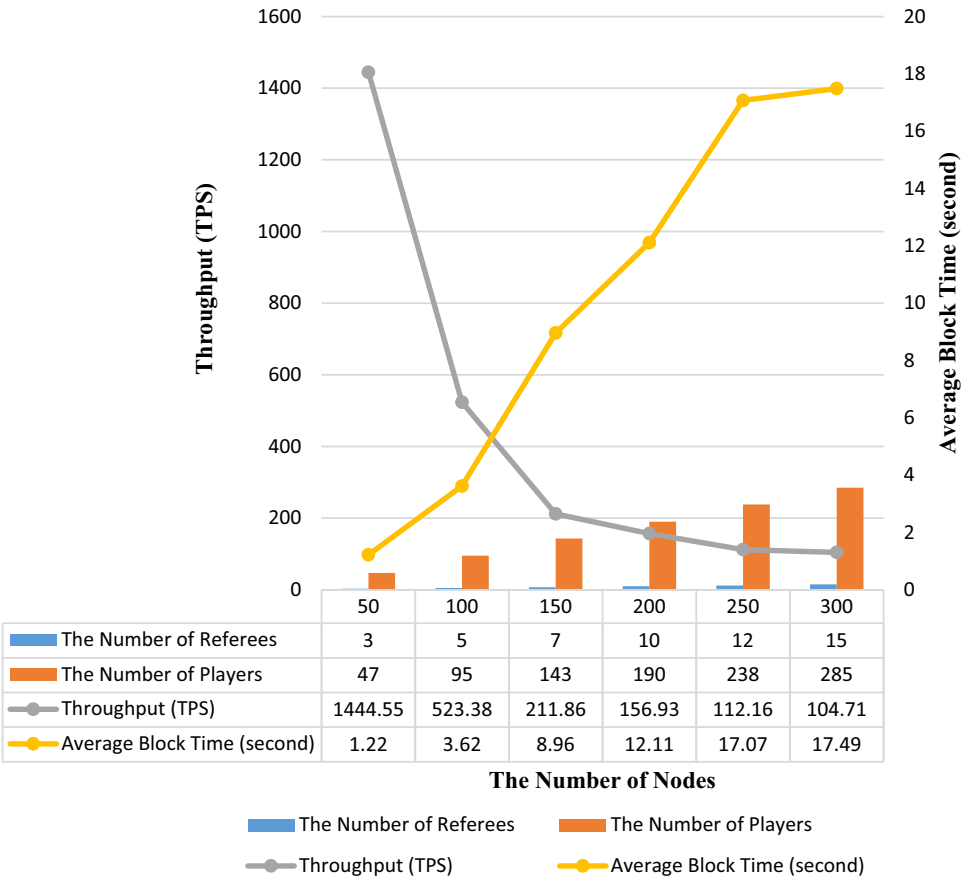
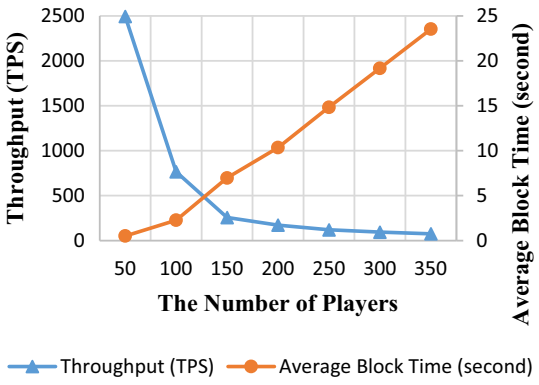


Fig. 13 Comparative analysis of the communication complexity of CGG-CP with PBFT and RB-BFT

