The Journal of Supercomputing
https://doi.org/10.1007/s11227-022-04435-1

Cellular Goore Game and its application to quality-of-service control in wireless sensor networks

Reyhaneh Ameri[1] · **Mohammad Reza Meybodi**[1] · **Mohammad Mehdi Daliri Khomami**[1]

Accepted: 7 March 2022

Abstract

The Goore Game (GG) is a model for collective decision-making under uncertainty, which can be used as a tool for stochastic optimization of a discrete variable function. The Goore Game has a fascinating property that can be resolved in an entirely distributed manner with no intercommunication between the players. In this paper, we introduce a new model called Cellular Goore Game (CGG). CGG is a network of Goore Games in which, at any time, every node (or node in a subset of the nodes) in the network plays the role of a referee that participates in a GG with its neighboring players (voters). Like GG, each player independently selects its optimal action between two available actions based on their gains and losses received from its adjacent referees. Players in CGG know nothing about how other players are playing or even how/why they are rewarded/penalized by the voters. CGG may be used for modeling systems that can be described as massive collections of simple objects interacting locally with each other. Through simulations, the behavior of CGG for different networks of players/voters is studied. This paper presents a novel CGG-based approach to efficiently solve the Quality-of-Service (QoS) control for clustered WSNs to show the potential of CGG. Also, a CGG-based QoS control algorithm for WSNs with multiple sinks is proposed that dynamically adjusts the number of active sensors during WSN operation. Several experiments have been conducted to evaluate the performance of these algorithms. The obtained results show that the proposed CGG-based algorithms are superior to the existing algorithms in terms of the QoS control performance metrics.

Keywords Goore Game · Learning automata · Cellular Goore Game · Wireless sensor networks

✉ Mohammad Reza Meybodi
mmeybodi@aut.ac.ir

Extended author information available on the last page of the article

Published online: 14 April 2022

1 Introduction

One of the interesting games studied in artificial games is the Goore Game (also referred to as the Gur game in the related literature). The Goore Game, formulated by Tsetlin [1], is a simple symmetric game among a set of players and a referee. It is a particular form of a cooperative game in which each player has two possible choices to vote corresponding to yes or no [2]. The referee evaluates the selected vote by players. The referee has a unimodal performance evaluation function G. At each instance, all players select one of their choices. The referee computes the ratio λ, the number of players voted "yes" divided by the total number of players voted. Then, the referee awards a dollar with probability G(λ) and assesses a dollar with probability 1—G(λ) to every player independently. Based on their individual gains and losses, the players then decide, again independently, how to cast their votes on the next iteration. After enough iterations, the number of players that will say yes is correlated with the maximum of G(λ). In most general situations, a player may not even be aware of either the existence of the other players or the number of players involved. Each player needs to know only the outcome of selecting his action.

The Goore Game has a fascinating property that can be resolved in an entirely distributed manner with no intercommunication between the players. The theory of learning automata has been used to develop GG. Learning automata (LAs) are models for adaptive decision making in unknown random environments [3]. A LA has a finite set of actions. Its goal is to learn the optimal action through repeated interaction with the environment. An optimal action is an action with the highest probability of getting reward from the environment. GG with LA is a GG which uses learning automata as the learning paradigm to model players with the aim of maximizing G(λ). GG has a single referee whose goal is to maximize the referee's performance criterion function, so this game can only optimize one criterion and limit GG applications. Also, a referee makes the final decision in GG and can be a single point of failure in GG.

GG has found crucial applications in the context of quality-of-service support in wireless sensor networks [4]. A WSN includes a set of sensor nodes capable of probing the environment and transferring the information collected to the base station (sink) or cluster head. An essential issue in WSNs is energy efficiency because the consumable energy is limited, and replacing or recharging a sensor node's battery is usually impossible. Significant energy is consumed for data transmission in WSNs, and reducing the communication distance is a way to conserve energy. Deploying multiple sinks and communicate through multi-hop communications are effective methods. QoS control also is one of the fundamental mechanisms to guarantee the effective operation of WSNs that usually consumes energy. QoS control in multi-hop clustered and multiple sinks WSNs are issues that are less discussed in research works. GG-based approaches are only applicable to single-hop WSNs with star-topology where sensor nodes communicate with the base station via a single hop. In large WSNs, clusters can ordinarily be effectively organized to apply the GG to each cluster. However, this trick assuredly is not applicable for overlapping clusters.

Most network applications may require considering local interaction like a network GG or multi-criteria optimization. Therefore, we first propose a new game called Cellular Goore Game (CGG) in this paper. CGG may be used as a model for large numbers of simple identical components with local interactions. CGG is a network of GG in which every node (cells) as a referee plays a Goore Game with its adjacent nodes. Each cell in CGG can simultaneously play the role of a player or a referee. CGG aims to maximize the sum of the objective functions in the referee nodes. And then, in order to show the potential of CGG, two algorithms based on CGG for Quality-of-Service (QoS) control in WSNs when clusters have overlap are also designed, one for clustered WSNs and one for WSNs with multiple sinks. Several experiments using network simulator NS3 have been conducted to evaluate the performance of these algorithms. Simulation results have demonstrated the effectiveness of the proposed algorithm to QoS control in WSNs in terms of QoS control performance metrics. Our work proposes a set of novel contributions that are summarized as follows:

1. We propose the Cellular Goore Game (CGG) as a novel model to maximize the sum of the objective functions in the referee nodes. CGG is a network of Goore Games that every node can play the role of referees, each of which participates in a Goore Game with its neighboring players.
2. We provide a rigorous analysis of the behavior of the proposed CGG. In this regard, we define a new criterion, the scaled average performance function.
3. We present a novel CGG-based approach to efficiently solve the Quality-of-Service (QoS) control for clustered WSNs *that c*an also be employed when clusters overlap.
4. We develop a CGG-based QoS control algorithm for WSN with multiple sinks, in which each sensor node can belong to overlapping clusters of several sinks.
5. We provide a comparative analysis of CGG-based QoS control protocols for WSN, emphasizing the QoS metrics using network simulator NS3. Simulation results demonstrate the significant performance improvement of the proposed algorithms in terms of QoS control performance metrics.

The rest of this paper is organized as follows. Learning automata, Goore-Game, Goore-Game with LA, and QoS control for WSNs are briefly introduced in Sect. 2. The proposed Cellular Goore Game is described in Sect. 3. In Sect. 4, several experiments have been conducted to study the behavior of CGG. Section 5 gives an application of CGG to QoS control in WSNs, along with providing simulations performed to evaluate the proposed methods. Section 6 concludes the paper.

2 Background

In this section, to provide background information for the rest of the paper, we give a brief overview of the Goore Game, learning Automata, the Goore Game with LA, and QoS control for Wireless Sensor Networks.

2.1 The Goore Game

This section will introduce the Goore Game in more details. The Goore Game is an example of a self-organization and self-optimization game studied in artificial intelligence. It was presented by Tsetlin [1] and analyzed in detail by Narendra & Thathachar [5] and Thathachar & Arvind [2]. The GG has been described in [5] using the following informal formulation:

> *"Imagine a large room containing N cubicles and a raised platform. One person (voter) sits in each cubicle and a Referee stands on the platform. The Referee conducts a series of voting rounds as follows. On each round the voters vote "Yes" or "No" (the issue is unimportant) simultaneously and independently (they do not see each other) and the Referee counts the fraction, λ, of "Yes" votes. The Referee has a uni-modal performance criterion $G(\lambda)$, which is optimized when the fraction of "Yes" votes is exactly λ^*. The current voting round ends with the Referee awarding a dollar with probability $G(\lambda)$ and assessing a dollar with probability 1-$G(\lambda)$ to every voter independently. On the basis of their individual gains and losses, the voters then decide, again independently, how to cast their votes on the next round."*

Each player plays solely in a greedy fashion, voting each time the way that seems to give the player the best payoff. The player does not attempt to predict the behavior of other players. Instead, each player performs by trial and error and only preferentially repeats those actions that produce the best result for that player. Figure 1 shows the pseudocode for GG's operation.

In the following few paragraphs, we discuss applications of GG reported in the literature. Tung et al. [6] have investigated the GG as a coordinator among mobile robots in the presence of restriction ability to communicate. In this application, the GG is used to make sure that the mobile robots choose their action to maximize the throughput of the overall collection and sorting system. In many specific applications such as battlefield communications, the optimum number of sensors sending information at any given time is a necessary QoS requirement. For such a requirement, some Goore Game-based QoS control approaches have been exploited to dynamically adjust the number of active sensors to the optimal one [7].

Iyer et al. [8] modeled the QoS in sensor networks as the optimum number of sensors required to send information to a base station (or sink node) at any given time. Finding this optimum number has been using GG. A sensor is considered a voter that chooses between transmitting data or remaining idle to preserve energy from GG modeling. Thus, each sensor exerts a GG player's role that either votes "On" or "Off" and acts accordingly. The base station is considered the GG Referee with a unimodal performance function G whose maximum is found. Frolik in [9] presented a new WSN QoS control strategy that maintains QoS under various network constraints in remote, harsh environs. In this upgraded original GG based strategy, the base station individually acquaints each sensor through acknowledgment packets whether the actual QoS is higher than the desired one or not. The sensor utilizes this information through a simple probabilistic automaton to decide whether to stay active or sleep independently. This methodology is not

```
Inputs
K // the maximum number of iterations
P // the set of voters
G(f) // a unimodal performance criterion for the referee
Notations:
k // the iteration counter
f // the ratio "Yes" to all votes
Begin
 k = 0
while k< K do
   For each voter_i in P do
      voter_i votes either "Yes" or "No" independently;
   EndFor
   The referee Counts how many Yes votes there are and sets f;
   For each voter_i in P do
      Generate a random variable R_i;
      IF R_i < G(f)
         The referee rewards a dollar to voter_i;
     ELSE
        The referee punishes the voter_i a dollar;
   EndFor
   k = k + 1;
end while
End
```

Fig. 1 Pseudocode of GG

suitable for high-performance, bandwidth-limited applications such as military applications. The number of regions covered by sensors maximized through GG in [10]. A dynamic clustering algorithm that employs the concept of connected dominating sets was utilized, and the problem was resolved by playing GG among the cluster nodes. They used distributed Ants algorithms and genetic algorithms to optimize the number of active nodes to consider the dynamic nature of WSNs.

Ayers et al. [11] introduce a modified GG algorithm called Gureen Game that improves QoS control and the original GG power consumption weakness for senor networks. The significant new mechanisms of the Gureen Game are player rotation, proactive referee, unambiguous reward/punishment. The sleep states introduced by Gureen Game have four times out parameters, whose values are assigned arbitrarily, making the setting of this parameter unclear. The Gureen Game approach proposes a mechanism of player rotation where, after an arbitrary number of executions, an active node moves to a sleep state. Semprebom et al. in [12] propose a communication approach based on GG called (m,k)-Gur Game for WSN. The (m,k)-Gur Game maintains QoS parameters as determined by the application and preserves the spatial coverage of the network [13]. introduces the QoS control technique to fit WSNs requirements with Quick Convergence is

called Adaptive Periodic Gur Game (APGur). APGur tries to decrease the time required to reach stability and support the wanted number of active sensors. Semprebom et al., also in [14], presented a new adaptive QoS control approach as Skip Game for QoS and energy management in IEEE 802.15.4 networks that dynamically adjusted the number of active nodes. Skip Game aims at finding a trade-off between increasing the network lifetime and concurrently providing the QoS and network coverage properly. This proposal enables nodes to alternate between active or inactive by transmitting messages periodically with specific patterns performed in each node in an autonomous and distributed way, driven by a state machine. Li et al. [7] considered a new framework based on the GG for dealing with the QoS control framework for WSNs. In this framework, the optimum sensor number is not determined in advance but learned by interacting with upper-level applications in a reinforcement manner. Moreover, the Estimator Goore Game (EGG) algorithm is devised to effectively improve the number of active sensors. Table 1 summarizes the literature review on the Goore Game.

2.2 Learning automata

A learning automaton [18] is an adaptive decision-making system that can improve its performance by learning how to choose the optimal action from a set of allowed actions through repeated interactions with the random environment. A learning automaton has a finite set of actions, and each action has a certain probability of getting rewarded by its environment. The aim is to learn to choose the optimal action through repeated interaction with the environment. If the learning algorithm is chosen correctly, then the iterative process of interacting with the environment can be made to result in the selection of the optimal action.

The LAs can be classified into two classes, fixed and variable structure LAs [18]. Variable structure LAs which is used in this paper is represented by sextuple...$P.G.T$, where a set of inputs actions (called response or reinforcement signal), is a set of internal states, a set of outputs, P denotes the state probability vector, G is the output mapping, and T is learning algorithm. The learning algorithm is used to modify the probability vector.

$$p_i(t+1) = p_i(t) + a\big(1 - p_i(t)\big) \tag{1}$$

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

$$p_i(t+1) = (1-b)p_i(t) \tag{2}$$

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

Let $_i$ be the action chosen at step t as a sample realization from distribution $P(t)$. In a linear learning algorithm, equation for updating probability vector $P(t)$ is defined by (1) for a favorable response ($\beta = 1$), and (2) for an unfavorable response ($\beta = 0$).

