The Journal of Supercomputing (2025) 81:834
https://doi.org/10.1007/s11227-025-07249-z

Cellular goore game with multiple learning automata in each cell and its applications

Mohammad Mehdi Daliri Khomami[1] · **Ali Mohammad Saghiri**[1] · **Mohammad Reza Meybodi**[1]

Accepted: 24 March 2025

Abstract

Recently, a model known as cellular goore game (CGG) has been introduced, which represents a hybrid approach combining principles from Cellular Automata theory, Learning Automata from reinforcement learning, and Goore Game from game theory. While this model has garnered attention, it has been noted that each cell in the traditional CGG model is equipped with only a single learning automaton, which may not be suitable for certain real-world scenarios. In this study, we aim to enhance the traditional CGG model by introducing multiple learners within each cell, resulting in the proposed model termed CGG-MLA (Cellular Goore Game with Multiple Learning Automatons). We apply this extended model to address the overlapping community detection problem in multilayer social networks, leveraging its optimization capabilities. Additionally, we investigate the temporal behavior of the CGG-MLA using a solution based on ordinary differential equations (ODEs) and conduct a thorough performance analysis to validate its effectiveness. Our experimental results demonstrate that the CGG-MLA-based approach surpasses alternative algorithms, including Pure-Chance-CGG, GenLouv, mInfomap, PMM, CLACD, and MLCD, in terms of metrics such as normalized mutual information (NMI), modularity, MinMaxCut and Coverage, thus highlighting its efficacy in overlapping community detection within multilayer social networks.

Keywords Cellular automata · Learning automata · Goore Game · Cellular Goore Game · Overlapping community · Multilayer social networks

✉ Mohammad Reza Meybodi
mmeybodi@aut.ac.ir

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

Ali Mohammad Saghiri
saghiri@aut.ac.ir

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

Published online: 06 May 2025

1 Introduction

Cellular automata theory is widely applied in scientific research and simulation as a discrete model. This model consists of simple, identical components known as cells, arranged in a regular grid pattern. Each cell can adopt a state from a finite set of possible states. The operation of a cellular automaton (CA) is governed by a specific local rule, which depends on the cell's immediate surroundings, including itself and its neighboring cells [1]. A configuration represents the global state of a CA, incorporating the states of all constituent cells. CAs adhere to local rules dictating system evolution over time, where interactions between the initial configuration and these rules determine subsequent changes [2]. Notably, CAs are effective for modeling real social systems comprised of vast collections of simple, interacting components. Conceptually, CAs resemble points in a lattice, adhering to a simple local rule and characterized as cellular and automated [3]. This characteristic implies predictable interactions among cells based on their lattice positions, enabling autonomous system operation without centralized control.

However, the learning automaton (LA) serves as a fundamental agent for decision-making in uncertain and stochastic environments. The LA functions as a learning paradigm aimed at selecting an action from its set of possibilities based on feedback from the environment. This model offers the advantage of constructing complex structures using a simple unit, facilitating the design and management of intricate learning tasks [4]. In numerous practical scenarios and applications, local interaction among learning automata (LAs) proves essential. Local interaction involves LAs exchanging information within a defined neighborhood or proximity. Such interaction can be illustrated through various graph structures like trees, meshes, or arrays. Employing a graph-based framework enables LAs to communicate with neighboring automata, allowing for knowledge sharing, action coordination, and collective contribution to decision-making. These local interaction mechanisms enhance the efficiency and effectiveness of the learning automata system, promoting cooperative behavior and optimizing learning outcomes across diverse contexts [5].

Presently, the fusion of cellular automata and learning automata has birthed novel models known as cellular learning automata (CLA). These composite models encompass various criteria, including static and dynamic, open or close, synchronicity or asynchronicity, as well as the configuration of either a singular or multiple automata within each cell. Static cellular learning automata (SCLAs) constitute one class, wherein the cellular structure of the CLA remains constant throughout its evolution. Many CLAs, including those documented in [6], fall into this category. SCLAs can be further categorized as closed or open. In closed SCLAs, the action of each LA relies on neighboring cells, while in open SCLAs, it depends on neighboring cells, a global environment, and an exclusive environment. SCLAs can also be synchronous or asynchronous. In synchronous SCLAs, all cells employ their local rules simultaneously, assuming the existence of an external clock that synchronizes cellular events [7]. Conversely, in an asynchronous SCLA, only some cells are activated at a given time, while the rest