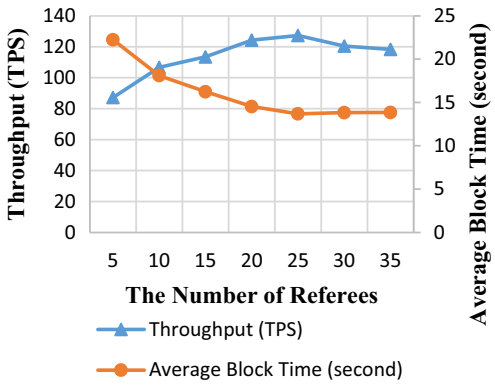


	50	100	150	200	250	300
The Number of Referees	3	5	7	10	12	15
The Number of Players	47	95	143	190	238	285
Throughput (TPS)	1444.55	523.38	211.86	156.93	112.16	104.71
Average Block Time (second)	1.22	3.62	8.96	12.11	17.07	17.49

Fig. 14 The effect of the number nodes on schema's performance in experiment 4



a) Throughput and average block time versus the number of players and 20 referees.



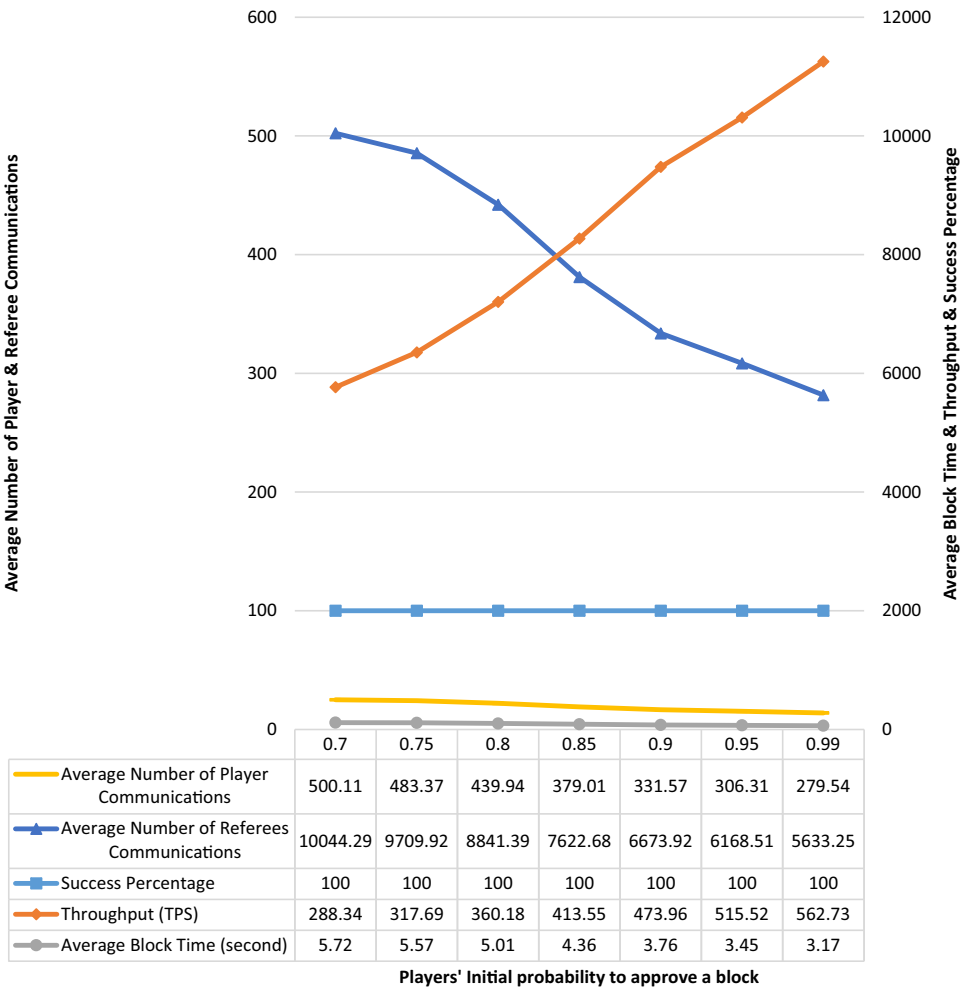
b) Throughput and average block time versus the number of referees and 250 players.