Table 1 Summary of literature review on the Goore Game

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Ranjit Iyer and Leonard Kleinrock [8]	2003	This paper defines wireless sensor network Quality of Service (QoS) in terms of how many deployed sensors are active and use the Gur Game's mathematical paradigm to dynamically adjust to the optimum number of sensors. The base station communicates QoS information to each of the sensors using a broadcast channel. The sensors independently determine whether to stay active or go to sleep using the probabilistic Gur game automaton	QoS Control For Sensor Networks	Finite State Learning Automata (FSLA)	k_t= the number of nodes voting yes in trial t

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Jeff Frolik [9]	2004	This work presents a new WSN QoS control strategy that maintains QoS under various network constraints in remote, harsh environs. In this upgraded original GG based strategy, the base station individually acquaints each sensor through acknowledgment packets whether the actual QoS is higher than the desired one or not. The sensor utilizes this information through a simple probabilistic automaton to decide whether to stay active or sleep independently. This methodology is not suitable for high-performance, bandwidth-limited applications such as military applications	QoS Control For WSNs	FSLA	$0.2 + 0.8e^{-0.002\left[(k_t-35)^2\right]}$ k_t= the number of nodes voting yes in trial t

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
B. John Oommen, Ole-Christoffer Granmo and Asle Pedersen [15]	2007	This article proposed the first arbitrarily accurate realistic solution, the so-called CGG-AdS, to the GG, which needs only a finite number of LA. The CGG-AdS* can attain an unbounded accuracy for the GG by utilizing at most d LA and by recursively pruning the solution space to support that the retained domain includes the solution to the game with a probability as close to unity as wanted. This paper includes the formal proofs and algorithms, the claims of the respective convergence results	Achieve Unbounded Resolution in Finite Player Goore Games	Variable Structure Learning Automata (VSLA)	$ae^{-\left[\left(\frac{\lambda^*-\lambda}{b}\right)^2\right]}$ λ= the fraction of "Yes" votes λ^* = the optimal fraction of "Yes" votes

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Syed I. Nayer and Hesham H. Ali [10]	2008	The number of regions covered by sensors maximizes through GG in this paper. A dynamic clustering algorithm that employs the concept of connected dominating sets was utilized, and the problem was resolved by playing GG among the cluster nodes. They used distributed Ants algorithms and genetic algorithms to optimize the number of active nodes to consider the dynamic nature of WSNs	Self-Optimizing WSNs	Distributed Ants Algorithm and Genetic Algorithms	$0.2 + 0.8e^{-0.002\left[(X_t-35)^2\right]}$ X_t= (the number of regions covered by active sensors)$^{\alpha}$ / the number of active sensors
Dragos Calitoiu [16]	2009	This paper proposed a novel agent-based algorithm for non-destructive searching when the agent visits the same target many times in an unpredictable environment. The distributed GG model was applied instead of classical collaborative and competitive strategies or individual strategies	Search efficiently for randomly located objects	VSLA	$0.9e^{-\left[\frac{(0.7-x)^2}{0.0625}\right]}$; $\forall x \in [0.1]$

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Megan Ayers and Yao Liang [11]	2011	A modified GG algorithm called Gureen Game is proposed that improves QoS control and the power consumption weakness of the original GG for senor networks. The significant new mechanisms of the Gureen Game are player rotation, proactive referee, unambiguous reward/punishment. The sleep states introduced by Gureen Game have four times out parameters, whose values are assigned arbitrarily, making the setting of this parameter unclear. The Gureen Game approach proposes a mechanism of player rotation where, after an arbitrary number of executions, an active node moves to a sleep state	QoS Control for WSNs	FSLA	$0.2 + 0.8e^{-0.002[(k-K)^2]}$ k = the number of number of the active nodes K = the known optimum number of the active nodes for the desirable QoS level

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Byung-Jun Yoon [17]	2011	This paper proposes a modified variant of the GG algorithm, which addresses two significant GG problems: proper normalization of drug response and proper design of a reward procedure to optimize combinatory drugs. The proposed algorithm analyses how a specific drug's concentration change affects the overall drug response by producing an informed estimate on how the concentration should be updated to improve the drug response	Effective optimization of combinatory drugs	FSLA	The normalized drug response function that measures the desirability of a given drug combination
Thathachar M.A.L. and Arvind M.T [2]	2013	This paper comprehensively analyses the GG with the players using the LA. A parallel algorithm involving a module of learning automata for each player is also introduced to improve the speed performance without decreasing the learning procedure's accuracy	Solution to the GG using VSLA	VSLA	$0.9e^{-\left[(0.3-x)*\frac{(0.3-x)}{0.01}\right]}$; $\forall x \in [0.1]$

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Tiago Semprebom, A.R. Pintoy, Carlos Montez, Francisco Vasquesx [12]	2013	The (m,k)-Gur Game proposed in this paper is an evolution of the GG approach, which targets a trade-off between improving the QoS of the network and, concurrently, increasing the spatial diversity. This approach is based on state machines, regulated by (m, k)-firm transmission patterns in nodes. The parameters m and k are individual for each node and are dynamically and autonomically adjusted by them and enable nodes to alternate at each period between active or inactive	The expected QoS and the spatial coverage diversity in WSNs	FSLA	$20 + 80e^{-(0.2/(N/10))\times(r-R)^2}$; R = the number of the active nodes R = the optimum number of messages desired to be received; N = the number of nodes composing the network;

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Eman M. Elshaheda, Rabie A. Ramadanb, Shahinaz M. Al-tabbakha, H. El-zaheda [13]	2014	The article introduces the QoS control technique to fit WSNs requirements with Quick Convergence is called Adaptive Periodic Gur Game (APGur). APGur tries to decrease the amount of time required to reach stability and support the wanted number of active sensors. In APGur, the GG is reapplied periodically to reduce the unbalance of energy consumption, and also, the idea of unambiguous reward/punishment is used [24] to accelerate the convergence	QoS Control in WSNs	FSLA	$0.2 + 0.8e^{-0.002\left[(k_t - n)^2\right]}$ k_t= the number of packets received at time t; n = the known optimum number of packets that the base station wants;

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Tiago Semprebom, Carlos Montez and Paulo Portugal [14]	2015	This paper proposed a new adaptive QoS control approach as Skip Game for QoS and energy management in IEEE 802.15.4 networks that dynamically adjusted the number of active nodes. Skip Game aims at finding a trade-off between increasing the network lifetime and concurrently providing the QoS and network coverage properly. This proposal enables nodes to alternate between active or inactive by transmitting messages periodically with specific patterns performed in each node in an autonomous and distributed way, driven by a state machine	QoS and Energy Management in WSNs	FSLA	$20 + 80e^{-(0.2/(N/10))\times(r-R)^2}$; R = the optimum number of messages desired to be received; N = the number of nodes composing the network; R = the number of the active nodes

Table 1 (continued)

Authors	Year	Summary of Findings	Application	Type of Learning	Performance Criterion Function
Shenghong Li, Hao Ge, Ying-Chang Liang, Feng Zhao, Jianhua Li [7]	2016	A novel efficient QoS control framework for WSNs in both stationary and non-stationary environments had been proposed that the optimum number of active sensors had been learned by interacting with the upper-level application in a reinforcement manner	QoS Control with incomplete information for WSNs	FSLA	$0.2 + 0.8e^{-0.002\left[\left(k_t - k^*\right)^2\right]}$; k^* = desirednumber; k_t = the number of nodes voting yes in trial t

*Continuous Goore Game with Adaptive dary Search

Two parameters a and b represent reward and penalty parameters, respectively. The parameter a (b) determines the amount of increase (decreases) of the action probabilities. R denotes the number of actions that can be taken by the LA. If $a = b$, the linear learning algorithm is called linear reward penalty (L_{RP}); if $a >> b$ the learning algorithm is called linear reward-ε penalty ($L_{R\varepsilon P}$); and finally, if $b = 0$, it is called linear reward inaction (L_{RI}) algorithm. Learning automata have been recently utilized in applications such as graph problem [19], image processing [20], internet of things [21], cloud computing [22], wireless sensor networks [23]–[25], p2p networks [26] and complex networks [27]–[30].

2.3 The Goore Game with LA

When the Goore Game was first investigated, Tsetlin utilized his so-called Tsetlin automaton to solve it. Later, more research was done in the LA area, and many families of LA proved to solve the Goore Game efficiently. For example, Thathachar et al. [2] proposed the solution to the Goore Game using Variable Structure Learning Automata (VSLA) [31]. Also, Fig. 2 shows the pseudocode for the operation of GG with learning automata. Each player is represented by a Learning Automata (LA) with its actions corresponding to the player's strategies. Each LA uses the $L_{R\text{-}I}$ learning algorithm for updating its action probabilities [32]. This scheme is based on the principle that whenever the automaton receives a favorable response (i.e., a reward) from the environment, the action probabilities are updated, whereas if the automaton receives an unfavorable response (i.e., a penalty) from the environment, the action probabilities remain unchanged. Let $n_1(k)$ be the total number of times the first action is selected at iteration k [2].

$$n_1(k) = \sum_{j}^{N} I\{\alpha_j(k) = \alpha_1\} \tag{3}$$

The environment gives a reinforcement signal $\beta(k) \in [0.1]$ based on $n_1(k)$. The goal of the players is to maximize E[β(k)] by choosing the appropriate number of first actions collectively.

$$E[\beta | n_1] = G(n_1/N) \tag{4}$$

where N is the number of players. At round k of the game, player i chooses its action $\alpha_i(k)$ from two actions α_1 and α_2 as a sample realization of its action probability vector $p(k) = (p_1(k), p_2(k))$. Then, the $L_{R\text{-}I}$ algorithm based on whether the chosen action is rewarded or penalized by the referee updates its action probabilities as described below [33]. As is evident in the bellow pseudocode, the referee to reward players counts how many Yes votes there are in this round and calculates the value of its function accordingly (G(f)). Then, for every player, generate a random variable, and if the generated random number is less than G(f), the player will be rewarded. Let $\alpha(k) = \alpha_i$ and λ $(0 < \lambda < 1)$ is the learning parameter or step-size. Then, the action probability vector p(k) is updated as follows:

Inputs
K // the maximum number of iterations
P // the set of players
G(f) // a unimodal performance criterion for the referee
LA // a set of learning automata where LA_i is the leaning automaton residing in $voter_i$
G // a set of unimodal performance criterion for the referees
T_{min} // the entropy threshold
Notations:
k // the iteration counter
P// the average entropy of leaning automata in CGG
f // the ratio of "Yes" votes to all votes
Begin
k = 0
while k< K or T< T_{min} do
For each $player_i$ in P **do**
LA_i chooses one of its action $\alpha_i(k)$ from two available actions α_1("YES") and α_2("NO") according to its action probability vector $p_i(k) = (p1(k), p2(k))$;
$voter_i$ sends $\alpha_i(k)$ to the Referee;
EndFor
The referee Counts how many Yes votes there are and sets f;
For each $player_i$ in P **do**
Generate a random variable M_i;
IF $M_i < G(f)$
The referee rewards to $voter_i$;
LA_i updates its action probability vector based on its reward in this round;
EndFor
//Calculate LAs Entropy
T= the average entropy of all LAs for the players;
k = k + 1;
end while
End

Fig. 2 Pseudocode of GG with learning automata

$$p_i(k+1) = p_i(k) + \lambda B(k)\big(1 - p_i(k)\big)$$

$$p_j(k+1) = p_j(k) - \lambda\beta(k)p_j(k). \quad for\ all\ j \neq i. \tag{5}$$

\;

The analysis of GG in which each player uses linear reward-inaction learning automata (L_{R-I}) is studied in [2]. In this work, a correspondence between stable stationary points of the algorithm and the Nash equilibria of the game has been established. They have also proposed a parallel algorithm comprising a module of learning automata for each player to improve the acceleration performance. Oommen et al. [15] have shown that an unbounded accuracy for the GG can be achieved

by employing at most a constant number of LAs and recursively pruning the solution space to guarantee the solution game with a probability as close to unity as arbitrarily. In [34], verification of arbitrarily accurate solutions to the Goore Game is experimentally studied. They have shown that unbounded accuracy for the GG can be obtained by utilizing no more than three learning automata and a recursive pruning of the solution space. Granmo et al. [35] studied the accuracy of the GG solution in the presence of a finite number of players. They have shown how to achieve the unbounded accuracy of the GG by utilizing no more than three stochastic learning automata and using recursive pruning the solution space to guarantee the solution to the game with a probability as close to unity as desired. Moreover, Granmo et al. [36] propose decentralized decision-making based on the Goore Game in which each decision-maker is inherently Bayesian learning. Moreover, they have provided theoretical results on the variance of the random rewards experienced by each decision-maker, and their results show the acceleration of learning in bandit problem-based learning.

2.4 QoS control for wireless sensor networks

A WSN is composed of some sensor nodes (SN), transmitting wirelessly the information they sense. The SN is ordinarily composed of power, processing, sensing, and communication unit. The wireless communication unit is responsible for transferring the sensor measurements to the outside world. The wireless communication unit can additionally ensure a mechanism for ad hoc communication between SNs. In some situations, it might be more energy efficient to transmit a message through multi-hop communications over short distances instead of a single hop long-distance transmission to the base station (BS) or sink node [37]. It is also probable that a sensor node may not reach the BS, which must communicate with BS through multi-hop communications. Another possibility is to deploy multiple sinks instead of one for several reasons: network density, coverage area, redundancy, distribution of traffic flows, network life span, and possible energy consumption.

Energy consumption is a significant concern in WSNs. SNs must operate autonomously and independently for a considerable period in areas where replacing the node's battery is not possible or not practical [14]. When a node's energy is depleted, it will be disconnected and influence network performance. However, redundant node deployment performs the network resilient to failures and environmental changes. Moreover, if the number of nodes is too large, it would either cause traffic congestion or prevent the network's management. In this regard, an ordinarily used QoS control metric defined as the optimum number of active sensors to provide the requisite service [9]. Transmission delay or packet loss also can be utilized as the QoS metrics [38]. The control mechanism for the optimum number of active sensor nodes in WSNs that are spatially distributed and autonomous should be distributed and autonomous. Deploying multiple sinks and communicate through multi-hop communications are effective methods to achieve higher QoS of WSNs. QoS control in clustered and multiple sinks WSNs are issues that are less discussed in research

works. A summary of the current state-of-the-art in QoS control with cluster-based or multiple sinks for WSNs is presented in the following.