maintain their states unchanged [5]. Additionally, a model of SCLA featuring multiple LAs within each cell was outlined in [8]. Depending on its structure, SCLA can also be classified as regular [2] or irregular [9]. In an Irregular CLA, the assumption of structural regularity is discarded, facilitating its application in solving problems modeled by graphs with irregular structures, as demonstrated in [10]. Dynamic cellular learning automata (DCLAs) form another category, where one or more aspects of the CLA, such as its structure, local rule, or neighborhood, may undergo changes over time. DCLAs can be further divided into closed DCLAs [13, 14] or open DCLAs, based on their operational characteristics. Moreover, DCLAs can be classified as either synchronous or asynchronous. Notably, all documented DCLAs are characterized as closed and asynchronous [11, 12].

The Goore Game has been extensively studied in the field of artificial intelligence, as discussed in [15]. It has been shown to effectively simulate complex decision-making processes and serves as a benchmark for AI research. Results indicate its capability to accurately predict outcomes based on different strategies, providing valuable insights into AI algorithm performance. Conceptually, envision a large room with N cubicles and a raised platform where a Referee oversees voters in each cubicle, akin to a voting system ensuring fair and accurate voting. Each round involves voters independently casting "Yes" or "No" votes, with the Referee calculating the fraction λ representing the proportion of "Yes" votes. To optimize the system, the referee uses a uni-modal performance function $G(\lambda)$ to precisely match the fraction of "Yes" votes to a predetermined value λ^*. Following each round, voters receive or lose a dollar based on probabilities determined by $G(\lambda)$, influencing subsequent voting decisions.

The application of learning processes to cellular networks extends to addressing various real-world issues, particularly those concerning social networks. With this aim, a novel model, the cellular goore game (CGG), is proposed, integrating multiple learning automatons (LAs) within each cell. The motivation for employing cellular goore game (CGG) structures with multiple learning automata (LAs) in each cell originates from the necessity for enhanced computational capabilities and more adaptable decision-making processes. Through the incorporation of multiple LAs within individual cells, CGGs acquire the capacity to manage complex information more effectively. This configuration enables the simultaneous consideration of diverse factors and strategies, thereby fostering the development of sophisticated and flexible behaviors. Additionally, the interaction among multiple LAs within the same cell promotes collaborative decision-making and enhances the utilization of available information resources. Ultimately, the integration of CGGs with multiple LAs per cell aims to achieve superior performance and resilience in tackling complex problems compared to CGGs equipped with only a single LA per cell. This paper focuses on investigating the behavior of the CGG with multiple LAs in each cell, particularly its steady-state dynamics. The analysis reveals that, under certain conditions, the CGG converges toward a globally stable state. Furthermore, an illustrative application of this innovative model in the context of multilayer overlapping community detection within social networks is presented.

In summary, our contribution can be summarized as follows:

- The cellular goore game with multiple learning automata (CGG-MLA) represents a recent advancement in modeling complex environments, such as social and information networks.
- It extends the naive cellular goore game (CGG) by incorporating multiple learning automata within each cell, providing applicability and adaptivity to the model for solving complex network problems.
- The CGG-MLA opens a new learning paradigm for solving many real problems in online, non-deterministic, dynamic, distributed or decentralized environments such as online social networks, cloud computing, internet of things and Blockchain technologies.
- The convergence behavior of the CGG-MLA model confirms that the algorithm converges to near optimal solution.
- From an application perspective, the proposed model is applied to the problem of detecting multilayer overlapping communities.

The remainder of this paper is organized as follows: Sect. 2 provides an overview of related works. Section 3 briefly explains the concept of learning automata. In Sect. 4, we introduce the definition of Cellular Goore Game with multiple LAs in each cell (CGG-MLA) and discuss its steady-state behavior, convergence behavior, and complexity analysis, respectively. Section 5 defines the application of the CGG with multiple LAs along with some requirements. Section 6 presents the proposed CGG-based algorithm for finding overlapping communities in multilayer networks. Section 8 outlines the evaluation metrics for CGG-MLA. Numerical results are presented in Sect. 9, and the paper concludes in Sect. 10.