Fig. 15 The effect of the number players and referees on schema's performance in experiment 4

6.6 Experiment 5

This experiment investigates the effects of various parameters on the performance of the CGG-CP, such as the players' initial probability, the entropy threshold, and the reward learning parameter of the $L_{R\text{-}I}$-player's node. Based on the results of this experiment with 1000 blocks, shown in Figs. 16 through 18, we can say the following:

- Figure 16 demonstrates the effects of the players' initial probability to approve a block on CGG-CP. As the initial probability increases, as illustrated in Fig. 16, the average block time and the average number of communications decrease while throughput increases. Furthermore, the success percentage for all initial probability values evaluated in this experiment is 100%.



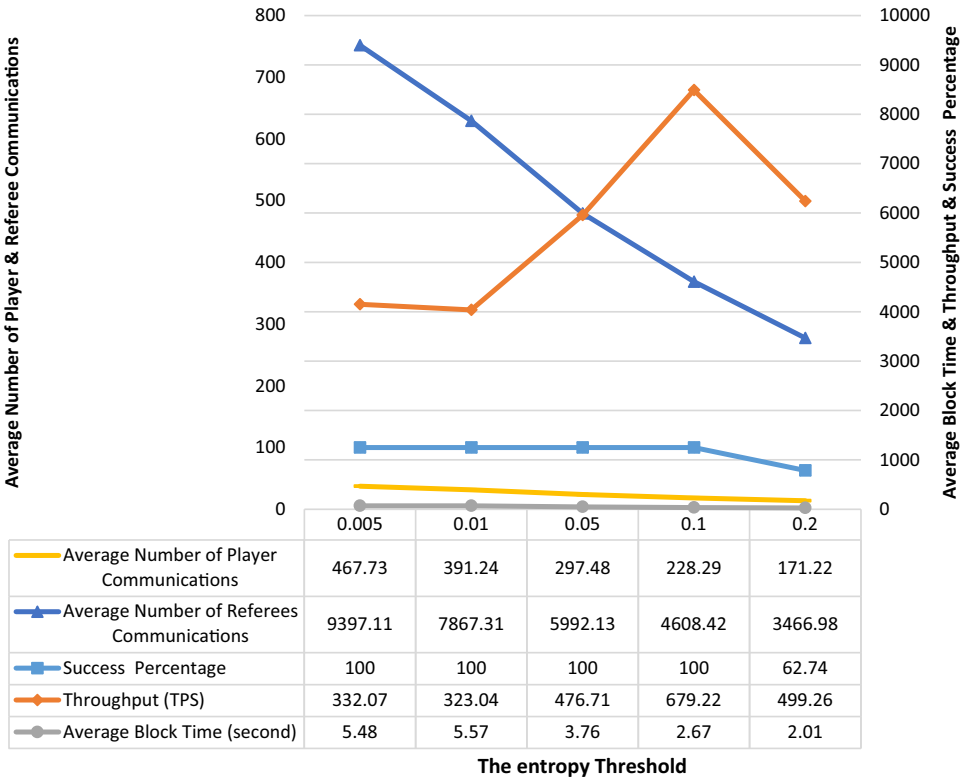
	0.7	0.75	0.8	0.85	0.9	0.95	0.99
Average Number of Player Communications	500.11	483.37	439.94	379.01	331.57	306.31	279.54
Average Number of Referees Communications	10044.29	9709.92	8841.39	7622.68	6673.92	6168.51	5633.25
Success Percentage	100	100	100	100	100	100	100
Throughput (TPS)	288.34	317.69	360.18	413.55	473.96	515.52	562.73
Average Block Time (second)	5.72	5.57	5.01	4.36	3.76	3.45	3.17

Fig. 16 The effect of the players' initial probability on CGG-CP's performance in experiment 5

- Figure 17 shows that as the entropy threshold in the CGG-CP increases, the average block time and the number of communications decrease. Figure 17 further indicates that the throughput increases before reaching 0.2 as the entropy threshold increases, and the success percentage is 100%. With these settings and test conditions, the success percentage of the proposed scheme goes down when the entropy threshold is set to a value greater than 0.1.
- Figure 18 demonstrates that as the incentive learning parameter of the L_{RI}-players node in the CGG-CP increases, the average block time and the number of communications fall while throughput increases. Figure 18 further shows that as the incentive learning parameter grows before reaching 0.25, the success percentage is 100%. When the reward learning parameter is adjusted to a value greater than 0.2 with these settings and the test environment, the success percentage of the proposed scheme decreases.