Tang et al. [39] presented a cluster-based linear WSN topology to analyze the QoS supporting and optimal energy allocation issues. In this paper, QoS supporting scheme can control the packet loss at individual clusters by continuously adjusting the data generating rate at individual clusters and quantitatively investigate the energy distribution in different clusters based on an optimal energy allocation criterion. [40] proposed an efficient QoS-aware data reporting combinatorial approach to provide the intra-cluster data reporting control (Intra DRC) and inter-cluster data reporting control (Inter DRC) to ensure the minimum end-to-end delay in clustered WSNs. Only partial nodes are permitted to report to satisfy the desired throughput requirement for IntraDRC. For InterDRC, cluster heads report delay-sensitive data through the shortest paths and report other data through a reporting tree to satisfy delay and energy requirements. Mazaheri et al. [41] have introduced a QoS base multipath hierarchical routing that performs cluster head election and route discovery using multiple criteria such as residual energy, remaining buffer size, signal-to-noise ratio, and distance to sink.

QoS concerns different WSN performance issues such as end-to-end delay, bandwidth, throughput, and latency [42]. In most WSN protocols, energy efficiency is considered an essential design issue to improve the network lifetime [43]. Also, due to the appearance of the multimedia and imagining sensors utilized in new WSN applications, QoS-aware energy-efficient protocols need to be developed. Nazir et al. [44] introduced Energy Efficient and QoS-aware Routing (EEQR) protocol for clustered wireless sensor networks to achieve QoS and energy efficiency. In this paper, prioritization of data is done based on message type and content to QoS for different traffic types. Then delay-sensitive messages are sent through the static sink, and delay-tolerant messages are sent through the mobile sink. [45] provided QoS-based multimedia streams by an ad hoc cluster-based protocol for logical sensor network topology. The proposed protocol drives the maximum diameter for individual clusters suitable for different types of multimedia streams. The Base station Controlled Adaptive Clustering Protocol proposed in [46] achieves fault tolerance and energy efficiency through a dual cluster head mechanism. It supports the desired QoS by including delay and bandwidth parameters in the route selection process.

Hammoudeh et al. [47] proposed a new cluster-based protocol called Route Optimization and Load-balancing (ROL), which utilizes various QoS metrics to satisfy application requirements such as network life, timely message delivery, and improve network robustness. This paper also defined a Nutrient-flow-based Distributed Clustering (NDC) balance protocol that operates seamlessly with any clustering algorithm to balance, as far as possible, the diameter and the membership of clusters. Deepa and Suguna [48] introduced a technique by merging Optimized QoS-based clustering with multipath routing protocol (OQoS-CMRP) in WSNs to reduce energy consumption in the sink coverage area. Authors in [49] have proposed centralized energy-efficient QoS-aware and Heterogeneously Clustered Routing (QHCR) for delay-sensitive real-time applications. The network area is categorized into different energy levels, and cost values are determined based on the base station's distance, initial energy levels, number of nodes, and weights. The proposed

model suggests intra-cluster multipath communication for the nodes at a significant distance from BS. Real-time multimedia and non-real-time data are transmitted over separate paths to decrease end-to-end delay conserve energy. Approaches with a genetic algorithm for energy-efficient clustering and routing in sensor-based networks are proposed in [50], enhancing the sensor's lifetime and QoS. The clustering algorithm balances the lifetime of the CHs and decreases the energy consumption of the sensor nodes. The routing scheme finds the optimum route from all cluster members to the CH by considering a trade-off between transmission distance and the number of hop-count.

The energy consumption of nodes reduces by deploying multiple sinks over the geographical area because the data packets are not needed to propagate through multiple hops to reach the sink [51]. Shen et al. [52] presented a QoS-aware multi-sink opportunistic routing (QMOR) to efficiently deliver multimedia information under QoS constraints. A redundancy reduction mechanism to decrease redundant multimedia traffic has been proposed in this paper by taking advantage of multiple sink nodes. Moreover, they selected and prioritized the forwarder list in a localized way to achieve an energy-efficient delivery of multimedia data while satisfying QoS requirements in delay and reliability. Enhanced and Threshold Sensitive Stable Election Protocol (ETSSEP) has been proposed in [53] that the selection of CH was made based on the residual energy only. Verma et al. [54] introduced QoS provisioning-based routing protocols based on multiple WSN-based IoT sinks. The proposed protocols selected Cluster Heads based on energy threshold, residual energy, distance, and node density variables and utilized three energy heterogeneity levels for network energy balancing. Stability period, network lifetime, network efficiency, networks remaining energy, throughput, latency, and reliability were the QoS provisioning performance metrics that have not been considered altogether on a single platform. Rehan et al. in [55] discussed a cross-layered on-demand multi-channel multi-sink routing protocol for QoS-aware communication in WSNs. This protocol established QoS-aware paths between source and sinks upon an event in the region of interest. It maintained load balancing during the communication session by dynamically shifting communication between the available sinks.

3 The Cellular Goore Game

The Cellular Goore Game (CGG), introduced in this paper, can be used as a model for systems consisting of simple identical components with local interactions to optimize one or more criteria. CGG is suitable for modelling systems that can be described as massive collections of simple objects interacting locally with each other. It is called cellular because it is made up of cells like points in a structure. CGG is a network of Goore Games in which at any time, every node (cell) in the network (or every node in a subset of the set of nodes in the network) plays the role of referees, each of which participates in a Goore Game with its neighboring players (voters). Like in GG, each player independently selects its optimal action between two available actions based on their gains and losses received from its adjacent referee. Players in CGG know nothing about how other

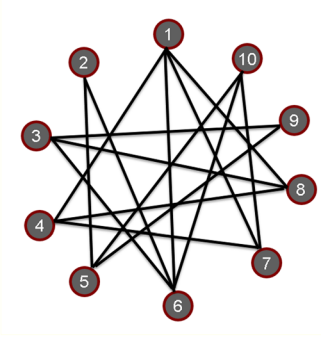
players are playing or even how/why they are rewarded/penalized. The difference between GG and CGG is that the neighborhood structure is included in this new game, which is necessary for some applications. It is also possible to optimize several criteria by simultaneously having several referees in the proposed model.

In CGG, each node simultaneously can play the role of a player or a referee. When a cell plays the role of a referee, the cells adjacent to this referee play the role of voters. In each round of CGG, all the nodes (as referees) synchronously and asynchronously play a GG with their neighboring cells (as voters). A voter may participate in more than one GG at the same time. As in GG, each referee i has a unimodal performance criterion $G_i(\lambda)$, which is optimized when the fraction of neighboring players cast "Yes" votes is exactly λ^*_i. On each round of CGG, the players vote "Yes" or "No" simultaneously and independently (they do not see each other), and the referees count the fraction, λ_i, of "Yes" votes for the adjacent player cells. The current voting round ends after each referee rewarding its players with probability $G_i(\lambda)$ and penalizing with probability $1—G_i(\lambda)$ simultaneously and independently. A player in CGG is rewarded or penalized by one of its neighboring referees selected randomly or according to some other policy. Based on the rewards received from its adjacent referee, each player then independently decides how to cast its vote for the next round. The procedure of CGG also is described step by step in Fig. 3 with a simple example.

In CGG presented in this paper, each player is modeled by a learning automaton with two actions (YES and NO) and uses the L_{R-I} learning algorithm to update its action probability vector. At round k of the game, player i chooses its action $\alpha_i(k)$ from two available actions α_1 and α_2 according to its action probability vector $p(k) = (p_1(k), p_2(k))$. Then, based on the average value of performance criterion of the corresponding neighboring referees' (or according to some other policy depending on application), the chosen action is either rewarded or penalized according to the learning algorithm. Each player in CGG has a predefined policy to calculate rewards from the received rewards of adjacent referees. This policy function, for example, can be the average or weighted average of rewards received. Also, according to the application of CGG, a player's policy may be selected one of the rewards received randomly. These policy functions of the players are effective in the performance and convergence result of CGG, and great care must be taken in selecting the proper functions appropriate to the application. Figure 4 demonstrates an example of a reward calculation scenario for the player with multiple referees. A CGG with learning automata can be formally defined as follows:

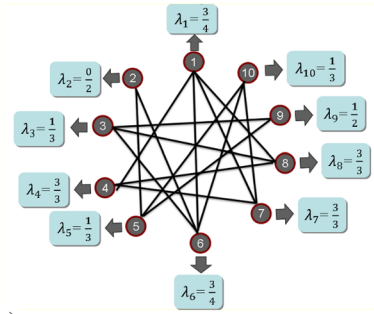
Definition 1 A Cellular Goore Game is a structure $CGG = (N, P, LA, R, G)$, where.

I. $N = (V, E)$ is an undirected network that determines the structure of CGG where $V = \{cell_1, cell_2, \ldots cell_n\}$ is the set of nodes which can be either players or referees, and E is the set of edges.
II. P is a subset of V playing the role of players.

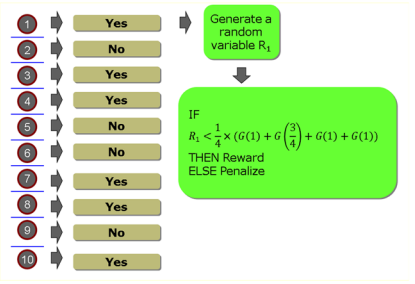


(a) *A sample structure for CGG in which each node simultaneously plays the role of a player and a referee.*

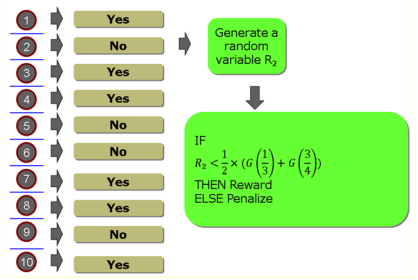
(b) *The referee nodes, with $G=0.9e^{-[(0.5-x)\ *(0.5-x)/0.0625]}$ as performance criterion, conduct a series of voting rounds. On each round, the voters vote either "Yes" or "No" (the issue is unimportant) simultaneously and independently (they do not see each other).*



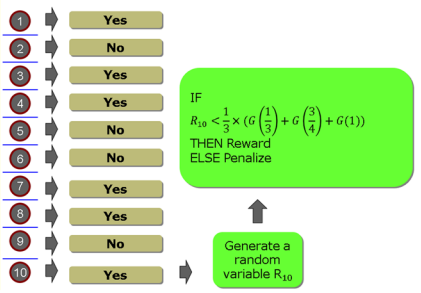
(c) *Referees i (1<= i<=10) count the fraction, λ_i, of "Yes" votes for its neighboring player nodes. For example, the first referee is adjacent to players 4,6,7 and 8, and λ_1 is equal to the ratio of its neighbors who choose yes.*



(d) *A player in CGG is rewarded or penalized based on its average rewards received from its adjacent referees. The first player is adjacent to referees 4,6,7 and 8, This player is rewarded or penalized based on the average reward received from these referees.*

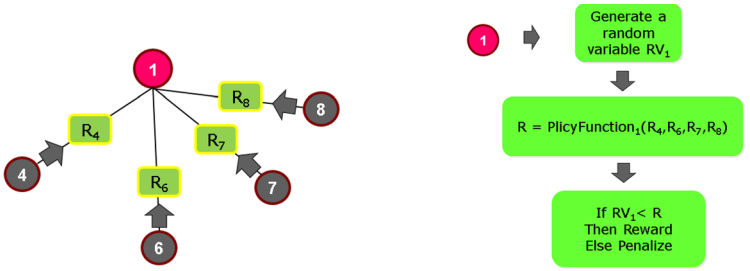


(e) *Rewarding or penalizing the second player 2 (note that this player is adjacent to referees 5 and 6)*



(f) *Rewarding or penalizing the player 10 (note that this player is adjacent to referees 5 and 7) which completes a round. A new round is started and all platers cast again, their votes.*

Fig. 3 A step-by-step illustration of CGG operation



(a) *Player 1 gains reward from adjacent referees (As shown in the graph, this player is adjacent to referees 4,6,7 and 8).* (b) *This player calculates its reward in each round based on its predefined policy function and received rewards.*

Fig. 4 An example of a reward calculation scenarios for the player with multiple referees

III. R is a subset of V playing the role of referees (the intersection of sets P and R may or may not be empty)
IV. $LA = \{LA_1, LA_2, \ldots, LA_n\}$ is a set of learning automata where LA_i is the learning automaton residing in $cell_i$.
V. $G = \{G_1, G_2, \ldots G_n\}$ is a set of unimodal performance criteria for the referees, where G_i is the performance criterion for $cell_i$.

The goal of CGG is to maximize the total performance criterion, that is the sum of G_i's where G_i is the performance criterion for $referee_i$.

CGG can be either synchronous or asynchronous. In synchronous CGG, in each round of CGG, all the cells in CGG are considered referees, each of which starts playing a Goore Game with its neighboring cells as players. In asynchronous CGG, only some of the nodes are considered referees at each round, and each plays a Goore Game with its neighboring nodes as players. In asynchronous CGG, referees' cells may play in either a time-driven or step-driven manner. In time-driven asynchronous CGG, each cell is assumed to have an internal clock that activates a cell as a referee, while in step-driven, a cell is activated as a referee in a fixed or random sequence. Also, CGG can be either homogeneous or inhomogeneous depending on whether the referees' performance criterion functions are identical or not. The performance criterion function for referees may differ depending on the specific application it is applied for. Moreover, one may assume that only some of the nodes can play the role of referee. We call a synchronous and homogeneous CGG for which P=V=R standard CGG. Figure 5 shows the pseudocode for the operation of standard CGG, which uses learning automata as the learning paradigm to model players.

Theorem 1 *The time complexity of the standard CGG algorithm with learning automata is* $O(KC|E|/|V|)$, *where* K *is the maximum number of iterations and* C *is the average latency of node responses. Also,* $|E|$ *is the size of the set of graph edges, and* $|V|$ *is the number of nodes.*

Inputs
N = (V, E) // an undirected network that determines the structure of CGG
V= *{cell₁, cell₂, …, cellₙ}* // the set of vertices
E // the set of edges
P // the set of player cells
R // the set of referee cells
LA // a set of learning automata where LA_i is the leaning automaton residing in $cell_i$
G // a unimodal performance criterion for the referees
K // the maximum number of iterations
T_{min} // the entropy threshold
Notations:
k // the iteration counter
T// the average entropy of leaning automatons which belong to LA
Begin
k = 0
while k< K or T< T_{min} do
  For each $cell_i$ in P **do in Parallel**
    LA_i chooses one of its action $\alpha_i(k)$ from two available actions α_1 ("YES") and α_2 ("NO") according to its action probability vector $p_i(k) = (p1(k), p2(k))$;
    $cell_i$ sends $\alpha_i(k)$ to adjacent referees;
  EndFor
  For each $cell_j$ in R **do in Parallel**
    $cell_j$ count the fraction, λ_j, of "Yes" votes for neighboring player cells;
    $cell_j$ rewards with probability $G(\lambda_j)$ every adjacent player cell independently;
  EndFor
  For each $cell_i$ in P **do in Parallel**
    $cell_i$ calculates its reward through its neighbor referees;
    LA_i updates its action probability vector based on its reward in this round;
  EndFor
  //Calculate LAs Entropy
  T = the average entropy of all LAs in the player cells;
  k = k + 1;
end while
End

Fig. 5 Pseudocode of the standard CGG with learning automata

Proof According to the Pseudocode above, the player nodes first select their action parallel and send it to the adjacent referees. Given that the average degree of nodes is equal to |E|/|V|, its time complexity equals O(C|E|/|V|), and C is the average latency of node responses. The referees then calculate the rewards of their neighboring players parallelly and send them that the time complexity of this section is equal to O(C|E|/|V|) due to the standard CGG(V = R). Eventually, players update their action probability vector based on its reward in this round that its time complexity equals O(c). Hence, the time complexity of the standard CGG is K(O(C|E|/|V|) + O(C|E|/|V|) + O(c)) = O(KC|E|/|V|).

4 Experiments

In this section, several experiments have been conducted to study the behavior of CGG in terms of two criteria: the average entropy and the scaled average performance function, which are defined in the following paragraphs.

Entropy may be used to study the changes that occur in the states of the cells of CGG. H(t) value will be zero if all CGG learning automata no longer change their selected action. Higher values of H(t) mean higher rates of changes in the actions selected by learning automata residing in the cells of the CGG. The average *entropy* of the CGG at iteration t is defined by the equation given below:

$$H(t) = -1/n \sum_{l=1}^{n} H^l(t) \tag{6}$$

$\boldsymbol{H^l(t)}$ is the *entropy* of $cell_l$ of CGG, and n is the number of player cells in CGG. The *entropy* of $cell_l$ is defined by the equation given below:

$$H^l(t) = \sum_{i \in C_l(t)} H_i(t) \tag{7}$$

$\boldsymbol{H_i(t)}$, which is given by Eq. (7), is the entropy of learning automaton LA_i. In this equation, $C_l(t)$ denotes the set of indices of learning automata residing in $cell_l$.

$$H_i(t) = \sum_{j=1}^{r_i} p_{ij}(t) \cdot \ln\left(p_{ij}(t)\right) \tag{8}$$

r_i is the number of actions of learning automaton LA_i. p_{ij}(t) is the probability of selecting action α_j by learning automaton LA_i at iteration t. The *scaled average performance function* G_{avg}(t), defined by below equation, is the average referees' performance criterion scaled between (0,1) through dividing the average performance function value at iteration t by the maximum value of the referees' performance criterion functions.

$$G_{\text{avg}}(t) = \frac{\sum_{i=1}^{N} G^i\left(\lambda_t^i\right)}{N \times \text{Max}\left(\left[G^1\left(\lambda_1^*\right). \ldots . G^N\left(\lambda_N^*\right)\right]\right)};$$

$$\lambda_t^i = \frac{\sum_{j}^{N^i} I\left\{\alpha_j(t) = \alpha_1(\text{action} = \text{yes})\right\}}{N^i};$$

$$\forall \lambda_i^*.\lambda_i \in [0.1] \tag{9}$$

where N is the number of referee cells and N^i is the total number of adjacent player in referee cell_i. G^i is a unimodal performance criterion of referee cell_i and I is the indicator function of event α_1 (action = yes). λ_t^i is the fraction of "Yes" votes

(action = yes) at round t for the adjacent player cells for referee cell_i, and λ_i^* is the optimum fraction of "Yes" votes in this cell that maximizes G^i.

For all the experiments, the learning algorithm used by the learning automata in each node is $\text{L}_{\text{R}-\text{I}}\,\text{L}_{\text{R}-\text{I}}$. Each referee has a unimodal performance criterion as given below unless another function is mentioned in the experiment:

$$G(\lambda) = 0.9e^{-\left[(0.7-\lambda)*\frac{(0.7-\lambda)}{0.0625}\right]}; \lambda = \frac{\sum_j^N I\{\alpha_j(k) = \alpha_1\}}{N}; \forall \lambda \in [0.1] \tag{10}$$

where I is the indicator function of event α_1 (action = yes). Stopping criteria is either when the number of iterations reach 10,000 or the average entropy value of less than 0.1. All the results reported are the averages taken over 30 runs to reduce the statistical bias. Table 2 describes the types of graphs that are used for the experiments. Experiments are conducted in python on a PC, which has a single CPU of Intel(R) Corei7-960 3.2 GHz and 16 GB of memory.

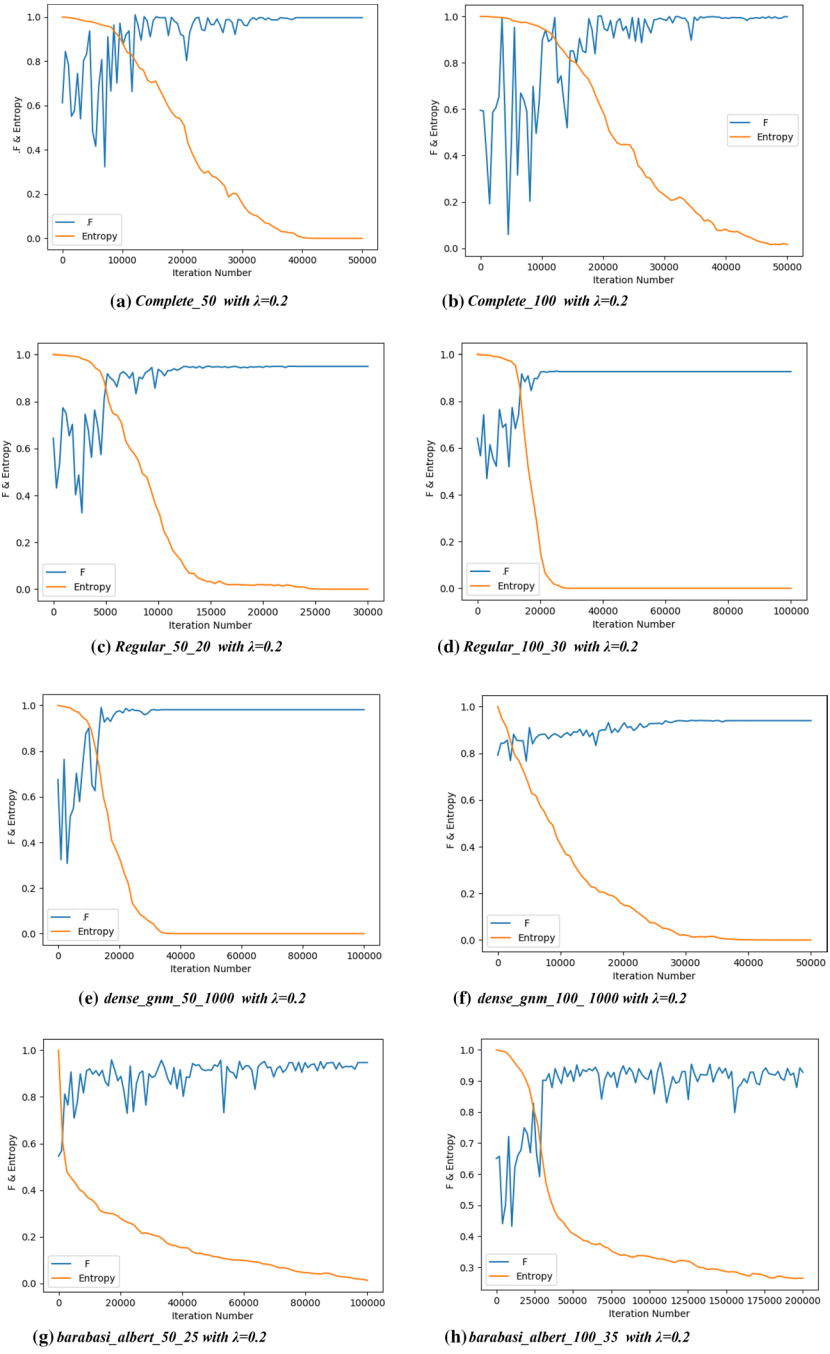
4.1 Experiment I

We analyze the effect of graph type on the behavior of the CGG by performing experiments on different graphs in terms of the average entropy and the scaled average performance function. We plot both the average entropy value and the $G_{\text{avg}}(t)$ versus iteration number. The results of this experiment are depicted in Fig. 6. As shown in Fig. 6, the complete, random regular, dense_gnm, and barabasi_albert graphs with the number of nodes 50 and 100 have been selected to perform these experiments. According to the obtained results in this experiment, we may conclude the following.

- $G_{\text{avg}}(t)$ gradually increases to its maximum value, and at the same time, entropy decreases and gets close to zero in different types of graphs (Complete, Regular, dense_gnm, and barabasi_albert) utilized in this experiment.
- In different types of graphs in this experiment, $G_{\text{avg}}(t)$ of standard CGG eventually converges to a value close to one. However, the round at which convergence occurs and the amount to which it converges is slightly different in various

Table 2 Graphs used in simulations

Graph Name	Description
Complete_n	A complete graph with n nodes
Regular_n_d	A random regular graph of n nodes each with degree d
dense_gnm_n_m	A graph is chosen uniformly at random from the set of all graphs with n nodes and m edges [56]
barabasi_albert_n_m	A graph of n nodes is grown by attaching new nodes, each with m edges that are preferentially attached to existing nodes with a high degree [57].



(a) *Complete_50 with λ=0.2*

(b) *Complete_100 with λ=0.2*

(c) *Regular_50_20 with λ=0.2*

(d) *Regular_100_30 with λ=0.2*

(e) *dense_gnm_50_1000 with λ=0.2*

(f) *dense_gnm_100_ 1000 with λ=0.2*

(g) *barabasi_albert_50_25 with λ=0.2*

(h) *barabasi_albert_100_35 with λ=0.2*

Fig. 6 the plot of G_{avg} and entropy versus iteration number for different graphs in experiment I

graphs. The regular graph has converged earlier than the others, and the barabasi_albert graph has converged later than other graphs in this experiment.

- The values of convergence G_{avg} are remarkably close in the various graphs presented in this experiment. G_{avg} (t) of standard CGG finally converges to the highest value in the complete graph and the lowest value in the barabasi_albert graph compared to other graphs. In general, it can be concluded that the value that converges will be higher if the degrees of the vertices of the graph are higher, but the rate of convergence will be slower.
- As can be seen from the results of this experiment, increasing the number of nodes in different types of graphs does not have much effect on the amount of convergence. However, it does affect the convergence rate, and the higher the number of nodes, the slightly lower the convergence rate.

4.2 Experiment II

This experiment is conducted to investigate the effect of graph type on the CGG in terms of G_{avg}. The result of this experiment is depicted in Fig. 7 for regular, barabasi_albert, and dense_gnm graphs with the same number of nodes and edges for a fair comparison. According to the result of this experiment, one may conclude that all three types of graphs converge in almost the same round of the CGG. Also, the values of convergence G_{avg} are approximately close in the different graphs presented in this experiment. G_{avg} (t) converges to the highest value in the regular graph and the lowest value in the dense_gnm graph. However, the convergence values in different graphs with the same number of edges and vertices are very close.

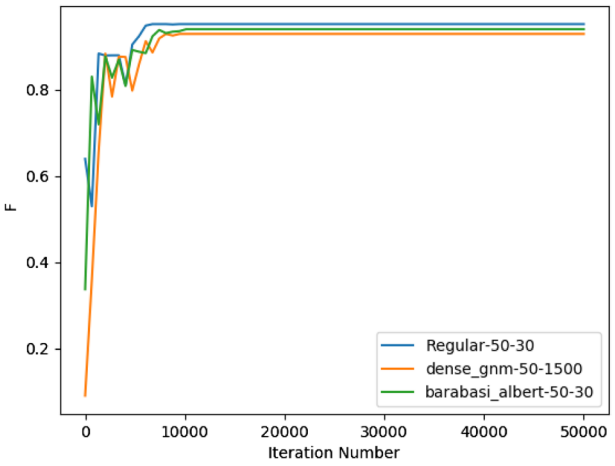
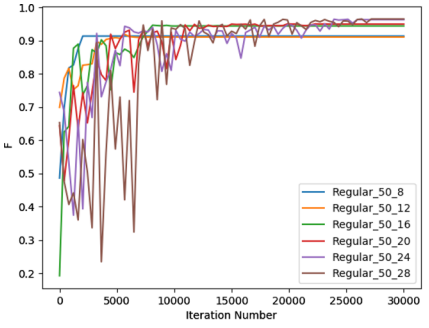


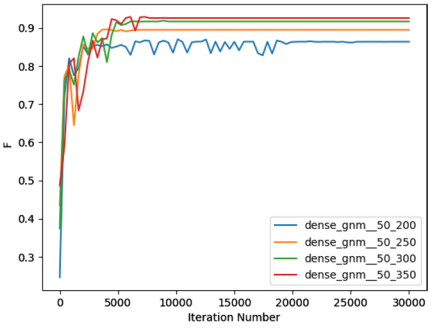
Fig. 7 Plot of G_{avg} versus iteration number for different type of graphs in experiment II with $\lambda = 0.1$

4.3 Experiment III

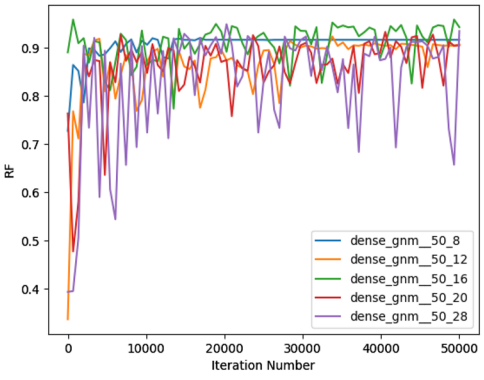
This experiment is conducted to show the impact of the degree of graph nodes on the behavior of CGG. For this purpose, CGG runs on graphs with different degrees of graph nodes, and the result is reported in Fig. 8. The complete, random regular, and dense_gnm graphs have been chosen to conduct these experiments, as shown in Fig. 8. We may observe that increasing the degree of the graph's vertices improves the average performance criterion function in almost all cases and reduces the speed of convergence of the algorithm. This result justifies the fact that the accuracy of the solution achieved is intricately related to the number of players participating in the Goore Game (resolution of gore game) [34]. For a moderate increase in the number of players, the value of the learning parameter needed to be drastically reduced for good accuracy, adversely affecting the algorithm's convergence speed.