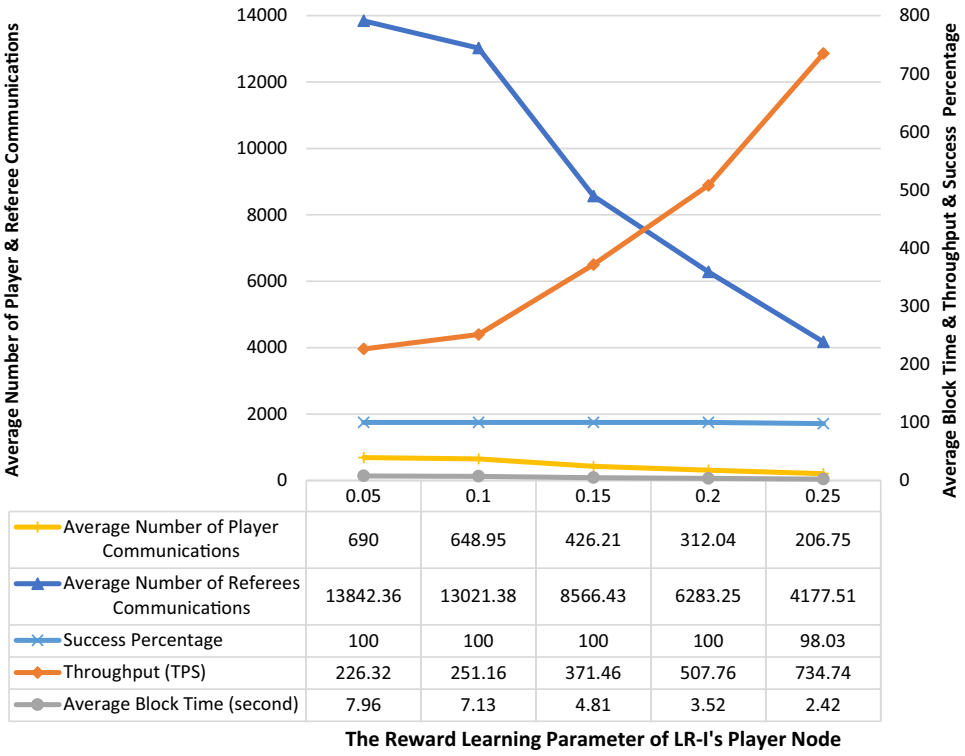
7 Conclusions

This paper proposes a novel consensus algorithm that can be used as intelligent consensus in the cognitive blockchain. Every cognitive peer in the blockchain network can agree on the distributed ledger's current state using the proposed consensus protocol's CGG-based approach in a partially synchronous timing model. In the CGG-CP, nodes with more credibility have more power over whether a block confirms or disproves something. The CGG-CP block verification process is explained as a distributed optimization problem that will be solved with the assistance of the CGG model. After a new block is published, the CGG must decide whether it is valid. Then, the result of the CGG is used to confirm or reject the block based on a policy that has already been set. The efficiency of the proposed algorithm depends on the convergence speed of the CGG model used in the presented consensus approach. We introduced strategies to accelerate the CGG's convergence, such as adjusting the action probabilities of the initial participants. The CGG-CP employs cognitive



	0.005	0.01	0.05	0.1	0.2
Average Number of Player Communications	467.73	391.24	297.48	228.29	171.22
Average Number of Referees Communications	9397.11	7867.31	5992.13	4608.42	3466.98
Success Percentage	100	100	100	100	62.74
Throughput (TPS)	332.07	323.04	476.71	679.22	499.26
Average Block Time (second)	5.48	5.57	3.76	2.67	2.01

Fig. 17 The effect of the entropy threshold on CGG-CP's performance in experiment 5