(a) *Regular graphs with =0.1.*



(b) *dense_gnm graphs with λ=0.1*



(c) *barabasi_albert graphs with λ=0.2*

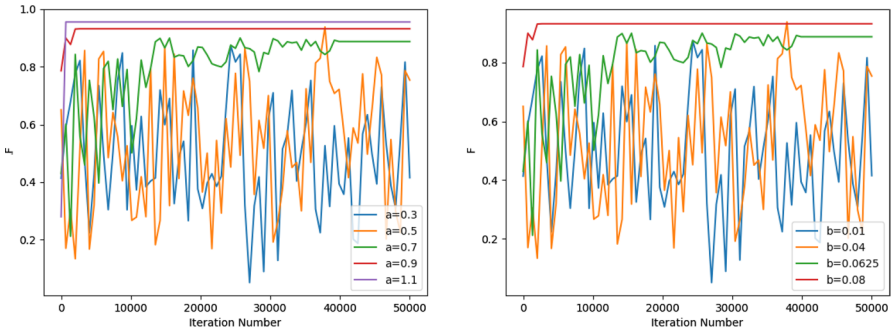
Fig. 8 Plot of G_{avg} versus iteration number in experiment III

4.4 Experiment IV

In the experiments, we investigate the behavior of CGG for different criterion functions. We use the Gaussian criterion function $g(\lambda) = ae^{-\left(\frac{(\lambda-\lambda_0)}{b}\right)}$, for different magnitudes and peakedness to be controlled by parameters a and b, respectively. The average performance criterion function G_{avg} versus iteration number is plotted for different values of learning parameters a and b. The results of this experiment are reported for the regular graph in Fig. 9. To examine the effect of parameter a, as you can see in Fig. 9a the values of other parameters, such as b, are kept constant for a fair comparison. To study the effect of b, as shown in Fig. 9b, the value of a and other parameters is constant. According to the results of this experiment, we may conclude that any values of a and b do not lead to convergence in CGG, and care must be taken in selecting the appropriate value.

5 Application of Cellular Goore Game to QoS control in wireless sensor networks

This section proposes two CGG-based algorithms for QoS control in WSN: one for clustered WSNs and one for multiple sinks WSNs. The proposed algorithms are named CGG-based QoS Algorithm for Clustered WSNs (CGG-QoSAC), and CGG-based QoS Algorithm for Multi Sinks WSNs (CGG-QoSAMS) are described in the following two subsections. Finally, the performance of these algorithms is analyzed through computer simulation in the experiment section and showed that they are superior to some QoS control methods in terms of the QoS control performance metrics.



(a) ***Regular_50_25 graph with λ=0.1 and b=0.0625.*** **(b)** ***Regular_50_25 graph with λ=0.1 and a=0.9.***

Fig. 9 Plot of G_{avg} versus iteration number to examine the effect of the degree of the criterion function in experiment IV

5.1 A CGG-Based algorithm for QoS control in clustered WSNs

Algorithms based on GG for finding the optimum number of sensors in WSNs dynamically are only applicable to one-hop WSNs with star-topology in which each sensor node communicates via a single hop to a Base Station (BS) [9]. When the scale of a network is large, it can usually be expertly divided into clusters. Each cluster has a cluster head (CH) serving as an intermediate between the cluster nodes and the base station (BS). In GG-based algorithms, one GG may be applied to each cluster [11]. However, if the clusters overlap, it may not be feasible to use one independent GG for each cluster because some sensors may belong to two or more clusters. To solve this problem, we present a QoS control algorithm that uses CGG. In this algorithm, cluster heads play the referee's role that rewards/punishes their cluster sensors using a unimodal performance criterion to attain optimal energy usage in the network. Each sensor node plays the role of a player that chooses between transmitting data or remaining idle.

Let us assume that in WSN, there are one base station B and m sensors S_1 through S_m, grouped into k clusters. We denote the ith cluster head by CH_i where $i \in \{1, 2\ldots, k\}$. The sensor network can be modeled as a graph $N=(V, E)$ where each node set V represents a sensor or a CH, and an edge in E represents a connection between a sensor and its CH. Figure 10 depicts an example of a WSN with overlapping clusters. Each CH node plays the role of a referee and has a specific unimodal performance criterion which is maximized at the ratio of the predefined optimum number of active sensor nodes to the total number of sensors in the cluster. A unimodal performance criterion can be generally defined as follows:

$$G_i(\lambda_t) = c + ae^{-\left(\frac{\left(\lambda_i^* - \lambda_t\right)}{b}\right)^2} \tag{11}$$

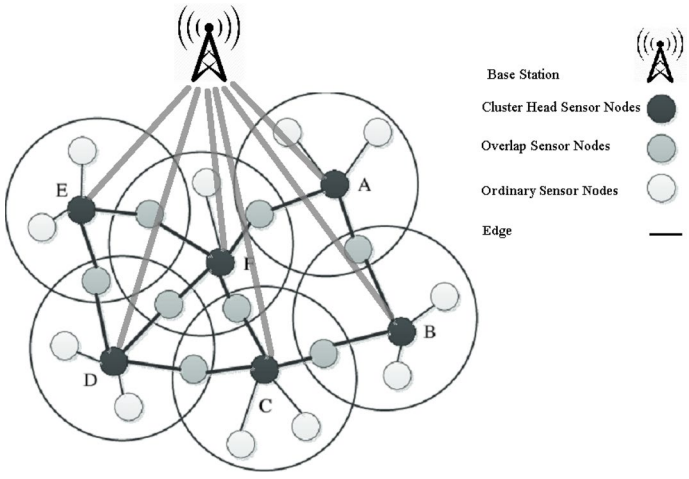


Fig. 10 Graphical model of a clustered WSN

where c, a, and b are learning parameters, λ_t is the number of active sensor nodes for its cluster in round t, and λ_i^* is the optimum number of active sensor nodes in the *ith* cluster. Each CH, due to its required QoS, specifies the desired value, and in order to achieve this desired value, it rewards its adjacent nodes according to the performance criterion function.

Each sensor (as a player) decides between sending data or remaining idle based on its reward received from CHs of its cluster. The goal of each player is to get the most rewards in total from the adjacent referees. Every sensor node that belongs to more than one cluster sends its status to the CHs of those clusters and receives a reward from them. According to the predefined policy, each sensor based on the values of received reward from neighboring CHs' calculates the reward value in this round. This predefined policy can be the average, maximum, random selection, or weighted average of received rewards. In this problem, to save energy, each sensor, according to each cluster's energy level, selects one of the cluster heads to which it belongs to send its data. Then, each player, based on its calculated reward, decides again to send data to the adjacent referees or to remain idle for the next round. Each player sensor can be modeled with a learning automaton with two actions (active and sleep).

For example, the WSN demonstrated in Fig. 10 consists of 19 ordinary sensors grouped into six overlapping clusters. The graph $N=(V, E)$ models this WSN that V includes ordinary sensors ($S=\{S_1, S_2, \ldots, S_{19}\}$) and cluster heads ($CH=\{CH_A, CH_B, CH_C, CH_D, CH_E, CH_F\}$), and E describes the connection set between ordinary sensors and their CH. In CGG-based approach for QoS control, each ordinary sensor node takes a player role, and each cluster head plays a referee role. Players only communicate with cluster heads that are in their clusters. In each round, players decide between sending data to their CHs or remaining idle to gain maximum rewards in CGG. Then CHs reward their neighboring players according to their performance criterion and the number of active adjacent sensors. In this WSN, an overlap sensor node between clusters A and B has a predefined policy to calculate the reward in each round. Each sensor can transmit its data to all the cluster heads to which it belongs or select one of them to save energy and reduce interference in the network. After several rounds, active sensors in each cluster dynamically converge to the optimal number of active sensors according to their cluster QoS needs. In this algorithm, after a percentage of the sensors have lost their energy, the sensors in sleep mode change their status and send packets. Figure 11 presents the pseudocode of CGG-based QoS control in clustered WSNs.

Theorem 1 *The time complexity of the CGG-based QoS control in clustered WSNs is O* ($KC|S|/k$), *where K is the maximum number of iterations and C is the average latency of node responses. Also*, $|S|$ *is the size of the ordinary sensors, and k is the number of clusters.*

Proof According to the pseudocode, the sensor nodes select their action parallel and send it to the adjacent candidate cluster head. Its time complexity is equal to O(C). The cluster heads then compute the rewards of their neighboring sensors and

Inputs
B // the base station
m // total number of sensors in the wireless network
S = {S_1, S_2,…,S_m} // Set of sensors in the wireless network
k // total number of clusters
K // the maximum number of iterations
T_{min} // the entropy threshold
Notations:
t // the iteration counter
CH = {CH_1, CH_2, …, CH_k} // Set of cluster heads in the wireless network
N = (V, E) // an undirected network that determines the structure of CGG
V= {$cell_1$, $cell_2$, …, $cell_{m+k}$} // the set of vertices
E // the set of edges
P // the set of player cells
R // the set of referee cells
LA // a set of learning automata where LA_i is the leaning automaton residing in ith sensor
G // a set of unimodal performance criterion {G_1,G_2,…,G_K} for the referees
TE = { TE_1, TE_2, …, TE_k }// the set of sum remaining energies in clusters
Policy_Functions = {PF_1, PF_2, …, PF_m} // the set of policy functions for player sensors
G // a set of unimodal performance criterion for the referees
T// the average entropy of all LAs in the player cells
Begin
k = 0
while k< K or T< T_{min} do
Sensors that belong to S are grouped into k clusters (the overlap between clusters is allowed);
The head of the clusters is also specified in the clustering and placed in the cluster set;
$V = S \cup CH$;
$E = \{(x, y) | x \in S \,\&\, y \in CH \,\&\, x \text{ is a member of a cluster headed by } y\}$;
P = S;
R = CH;
For each S_i in S **do in Parallel**
LA_i chooses one of its action $\alpha_i(t)$ from two available actions α_1("Active") and α_2("Sleep") according to its action probability vector $p_i(t)$ = (p1(t), p2(t));
If ($\alpha_i(t) == \alpha_1$ and S_i has enough energy to send information)
$CH_{condidated}$ = S_i selects one of the adjacent referees(cluster heads) based on their remaining energy level;
S_i sends the information it receives to the $CH_{condidated}$;
EndFor
For each CH_j in CH **do in Parallel**
CH_j count the fraction, λ_j, of adjacent players (sensors) sent information to it in this round;
CH_j rewards with probability $G_i(\lambda_j)$ to every adjacent player cell independently by sending packets with TE_i;
EndFor
For each S_i in S **do in Parallel**
//S_i calculates its reward through its neighbor referees;
R_i=PF_i (reward set from its neighbor referees in this round, the set of sum remaining energies in its neighbor clusters)
LA_i updates its action probability vector based on R_i;
EndFor
For each CH_j in CH **do in Parallel**
CH_j sends the information gathered in this round to B;
EndFor
//Calculate LAs Information Entropy
T = average entropy of all LAs in the player cells;
k =k + 1;
end while
End

Fig. 11 Pseudocode of CGG-based QoS control in clustered WSNs

transmit that the time complexity of this section is equal to O(C|S|/k). Sensor nodes update their action probability vector based on its reward in this round that its time complexity equals O(C). Finally, cluster heads send the information gathered to the base station (O(C)). Hence, the time complexity of the CGG-based QoS control in clustered WSNs is K(O(C) + O(C|S|/k) + O(C) + O(C)) = O(KC|S|/k).

5.2 A CGG-Based algorithm for QoS control in WSNs with multiple sinks

Traditional WSNs have one data gathering point called base station or sink [55]. In a large WSN, energy consumption becomes inefficient while collecting all information in a single sink [58]. Another issue may be the single point of failure related to a single sink, due to energy reduction or other problems. A feasible solution to these issues is to employ a multiple-sink approach to minimize energy consumption by reducing the distance between sensors and sinks. Determining the optimal placement of multiple sink nodes in WSNs has been studied by researchers [59]. In this section, a CGG-based QoS control algorithm for WSNs with multiple sinks is proposed, which dynamically adjusts the number of active sensors during the operation of the WSN.

We assume in WSN there are m sensors denoted by $S = \{S_1, S_2, \ldots, S_m\}$ and k sink nodes denoted by $B = \{B_1, B_2, \ldots, B_k\}$. Sensors are clustered into k groups, denoted by $C = \{C_1, C_2, \ldots, C_k\}$, based on their distance to the sink nodes, considered cluster heads. The overlap between clusters is allowed. The network can be modeled as graph $N = (V, E)$ where V is the set of nodes that include sensors nodes and sink nodes and E defined below is the set of edges. Each edge represents a connection between a sensor and its sink.

$$E = \left\{ \left(x.B_i \right) | x \in S \& B_i \in B \& x \in C \right\} \tag{12}$$

Figure 12 represents an example of a WSN with multiple sinks and overlapping clusters. Each sink plays the role of a referee and each sensor plays the player's role in CGG-based algorithm. Each sink rewards its cluster sensors employing a unimodal performance criterion, and then each sensor chooses between remaining active or idle based on its received reward. Figure 13 presents the pseudocode of CGG-based QoS control in WSNs with multiple sinks. A performance criterion can be defined as follows:

$$G_i\left(\lambda_t\right) = c + a e^{-\left(\frac{\left(\lambda_i^* - \lambda_t\right)}{b}\right)^2} \tag{13}$$

where c, a, and b are learning parameters, λ_t is the number of active sensor nodes for its sink in round t, and λ_i^* is the optimum number of active sensor nodes in the ith sink. Each sink defines the desired value based on its required QoS, and it rewards its adjacent nodes to gain this desired value according to the performance criterion function. Players interact with sinks that are in their groups. Players to achieve maximum rewards choose between sending data to their sink or remaining asleep. Next,

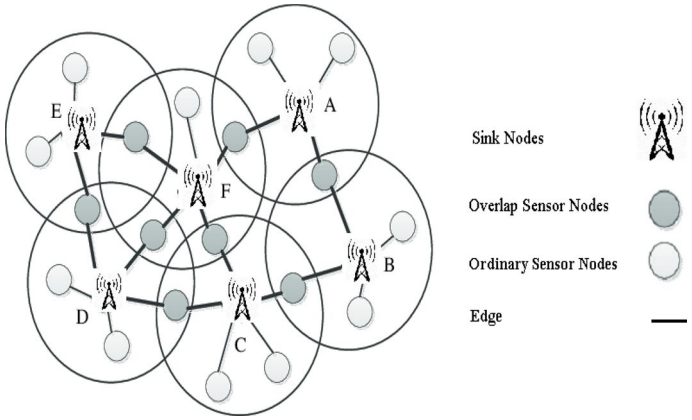


Fig. 12 Graphical model of a WSN with multiple sinks

according to their performance criterion and the number of active neighboring sensors, sinks as referee roles reward their adjacent players. An overlap sensor node between multiple sinks has a predefined policy to calculate the reward in each round (for example, this policy can be the average reward obtained). Each sensor can send its data to all the sinks to which it belongs or select one of them to save energy and reduce interference in the network. Sinks approximately after several rounds gain the optimal number of active sensors according to their QoS needs. Also, in the proposed method, after a percentage of the sensors have lost their energy, the sensors in sleep mode change their status and send packets.

Theorem 1 *The time complexity of the CGG-based QoS control in WSNs with multiple sinks is O* (*KC*|*S*|/*k*), *where K is the maximum number of iterations and C is the average latency of node responses*. *Also*, |*S*| *is the size of the ordinary sensors*, *and k is the number of sinks*.

Proof According to the Pseudocode, the sensor nodes select their action parallelly and send it to the adjacent candidate sink. So its time complexity is equal to O(C). The sinks then calculate the rewards of their neighboring sensors and transmit them that the time complexity of this section is equal to O(C|S|/k). Eventually, sensor nodes update their action probability vector based on its reward in this round that its time complexity equals O(C). Hence, the time complexity of the CGG-based QoS control in clustered WSNs is $K(O|C|+O(C|S|/k)+O(|C|))=O(KC|S|/k)$.

5.3 Experiments

In this section, several experiments using NS3 network simulator tool version 3.29 [60] are conducted to evaluate the proposed algorithms. General simulation parameters are given in Table 3. Sensors are deployed randomly throughout a square with coordinates ranging from 0 to 100, as shown in Figs. 14 and 15 in experiments.

Inputs
k // total number of sink nodes
$B = \{B_1, B_2, \ldots, B_k\}$ // Set of sink nodes in the wireless network
m // total number of sensors in the wireless network
$S = \{S_1, S_2, \ldots, S_m\}$ // Set of sensors in the wireless network
K // the maximum number of iterations
T_{min} // the entropy threshold
Notations:
t // the iteration counter
N = (V, E) // an undirected network that determines the structure of CGG
$V = \{cell_1, cell_2, \ldots, cell_{m+k}\}$ // the set of vertices
E // the set of edges
P // the set of player cells
R // the set of referee cells
LA // a set of learning automata where LA_i is the leaning automaton residing in ith sensor
$Policy\_Functions = \{PF_1, PF_2, \ldots, PF_m\}$ // the set of policy functions for player sensors
G // a set of unimodal performance criterion $\{G_1, G_2, \ldots, G_K\}$ for the referees
T// average entropy of all LAs in the player cells
Begin
k = 0
while t< K or T< T_{min} do
Sensors that belong to S are grouped into k clusters based on sink nodes which are considered the cluster heads of those clusters (the overlap between clusters is allowed).
$V = S \cup B$;
$E = \{(x.y) | x \in S \,\&\, y \in B \,\&\, x \text{ is a member of a cluster headed by } y\}$;
P = S;
R =B;
For each S_i in S **do in Parallel**
LA_i chooses one of its action $\alpha_i(t)$ from two available actions α_1("Active") and α_2("Sleep") according to its action probability vector $p_i(t) = (p1(t), p2(t))$;
If ($\alpha_i(t) == \alpha_1$ and S_i has enough energy to send information)
S_i sends the information it receives to the adjacent referees (sink nodes);
EndFor
For each B_j in B **do in Parallel**
B_j count the fraction, λ_j, of adjacent players (sensors) sent information to it in this round;
B_j rewards with probability $G_i(\lambda_j)$ to every adjacent player cell independently by sending packets;
EndFor
For each S_i in S **do in Parallel**
//S_i calculates its reward through its neighbor referees;
$R_i = PF_i$(reward set from its neighbor referees in this round, the set of sum remaining energies in its neighbor clusters)
LA_i updates its action probability vector based on R_i;
EndFor
//Calculate LAs Information Entropy
T = average entropy of all LAs in the player cells;
k =k + 1;
end whiles
End

Fig. 13 Pseudocode of CGG-based QoS control in WSNs with multiple sinks

Table 3 General simulation parameters

Simulation parameter	Value
Simulation time	2000s
Number of nodes	100
Number of clusters	5
Playground size	100×100 m^2
Transmission power	1.1 mW
Package length	4000 bits
Header length	48 bits
Bit rate	250 kbps
Total energy of network	80 J

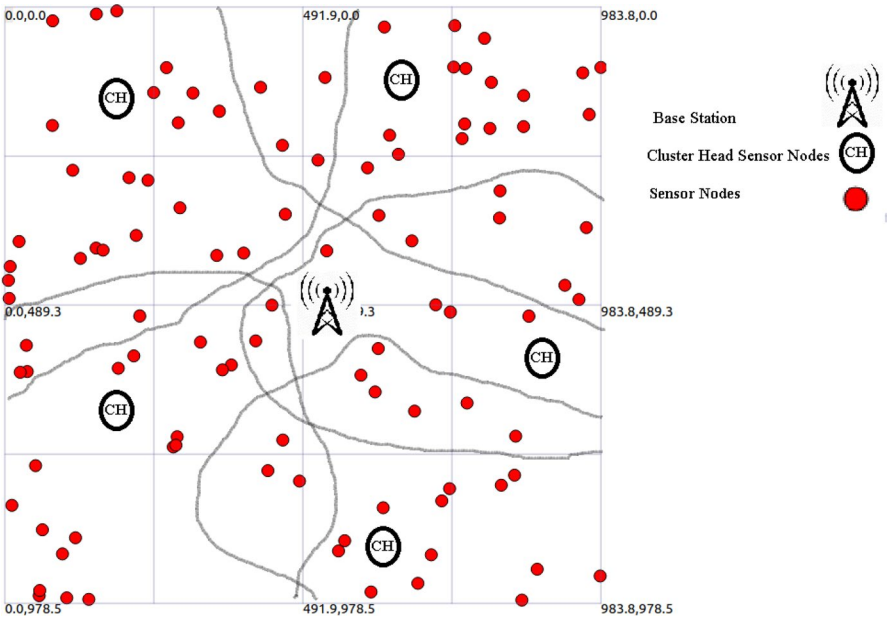


Fig. 14 Network scenario of 100 randomly deployed sensor nodes in a clustered WSN with five clusters

Clustering is based on the distance between sensors. In order to study the performance of the proposed algorithm, the average entropy, the scaled average performance function, and QoS provisioning performance metrics: stability period, network lifetime, network efficiency, networks remaining energy and throughput are utilized, which are defined in the following paragraphs.

The number of rounds between the start of the network and the first sensor node dead is defined as the stability period [61]. When the first node dies, the network tends to lose its stability. The network lifetime is the number of rounds until the last sensor node dies [54]. The network efficiency is the number of rounds that half

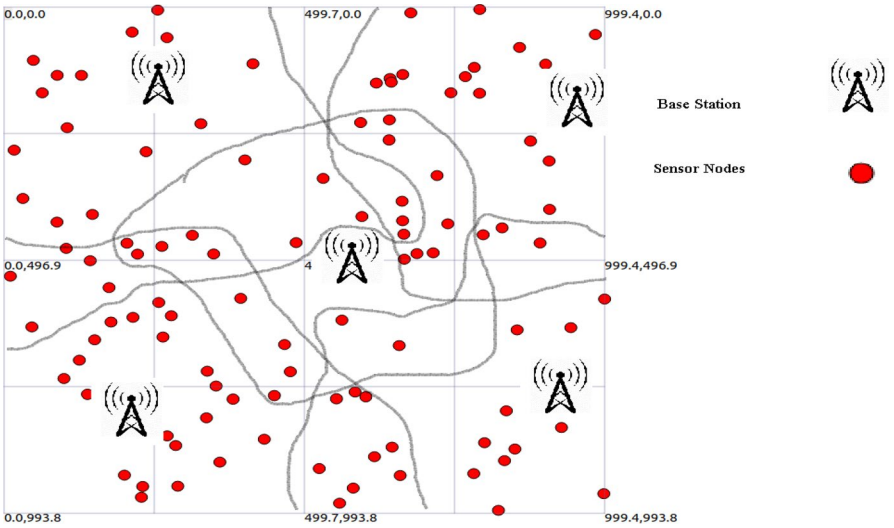
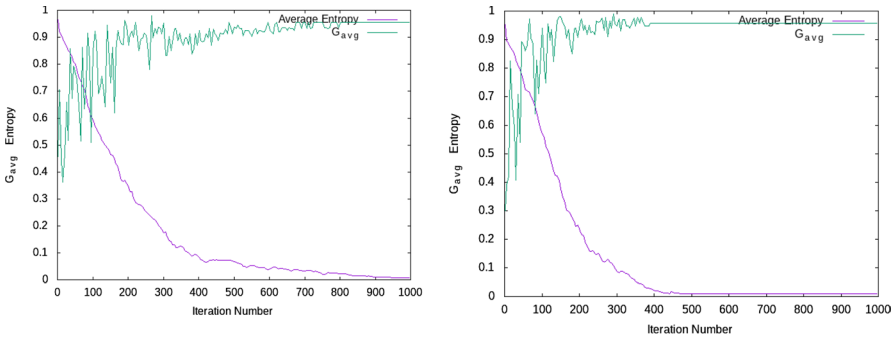


Fig. 15 Network scenario of 100 randomly deployed sensor nodes in a clustered WSN with five sink nodes

sensor nodes dead and shows the overall load balancing in the network. The network remaining energy metric is defined as the speed by which the energies of nodes are depleted. The network throughput measures the number of packets per unit time received at the sink [62].

5.3.1 Experiment I

This experiment evaluates the proposed CGG-based algorithms in terms of the average entropy and $G_{\text{avg}}(t)$. $G_{\text{avg}}(t)$ metric defined in the previous section indicates a level of QoS in WSN implicitly. Figure 16a shows the simulation results for



(a) *CGG-QoSAC algorithm.* (b) *CGG-QoSAMS algorithm.*

Fig. 16 the plot of G_{avg} and the average entropy of the proposed algorithms for WSNs

clustered WSNs with five clusters, and Fig. 16b presents the simulation results for WSNs with five sinks and $\lambda = 0.1$. The results confirm that the CGG approach maintains good QoS values in WSNs because the G_{avg} has converged to an acceptable value in a satisfactory duration. This figure shows that the value of average entropy is high at early rounds and gradually decreases. This indicates the decreasing rates of changes, and finally, the rate of change reaches zero in the actions selected by learning automata residing in the sensor nodes of the CGG.

5.3.2 Experiment II

This experiment investigates the QoS provisioning performance of the CGG-based QoS algorithm for clustered WSNs. We compare the performance of the CGG-QoSAC algorithm with the ETSSEP protocol [53] and the O-ETSSEP protocol [54] for demonstrating a comparative analysis. Also, we compare a protocol that utilizes the original Goore Game [2] separately in each disjoint cluster for QoS (GG-QoSAC) with the proposed method. In this paper, the fundamental radio energy consumption model [63] is applied for the energy consumption model of WSN in NS3. The specifications of the packets exchanged in the network, such as packet length, are given in Table 3. Each sensor node has 0.5 J energy as initial power, and the base station and cluster heads have infinite energy. The results of this experiment are given in Figs. 17, 18, 19 and Table 4. According to the results of this experiment based on QoS provisioning performance metrics, we may conclude the following.