	0.05	0.1	0.15	0.2	0.25
Average Number of Player Communications	690	648.95	426.21	312.04	206.75
Average Number of Referees Communications	13842.36	13021.38	8566.43	6283.25	4177.51
Success Percentage	100	100	100	100	98.03
Throughput (TPS)	226.32	251.16	371.46	507.76	734.74
Average Block Time (second)	7.96	7.13	4.81	3.52	2.42

Fig. 18 The effect of the reward learning parameter of $L_{R\text{-}I}$'s player node on CGG-CP 's performance in experiment 5

engines to automate tasks related to consensus. This results in reducing the cost of participation and improving security by limiting message manipulation. The CGG-CP ensures high network scalability, allows numerous validation nodes to achieve consortium blockchain consensus in a decentralized manner, and reduces the influence of failure nodes

while maintaining fast transaction speed. We thoroughly examine the CGG-CP's tolerance, safety, liveliness, participation cost, and complexity. To show the superiority of the CGG-CP, we assess it through several simulations in the blockchain simulator and compare it with the existing schemes. According to experimental data, the proposed consensus could compete with existing protocols in terms of consensus performance metrics.

In the future, we aim to develop new solutions to accelerate the convergence of the CGG and, thereby, reduce the communication complexity of the proposed consensus algorithm. We also plan to theoretically prove the consensus mechanism's effectiveness in terms of fault tolerance, safety, and liveness. Furthermore, we will intend to apply other artificial intelligence methods, such as outlier analysis and clustering, as knowledge discovery techniques in the cognitive engines of the proposed consensus protocol.

Author contributions R.Ameri and M.Meybodi contributed to the design of the research, to the analysis of the results and to review the manuscript. R.Ameri designed, implemented and performed the experiments of the research.

Funding The authors have not disclosed any funding.

Data availability Enquiries about data availability should be directed to the authors.

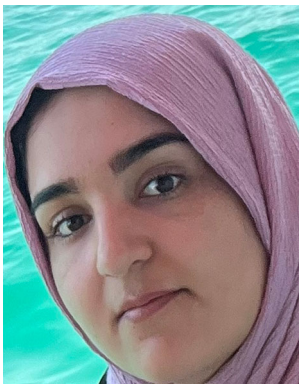
Declarations

Conflict of interest The authors have not disclosed any competing interests.