- The stability period achieved by the CGG-QoSAC algorithm is 588 rounds which previously were 1612 rounds, 1934 rounds, and 476 rounds in the case of ETSSEP, O-ETSSEP, and GG-QoSAC protocols, respectively, as presented

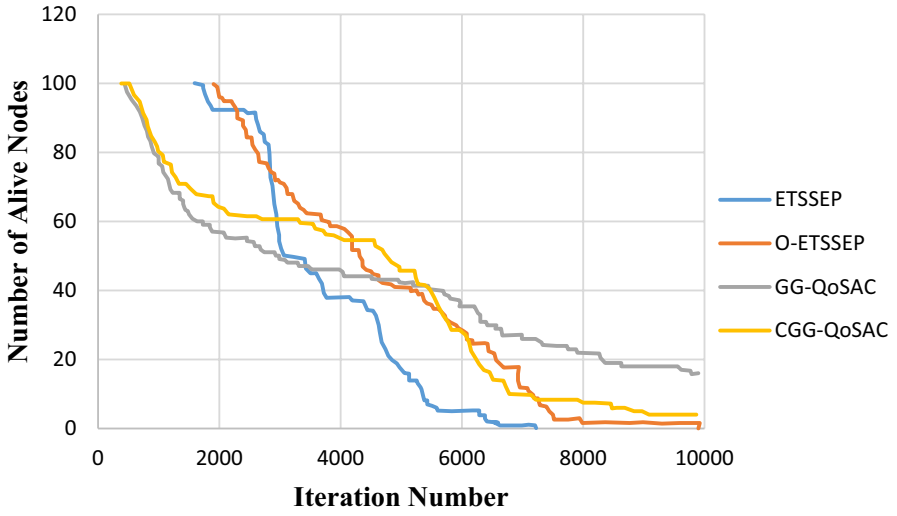


Fig. 17 Comparative analysis of alive nodes versus iteration number of CGG-QoSAC with ETSSEP, GG-QoSAC and O-ETSSEP

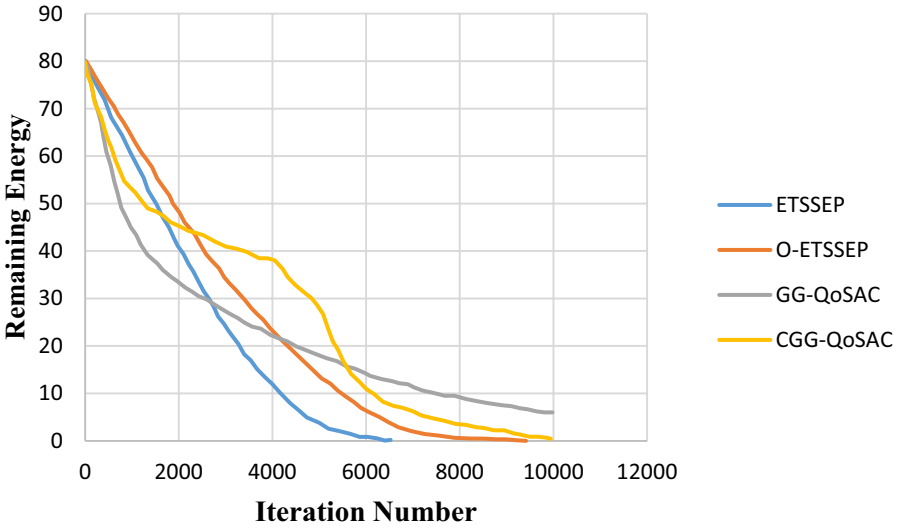


Fig. 18 Comparative analysis of networks remaining energy of CGG-QoSAC with ETSSEP, GG-QoSAC and O-ETSSEP

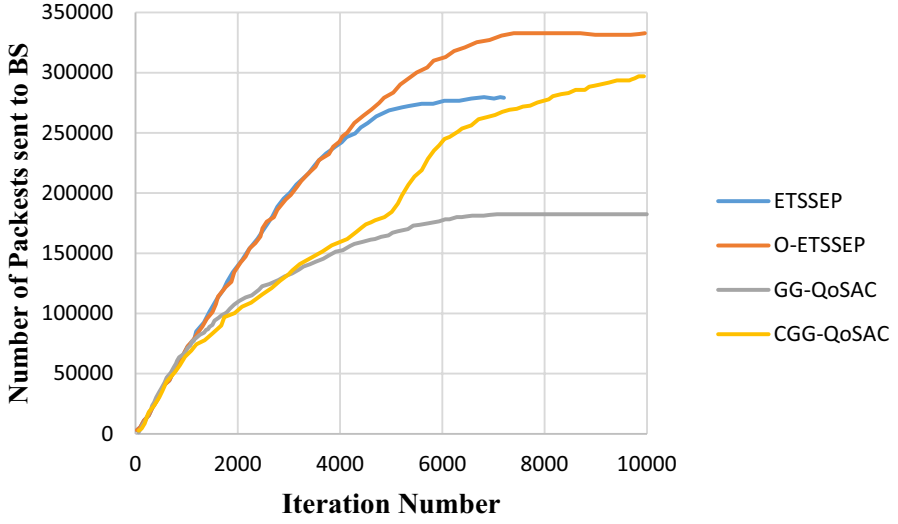


Fig. 19 Throughput comparison of CGG-QoSAC with ETSSEP, GG-QoSAC and O-ETSSEP

in Table 4 and Fig. 17. The number of alive nodes versus iteration number of the CGG-QoSAC algorithm plots in Fig. 17. Of course, as the results show, the stability period of CGG-QoSAC is more extended than GG-QoSAC. The stability period of CGG-QoSAC is lower than ETSSEP, O-ETSSEP because

Table 4 Comparative analysis of the CGG-QoSAC algorithm with the O-ETSSEP protocol

Algorithms	Stability Period (rounds)	Network life (rounds)	Network efficiency (rounds)	Throughput
ETSSEP	1612	7225	3065	2.27×10^5
O-ETSSEP	1934	9948	4317	3.31×10^5
GG-QoSAC	476	15,531	2930	1.82×10^5
CGG-QoSAC	588	10,912	4723	2.91×10^5

while learning the optimal operation, a small number of sensors lose their energy due to the random selection of operations by sensors.

- As presented in Table 4, the last sensor died at 10,912 rounds in the CGG-QoSAC algorithm compared to 7225 and 9948 rounds in the ETSSEP and O-ETSSEP protocols, respectively. CGG-QoSAC performs much better than the ETSSEP and O-ETSSEP protocols in network lifetime. Some sensors do not send packets due to the QoS requirement, and their energy is conserved. It can be noticed from the results that the GG-QoSAC protocol has a more extended network lifetime than the CGG-QoSAC protocol. As shown in Fig. 18, after about 7000 rounds, all the sensors go to sleep mode and do not send any packets in the GG-QoSAC protocol, and this lifetime is practically not beneficial for the network. In the proposed method, after a percentage of the sensors have lost their energy, the sensors in sleep mode change their status and send packets. As a result, the capacity of all sensors has been used until the last moment.
- It is evident from Table 4 that half of the network sensors are dead after 4723 rounds in CGG-QoSAC. The CGG-QoSAC algorithm performs better than other protocols in terms of network efficiency.
- The network remaining energy signifies the depletion rate of the energies of sensor nodes in the network, as shown by Fig. 18. The network's total energy reduces slowly in this algorithm. The network's total energy is 80 J, which reduces at a much slower pace than the other protocols.
- Fig. 19 plots the throughput of the CGG-QoSAC algorithm. Because some sensors do not send packets per round, the throughput of this algorithm is slower than the O-ETSSEP protocol. According to the quality of services required, it is assumed that this throughput is sufficient for the network since the proposed algorithm has reached the number of active sensors required per round. As shown in Fig. 19, the proposed protocol has improved throughput compared to the ETSSEP and GG-QoSAC protocols.

5.3.3 Experiment III

This experiment is conducted to study the QoS provisioning performance of the CGG-based QoS algorithm for multi sinks WSNs. Also, the performance of the CGG-QoSAMS is compared with the MSO-ETSSEP protocol presented in [54] and a protocol that utilizes the original Goore Game [2] independently in each disjoint group of sensors for QoS (GG-QoSAMS). In this experiment, there are

Table 5 Comparison of CGG-QoSAMS algorithm and MSO-ETSSEP protocol

Algorithms	Stability Period (rounds)	Network life (rounds)	Network efficiency (rounds)	Throughput
MSO-ETSSEP	2055	8445	4056	3.48×10^5
GG-QoSAMS	457	16,754	3090	2.14×10^5
CGG-QoSAMS	514	11,013	5163	2.99×10^5

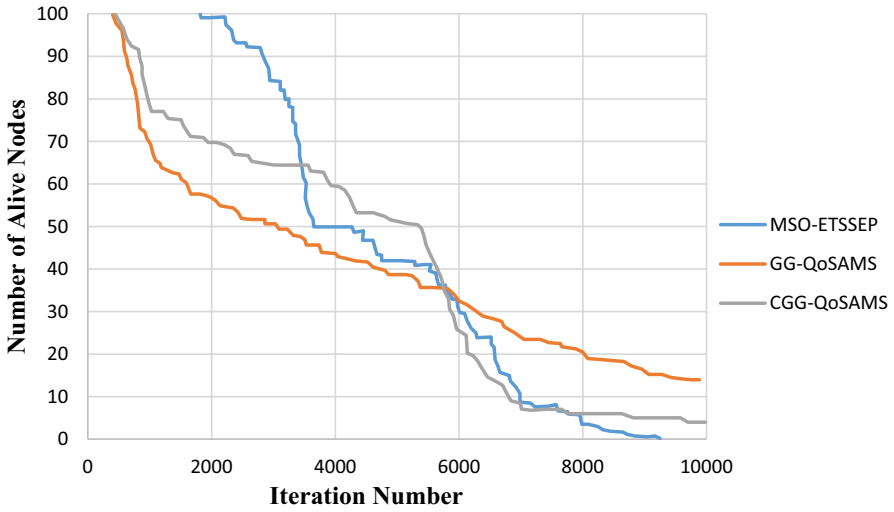


Fig. 20 Comparative analysis of alive nodes versus iteration number of CGG-QoSAMS with GG-QoSAMS and MSO-ETSSEP

four sinks with infinite energy and several sensors with an initial energy of 0.5 in the network. According to this assumption, similar to MSO-ETSSEP protocol, the energy consumed by the sinks is not considered in the network's remaining energy. The simulation parameters defined for the network and the energy values consumed by the nodes are presented in Table 3. From the result of this experiment given in Table 5 and Figs. 20, 21, and 22, we may conclude the following.

- Figure 20 outlines the number of alive nodes versus the iteration number of the CGG-QoSAMS. As shown in Table 5 and Fig. 20, the stability period achieved by this algorithm is 514 rounds which is lower than the MSO-ETSSEP protocol. According to sensors' random selection of operations, few sensors utilize their energy in the elementary rounds.
- From Fig. 21 and Table 5, we observe that the network lifetime of CGG-QoSAMS has enhanced comprehensively compared to the MSO-ETSSEP protocol. Since some sensors are in sleep mode; as a result, their energy is saved, and the network lifetime is significantly improved.

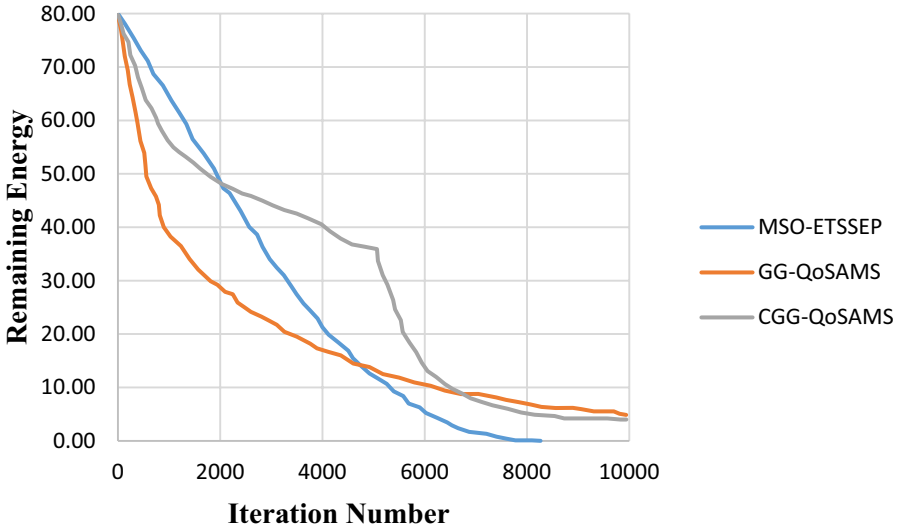


Fig. 21 Comparative analysis of networks remaining energy of CGG-QoSAMS with GG-QoSAMS and MSO-ETSSEP

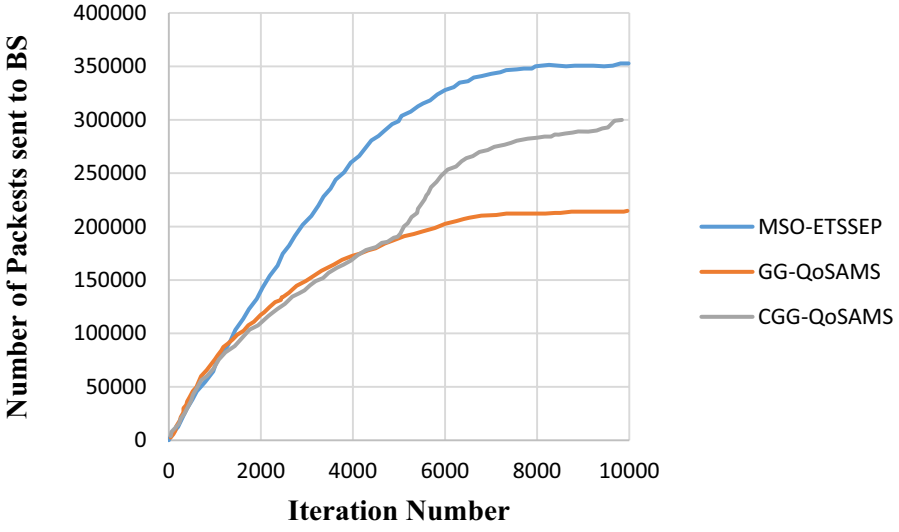


Fig. 22 Throughput comparison of CGG-QoSAMS with GG-QoSAMS and MSO-ETSSEP

- From Table 5, half of the network sensors are dead at (after) 5163 rounds in the CGG-QoSAMS algorithm compared to 40,156 rounds in the MSO-ETSSEP protocol and 3090 rounds in the GG-QoSAMS protocol.
- Fig. 21 shows the network remaining energy in CGG-QoSAMS. As it is evident from Fig. 21, the network's total energy decreases at a low rate, especially in the

final rounds of this algorithm. It can be concluded that the network's total energy is 80 J, which decreases at a much slower pace than the other protocols.