References

1. Hu, X., Song, X., Cheng, G., Wu, H., Gong, J.: Efficient sharing of privacy-preserving sensing data on consortium blockchain via group key agreement. Comput. Commun. **194**, 44–54 (2022)
2. Wang, X., et al.: Capacity analysis of public blockchain. Comput. Commun. **177**, 112–124 (2021)
3. Zarrin, J., Wen Phang, H., Babu Saheer, L., Zarrin, B.: Blockchain for decentralization of internet: prospects, trends, and challenges. Cluster Comput. **24**(4), 2841–2866 (2021)
4. Lamport, L.: The weak Byzantine generals problem. J. ACM **30**(3), 668–676 (1983)
5. Ballandies, M.C., Dapp, M.M., Pournaras, E.: Decrypting distributed ledger design—taxonomy, classification and blockchain community evaluation. Cluster Comput. **25**(3), 1817–1838 (2022)
6. Liu, Y., Yu, F.R., Li, X., Ji, H., Leung, V.C.M.: Blockchain and machine learning for communications and networking systems. IEEE Commun. Surv. Tutorials **22**(2), 1392–1431 (2020). https://doi.org/10.1109/COMST.2020.2975911
7. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: "An overview of blockchain technology: Architecture, consensus, and future trends", In. IEEE Int.Congress on Big Data (BigData congress) **2017**, 557–564 (2017)
8. Yu, F.R., Liu, J., He, Y., Si, P., Zhang, Y.: Virtualization for distributed ledger technology (vDLT). IEEE Access **6**, 25019–25028 (2018)
9. Gadekallu T. R., et al., Blockchain for the metaverse: A review, arXiv Prepr. arXiv2203.09738 (2022)
10. Ameri, R., Meybodi, M.: "Cognitive Blockchain and Its Application to Performance Optimization in Blockchain Systems," *Technical report of the Amirkabir University* (2022)
11. Ameri, R., Meybodi, M.R., Daliri Khomami, M.M.: Cellular Goore Game and its application to quality-of-service control in wireless sensor networks. J. Supercomput. **78**(13), 1–48 (2022)
12. He, F., Feng, W., Zhang, Y., Liu, J.: An improved byzantine fault-tolerant algorithm based on reputation model. Electronics **12**(9), 2049 (2023)
13. Nguyen, G.T., Kim, K.: A survey about consensus algorithms used in blockchain. J. Inf. Process. Syst. **14**(1), 101–128 (2018). https://doi.org/10.3745/JIPS.01.0024
14. Coulouris, G., Dollimore, J., Kindberg, T.: Distributed Systems: Concepts and Design Edition 3. Addison-Wesley, Boston (2001)
15. C. Cachin, "Blockchains and consensus protocols: Snake oil warning," In: 2017 13th European Dependable Computing Conference (EDCC), 1–2 (2017)
16. Thathachar, M.A.L., Sastry, P.S.: Networks of learning automata: techniques for online stochastic optimization. Springer, Boston (2004)
17. Akbari Torkestani, J.: An adaptive learning to rank algorithm: learning automata approach. Decis. Support Syst. **54**(1), 574–583 (2012). https://doi.org/10.1016/j.dss.2012.08.005
18. Lee, B.H., Lee, K.Y.: Application of S-model learning automata for multi-objective optimal operation of power systems. IEE Proc.-Gen. Transm. Distrib. **152**(2), 295–300 (2005)
19. Tsetlin, M.L.: Automaton theory and modeling of biological systems. Academic Press, New York (1973)
20. Narendra, K., Thathachar, M.: Learning automata: an introduction. Courier corporation **32**(6), (2012)
21. Thathachar, M.A.L., Arvind, M.T.: Solution of Goore game using modules of stochastic learning automata. J. Indian Inst. Sci. **77**(1), 47–61 (1997)
22. Cao, Y.U., Kahng, A.B., Fukunaga, A.S.: Cooperative mobile robotics: antecedents and directions. In: Arkin, R.C., Bekey, G.A. (eds.) Robot colonies. Springer, Boston (1997)
23. Chen, D., Varshney, P.K.: QoS support in wireless sensor networks: a survey. Int. Conf. on Wireless Netw. **233**, 1–7 (2004)
24. Rezvanian, A., Saghiri, A.M., Vahidipour, S.M., Esnaashari, M., Meybodi, M.R.: Recent advances in learning automata. Stud. Comput. Intell. **754**, 1–458 (2018). https://doi.org/10.1007/978-3-319-72428-7
25. Norman, M.F.: On the linear model with two absorbing barriers. J. Math. Psychol. **5**(2), 225–241 (1968). https://doi.org/10.1016/0022-2496(68)90073-4
26. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized Bus. Rev. **8**, 21260 (2008)
27. Zhang, L., et al.: BDSS: blockchain-based data sharing scheme with fine-grained access control and permission revocation in medical environment. KSII Trans. Int. Inf. Syst. (2022). https://doi.org/10.3837/tiis.2022.05.012
28. Salah, K., Rehman, M.H.U., Nizamuddin, N., Al-Fuqaha, A.: Blockchain for AI: review and open research challenges. IEEE Access **7**, 10127–10149 (2019). https://doi.org/10.1109/ACCESS.2018.2890507
29. Begicheva A., Kofman A., "Fair proof of stake. technical report." Fair block delay distribution in proof-of-stake project (2018)
30. Larimer, D.: Delegated proof-of-stake (dpos). Bitshare whitepaper **81**, 85 (2014)

31. Ren, L.: "Proof of stake velocity: Building the social currency of the digital age". Self-published white Paper. [Online]. https://coss.io/documents/whitepapers/reddcoin.pdf (2014)
32. "NEM Blockchain's NIS1 - The Developer's Sandbox." https://nemplatform.com/ (2021) Accessed 31 August 31
33. Karantias, K., Kiayias, A., Zindros, D.: Proof-of-burn. In: Bonneau, J., Heninger, N. (eds.) International conference on financial cryptography and data security. Springer, Cham (2020)
34. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of space. Lect. Notes in Comput. Sci. **9216**, 585–605 (2015). https://doi.org/10.1007/978-3-662-48000-7_29
35. Hyperledger, "PoET 1.0 Specification — Sawtooth v1.0.5 documentation," sawtooth.hyperledger.org. https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html Accessed 31 August 2021
36. Sabt, M., Achemlal, M., Bouabdallah, A.: Trusted execution environment: what it is, and what it is not. IEEE Trustcom/BigDataSE/ISPA **1**, 57–64 (2015)
37. Castro, M., Liskov, B.: Practical byzantine fault tolerance. OSDI **1999**(99), 173–186 (1999)
38. Hyperledger, "Hyperledger – Open Source Blockchain Technologies," *Hyperledger*, 2019. https://www.hyperledger.org/ Accessed 22 November 2021
39. Zheng, Z., Xie, S., Dai, H.-N., Chen, X., Wang, H.: Blockchain challenges and opportunities: a survey. Int. J. Web Grid Serv. **14**(4), 352–375 (2018)
40. Muratov F., Lebedev A., Iushkevich A., Nasrulin B., Takemiya M., "YAC: BFT consensus algorithm for blockchain," arXiv Prepr. arXiv1809.00554 (2018)
41. D. Schwartz, N. Youngs, and A. Britto, "The Ripple protocol consensus algorithm," *Ripple Labs Inc White Pap.*, vol. 5, no. 8, pp. 1–8, 2014, [Online]. http://www.naation.com/ripple-consensus-whitepaper.pdf
42. Stellar, "Stellar consensus protocol - Stellar developers," *URL:* https://www.stellar.org/developers/guides/concepts/scp.html, 2016. https://www.stellar.org/developers/guides/concepts/scp.html Accessed 31 August 2021
43. J. Kwon, "TenderMint: Consensus without Mining," *the-Blockchain.Com*, vol. 6, pp. 1–10, 2014, [Online]. tendermint.com/docs/tendermint.pdf
44. Qin, H., Cheng, Y., Ma, X., Li, F., Abawajy, J.: Weighted byzantine fault tolerance consensus algorithm for enhancing consortium blockchain efficiency and security. J. King Saud Univ. Inf. Sci. **34**(10), 8370–8379 (2022)
45. Tang, S., Wang, Z., Jiang, J., Ge, S., Tan, G.: Improved PBFT algorithm for high-frequency trading scenarios of alliance blockchain. Sci. Rep. **12**(1), 1–12 (2022)
46. M. Salimitari, M. Joneidi, and M. Chatterjee, "AI-enabled blockchain: An outlier-aware consensus protocol for blockchain-based iot networks," In: 2019 IEEE Global Communications Conference, GLOBECOM 2019—Proceedings, 1–6, https://doi.org/10.1109/GLOBECOM38437.2019.9013824. (2019)
47. Liu, X., Liu, Y., Li, X., Cao, H., Wang, Y.: FP-BFT: a fast pipeline byzantine consensus algorithm. IET Blockchain (2023). https://doi.org/10.1049/blc2.12030
48. G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, "Narwhal and tusk: a dag-based mempool and efficient bft consensus," In Proceedings of the Seventeenth European Conference on Computer Systems. 34–50 (2022)
49. Bugday, A., Ozsoy, A., Öztaner, S.M., Sever, H.: Creating consensus group using online learning based reputation in blockchain networks. Pervasive Mob. Comput. **59**, 101056 (2019). https://doi.org/10.1016/j.pmcj.2019.101056
50. M. Liu, Y. Teng, F. R. Yu, V. C. M. Leung, and M. Song, "Deep Reinforcement Learning Based Performance Optimization in Blockchain-Enabled Internet of Vehicle," In: IEEE International Conference on Communications, 2019-May. 1–6 https://doi.org/10.1109/ICC.2019.8761206 (2019)