- The throughput of the CGG-QoSAMS algorithm is shown in Fig. 22. The throughput is significantly enhanced in the proposed protocol compared with the GG-QoSAMS protocol. It is observed from Table 5 that the throughput of this algorithm is worse than the MSO-ETSSEP protocol. Network throughput has decreased because some sensors become idle based on the quality of services required in some rounds. However, this throughput is acceptable for the network.

6 Conclusions

This paper proposes a novel model called Cellular Goore Game, and its behavior is studied through several experiments. To show the applicability of CGG, two novel QoS control algorithms for WSNs when clusters have overlap, one for clustered WSNs and one for WSNs with multiple sinks, are designed. The proposed algorithms dynamically adjust the number of active sensor nodes aiming to preserve the QoS parameters. Several computer simulations have been conducted to evaluate the suggested QoS control protocols in ns3. They are compared with QoS protocols proposed in [53, 54] and the original GG-based [2] for QoS control to show the superiority of the suggested algorithms. Experimental results showed that the proposed algorithms in QoS control could compete with existing algorithms in terms of QoS control performance metrics. However, in future, we intend to define new metrics to evaluate the behavior of the CGG and apply the CGG model for more applications. Also, we will theoretically investigate the convergence behavior of CGG and will describe the different types of CGG along with their formal definition.

References

1. Tsetlin ML (1973) Automaton theory and modeling of biological systems, vol 102. Academic Press, New York
2. Thathachar MAL, Arvind MT (1997) Solution of Goore game using modules of stochastic learning automata. J Indian Inst Sci 77(1):47–61
3. Thathachar MAL, Sastry PS (2002) Varieties of learning automata: an overview. IEEE Trans Syst Man Cybern Part B Cybern 32(6):711–722. https://doi.org/10.1109/TSMCB.2002.1049606
4. Chen D, Varshney PK (2004) QoS support in wireless sensor networks: a survey. Int Conf Wirel Netw 233:1–7
5. Narendra K, Thathachar M (2012) Learning automata: an introduction. IEEE Trans Syst Man Cybern B Cybern, vol. 32, no. 6.
6. Tung B, Kleinrock L (1996) Using finite state automata to produce self-optimization and self-control. IEEE Trans Parallel Distrib Syst 7(4):439–448
7. Li S, Ge H, Liang Y-C, Zhao F, Li J (2016) Estimator Goore Game based quality of service control with incomplete information for wireless sensor networks. Signal Process 126:77–86
8. Iyer R and Kleinrock L (2003) "QoS control for sensor networks," in *IEEE International Conference on Communications, 2003. ICC'03*, vol 1, pp 517–521

9. Frolik J (2004) "QoS control for random access wireless sensor networks," in 2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No. 04TH8733), vol 3, pp 1522–1527
10. Nayer SI and Ali HH (2008) "A dynamic energy-aware algorithm for self-optimizing wireless sensor networks," in International Workshop on Self-Organizing Systems, pp 262–268
11. Ayers M and Liang Y (2011) "Gureen Game: An energy-efficient QoS control scheme for wireless sensor networks," in 2011 International Green Computing Conference and Workshops, pp 1–8
12. T. Semprebom, A. R. Pinto, C. Montez, and F. Vasques, "Energy consumption and spatial diversity trade-off in autonomic Wireless Sensor Networks: The (m, k)-Gur Game approach," in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 135–140
13. Elshahed EM, Ramadan RA, Al-Tabbakh SM, El-zahed H (2014) Modified gur game for WSNs QoS control. Proced Comput Sci 32:1168–1173
14. Semprebom T, Montez C, de Araújo GM, and Portugal P (2015) "Skip game: an autonomic approach for QoS and energy management in IEEE 802.15. 4 WSN," in 2015 IEEE Symposium on Computers and Communication (ISCC), 14(2), 1–9, 2014.
15. Oommen BJ, Granmo O-C, Pedersen A (2007) "Using stochastic AI techniques to achieve unbounded resolution in finite player Goore Games and its applications", In: IEEE Symposium on Computational Intelligence and Games, pp 161–167
16. Calitoiu D (2009) "New search algorithm for randomly located objects: A non-cooperative agent based approach," in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp 1–6
17. Yoon B-J (2011) Enhanced stochastic optimization algorithm for finding effective multi-target therapeutics. BMC Bioinformat 12(1):1–11
18. Thathachar MAL, Sastry PS (2004) Networks of learning automata : techniques for online stochastic optimization. Springer, Boston
19. Beigy H and Meybodi MR (2002) "A New Distributed Learning Automata Based Algorithm For Solving Stochastic Shortest Path Problem," in *JCIS*, pp 339–343
20. Anari B, Torkestani JA, Rahmani AM (2017) Automatic data clustering using continuous action-set learning automata and its application in segmentation of images. Appl Soft Comput 51:253–265
21. Wheeldon A, Shafik R, Rahman T, Lei J, Yakovlev A, Granmo O-C (2020) Learning automata based energy-efficient AI hardware design for IoT applications. Philos Trans R Soc A 378(2182):20190593
22. Rahmanian AA, Ghobaei-Arani M, Tofighy S (2018) A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. Futur Gener Comput Syst 79:54–71
23. Akbari Torkestani J, Meybodi MR (2010) An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata. Comput Netw 54(5):826–843. https://doi.org/10.1016/j.comnet.2009.10.007
24. Esnaashari M, Meybodi MR (2011) A cellular learning automata-based deployment strategy for mobile wireless sensor networks. J Parallel Distrib Comput 71(7):988–1001. https://doi.org/10.1016/j.jpdc.2010.10.015
25. Mostafaei H, Meybodi MR (2013) Maximizing lifetime of target coverage in wireless sensor networks using learning automata. Wirel Pers Commun 71(2):1461–1477. https://doi.org/10.1007/s11277-012-0885-y
26. Saghiri AM, Meybodi MR (2018) An adaptive super-peer selection algorithm considering peers capacity utilizing asynchronous dynamic cellular learning automata. Appl Intell 48(2):271–299
27. Rezvanian A, Meybodi MR (2015) Finding minimum vertex covering in stochastic graphs: a learning automata approach. Cybern Syst 46(8):698–727
28. Khomami MMD, Rezvanian A, Meybodi MR (2018) A new cellular learning automata-based algorithm for community detection in complex social networks. J Comput Sci 24:413–426
29. Khomami MMD, Rezvanian A, Bagherpour N, Meybodi MR (2018) Minimum positive influence dominating set and its application in influence maximization: a learning automata approach. Appl Intell 48(3):570–593
30. Khomami MMD, Rezvanian A, Meybodi MR (2016) Distributed learning automata-based algorithm for community detection in complex networks. Int J Mod Phys B 30(8):1650042
31. Rezvanian A, Saghiri AM, Vahidipour SM, Esnaashari M, Meybodi MR (2018) Recent advances in learning automata. Stud Comput Intell 754:1–458. https://doi.org/10.1007/978-3-319-72428-7
32. Norman MF (1968) On the linear model with two absorbing barriers. J Math Psychol 5(2):225–241. https://doi.org/10.1016/0022-2496(68)90073-4
33. Thathachar MAL, Sastry PS (2011) Networks of learning automata: techniques for online stochastic optimization. Springer Science & Business Media, Heidelberg

34. Oommen BJ, Granmo O-C, and Pedersen A (2006) "Empirical verification of a strategy for unbounded resolution in finite player goore games," In Australasian Joint Conference on Artificial Intelligence, pp 1252–1258.
35. Granmo O-C, Oommen BJ, Pedersen A (2012) Achieving unbounded resolution in finite player goore games using stochastic automata, and its applications. Seq Anal 31(2):190–218
36. Granmo O-C, Glimsdal S (2013) Accelerated Bayesian learning for decentralized two-armed bandit based decision making with applications to the Goore game. Appl Intell 38(4):479–488
37. Yaacoub E, Abu-Dayya A, and Matin MA (2012) "Multihop routing for energy efficiency in wireless sensor networks," In Wireless sensor networks-technology and protocols, In Tech Press, pp 165–186, Springer, Berlin, Germany.
38. Shirazi GN, Wang P, Dong X, Eu ZA, and Tham C-K, (2008) "A QoS network architecture for multi-hop, multi-sink target tracking WSNs," In 2008 11th IEEE Singapore International Conference on Communication Systems, pp 17–21.
39. Tang S, Li W (2006) QoS supporting and optimal energy allocation for a cluster based wireless sensor network. Comput Commun 29(13–14):2569–2577
40. Choe HJ, Ghosh P, Das SK (2010) QoS-aware data reporting control in cluster-based wireless sensor networks. Comput Commun 14(2), 1–9, February 2014.
41. Mazaheri MR, Homayounfar B, Mazinani SM (2012) Qos based and energy aware multi-path hierarchical routing algorithm in wsns. Wirel Sens Netw 4(2):31
42. Fapojuwo AO, Cano-Tinoco A (2009) Energy consumption and message delay analysis of QoS enhanced base station controlled dynamic clustering protocol for wireless sensor networks. IEEE Trans Wirel Commun 8(10):5366–5374
43. Singh SK, Kumar P, Singh JP (2017) A survey on successors of LEACH protocol. Ieee Access 5:4298–4328
44. Nazir B, Hasbullah H (2013) Energy efficient and QoS aware routing protocol for clustered wireless sensor network. Comput Electr Eng 39(8):2425–2441
45. Diaz JR, Lloret J, Jimenez JM, Rodrigues JJPC (2014) A QoS-based wireless multimedia sensor cluster protocol. Int J Distrib Sens Netw 10(5):480372
46. Shiva Prakash T, Raja KB, Venugopal KR, Iyengar SS, and Patnaik LM (2014) "Base station controlled adaptive clustering for Qos in wireless sensor networks," Int J Comput Sci Netw Secur 14(2)
47. Hammoudeh M, Newman R (2015) Adaptive routing in wireless sensor networks: QoS optimisation for enhanced application performance. Inf Fusion 22:3–15
48. Deepa O, Suguna J (2020) An optimized QoS-based clustering with multipath routing protocol for wireless sensor networks. J King Saud Univ Inf Sci 32(7):763–774
49. Amjad M, Afzal MK, Umer T, Kim B-S (2017) QoS-aware and heterogeneously clustered routing protocol for wireless sensor networks. IEEE Access 5:10250–10262
50. Hamidouche R, Aliouat Z, Gueroui AM (2018) Genetic algorithm for improving the lifetime and QoS of wireless sensor networks. Wirel Pers Commun 101(4):2313–2348
51. Kaur T, Kumar D (2020) A survey on QoS mechanisms in WSN for computational intelligence based routing protocols. Wirel Netw 26(4):2465–2486
52. Shen H, Bai G, Tang Z, Zhao L (2014) QMOR: QoS-aware multi-sink opportunistic routing for wireless multimedia sensor networks. Wirel Pers Commun 75(2):1307–1330
53. Kumar S, Verma SK, Kumar A (2015) Enhanced threshold sensitive stable election protocol for heterogeneous wireless sensor network. Wirel Pers Commun 85(4):2643–2656
54. Verma S, Sood N, Sharma AK (2019) QoS provisioning-based routing protocols using multiple data sink in IoT-based WSN. Mod Phys Lett A 34(29):1950235
55. Rehan W, Fischer S, Rehan M, Mawad Y, Saleem S (2020) QCM2R: A QoS-aware cross-layered multichannel multisink routing protocol for stream based wireless sensor networks. J Netw Comput Appl 156:102552
56. Knuth DE (2014) Art of computer programming, volume 2: Seminumerical algorithms. Addison-Wesley Professional
57. Barabási A-L, Albert R (1999) Emergence of scaling in random networks. Science (80-) 286(5439):509–512
58. Jain TK, Saini DS, Bhooshan SV (2015) Lifetime optimization of a multiple sink wireless sensor network through energy balancing. J. Sensors 2015:1–6
59. Vincze Z, Vida R, and Vidacs A, (2007) "Deploying multiple sinks in multi-hop wireless sensor networks," In IEEE international conference on pervasive services, pp 55–63.

60. Nsnam, (2011) "Ns-3 a Discrete-Event Network Simulator for Internet Systems," Ns-3, https://www.nsnam.org/ (Accessed Aug. 30, 2021).
61. Rodríguez A, Del-Valle-Soto C, Velázquez R (2020) Energy-efficient clustering routing protocol for wireless sensor networks based on yellow saddle goatfish algorithm. Mathematics 8(9):1515
62. Alazzawi L, Elkateeb A (2008) Performance evaluation of the WSN routing protocols scalability. J Comput Syst Netw Commun 2008:1–9
63. Heinzelman WB, Chandrakasan AP, Balakrishnan H (2002) An application-specific protocol architecture for wireless microsensor networks. IEEE Trans Wirel Commun 1(4):660–670

Authors and Affiliations

Reyhaneh Ameri[1] · Mohammad Reza Meybodi[1] · Mohammad Mehdi Daliri Khomami[1]

Reyhaneh Ameri
r.ameri@aut.ac.ir

Mohammad Mehdi Daliri Khomami
m.daliri@aut.ac.ir

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