51. Liu, M., Yu, F.R., Teng, Y., Leung, V.C.M., Song, M.: Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: a deep reinforcement learning approach. IEEE Trans. Ind. Informatics **15**(6), 3559–3570 (2019). https://doi.org/10.1109/TII.2019.2897805
52. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002). https://doi.org/10.1145/571637.571640
53. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: speculative Byzantine fault tolerance. ACM Trans. Comput. Systems **27**(4), 45–58 (2009). https://doi.org/10.1145/1658357.1658358
54. R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 BFT protocols," In Proceedings of the 5th European conference on Computer systems, 363–376 (2010)
55. Chen, P., Han, D., Weng, T.-H., Li, K.-C., Castiglione, A.: A novel Byzantine fault tolerance consensus for Green IoT with intelligence based on reinforcement. J. Inf. Secur. Appl. **59**, 102821 (2021)
56. Riahi, K., Abouaissa, A., Idoumghar, L.: "A Reinforcement Learning-Based Node Selection for PBFT Consensus." Ninth Int. Conf. Software Defined Syst. (SDS) **2022**, 1–3 (2022)
57. Goh, Y., Yun, J., Jung, D., Chung, J.-M.: Secure trust-based delegated consensus for blockchain frameworks using deep reinforcement learning. IEEE Access **10**, 118498–118511 (2022)
58. Yun, J., Goh, Y., Chung, J.-M.: DQN-based optimization framework for secure sharded blockchain systems. IEEE Int. Things J. **8**(2), 708–722 (2020)
59. King, S., Nadal, S.: "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," self-published paper. August, 19(1) [Online]. https://peercoin.net/assets/paper/peercoin-paper.pdf (2012)
60. "NEM XEM whitepapers - whitepaper.io." https://whitepaper.io/document/583/nem-whitepaper Accessed 08 December 2022
61. P4Titan, "Slimcoin. A Peer-to-Peer Crypto-Currency with Proof-of-Burn 'Mining without Powerful Hardware,'" http://Www.Slimcoin.Org/ 2014, [Online]. www.slimcoin.org (2014)
62. Park S., Kwon A., Fuchsbauer G., Gaži P., Alwen J., Pietrzak K., "Spacemint: A cryptocurrency based on proofs of space," In Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22, 2018, 480–499.
63. Salomaa, A.: Public-key cryptography. Springer, Berlin (1996)
64. Wang, W., et al.: BSIF: Blockchain-based secure, interactive, and fair mobile crowdsensing. IEEE J. Sel. Areas Commun. **40**(12), 3452–3469 (2022)
65. Mohsenzadeh, A., Bidgoly, A.J., Farjami, Y.: A novel reputation-based consensus framework (RCF) in distributed ledger technology. Comput. Commun. **190**, 126–144 (2022)



Reyhaneh Ameri received a B.Sc. degree in Computer Engineering from Iran University of Science & Technology in 2014 and an M.Sc. degree in Computer Engineering from the Sharif University of technology in 2016. She is currently a Ph.D. candidate in Computer Engineering, working under Professor Mohammad Reza Meybodi. Her research interests include blockchain, artificial Intelligence, Learning automata, Machine Learning, and software development.

Mohammad Reza Meybodi received the B.Sc. and M.Sc. degrees in economics from Shahid Beheshti University, Tehran, Iran, in 1973 and 1977, respectively, the M.Sc. and Ph.D. degrees in computer science from Oklahoma University, Norman, OK, USA, in 1980 and 1983, respectively. He was an Assistant Professor with Western Michigan University, Kalamazoo, MI, USA, from 1983 to 1985, and an Associate Professor with Ohio University, Athens, OH, USA, from 1985 to 1991. He is currently a Full Professor with the Computer Engineering Department, Amirkabir University of Technology, Tehran. His research interests include wireless networks, fault tolerant systems, learning systems, parallel algorithms, soft computing, and software development.