2 Related works

This section provides a short overview of three aspects: first, recent applications of the Cellular Goore Game; second, a learning model based on CLA; and third, the application of overlapping community detection in multilayer social networks.

Recently, the cellular goore game (CGG) model has been studied extensively across various domains, leading to the development of numerous algorithms based on its framework. In [16], the cellular goore game (CGG), a novel extension of the Goore Game (GG), was introduced to model network problems in domains such as computer networks, grid computing, and social networks. Unlike GG, CGG forms a network of GGs where each node acts as a referee, participating in local GGs with its neighboring players. To showcase the model's potential, the CGG-Clique algorithm was proposed for solving the maximum clique problem in networks. In this algorithm, cells independently select optimal actions based on gains and losses from adjacent referees. The complexity of the CGG-Clique algorithm was analyzed, demonstrating its scalability. Extensive experiments on the DIMACS benchmark validated the algorithm's effectiveness by comparing it with seven existing methods, highlighting its superiority in efficiently identifying larger cliques. Future work includes introducing new evaluation metrics for CGG behavior and applying the model to identify maximum cliques in multilayer social

networks. Ameri et al. [17] studied the CGG model through extensive experiments and demonstrated its applicability by developing two QoS control algorithms for wireless sensor networks (WSNs) with overlapping clusters: one for clustered WSNs and another for WSNs with multiple sinks. These algorithms dynamically regulate the number of active sensor nodes to maintain QoS parameters effectively. The proposed methods were evaluated using ns3 simulations and compared with existing QoS control protocols, including those based on Goore Game models and other approaches. The results indicated that the proposed algorithms outperformed or were comparable to existing methods in terms of QoS performance metrics. In [18], the cellular goore game consensus protocol (CGG-CP) was extended to propose an enhanced consensus algorithm, ICGG-CP, aimed at improving fault tolerance and reducing communication complexity. This protocol retains the three primary roles of CGG-CP—referee, player, and owner—but introduces mechanisms to address owner and referee failures. A secondary owner is selected among referees to ensure operational continuity in the event of owner failure, and referee fault tolerance is improved through a reporting mechanism for referee misconduct. To enhance block validation, ICGG-CP uses a weighted average reward system, prioritizing nodes with higher credibility. Additionally, the protocol reduces communication costs by allowing players to inform referees of action convergence, avoiding redundant communication. The study provides a theoretical analysis of the CGG's convergence behavior and demonstrates its effectiveness through comprehensive experiments, which show significant improvements in fault tolerance and communication efficiency. Future work includes methods to accelerate CGG convergence, further reducing communication costs and increasing throughput. The integration of AI techniques, such as outlier analysis and clustering, is also planned to enhance the cognitive engines of the consensus algorithm. *Khomami* et al. [19] introduced six CGG-based algorithms utilizing learning automata to estimate network measures specific to stochastic multilayer graphs. These algorithms effectively model user behavior dynamics in multilayer networks, offering more accurate representations of relationship complexities and behavioral uncertainties. They also analyze how user behavior influences network measures, enhancing the understanding of social network structures. The proposed framework shows significant potential for applications in online social networks, such as modeling human activity and user behavior. Future research aims to extend these measures and algorithms to broader applications, including multilayer network sampling and identifying influential nodes for viral marketing. In [20], a novel stochastic multilayer shortest path algorithm was proposed, addressing the complexity of analyzing multilayer graphs with stochastic elements. Built upon the CGG framework and leveraging reinforcement learning with learning automata, the algorithm efficiently determines the shortest path between nodes in stochastic multilayer graphs—a task not previously explored with this approach. Each graph cell is equipped with a learning automaton and a referee, which collaborate to optimize the learning process. The automaton performs two actions—designating membership or non-membership in the shortest path—to identify candidate paths. A referee evaluates these paths by calculating their length and expected value, providing reinforcement signals to guide the automaton toward the optimal solution. The algorithm has demonstrated effectiveness across

various types of stochastic multilayer graphs, making it applicable to domains such as biology, economics, engineering, and social networks. Its adaptive learning capabilities and ability to function in dynamic environments highlight its potential for advancing our understanding of complex systems and their interdependencies.

A learning automaton (LA) is a simple decision-maker designed to learn actions that yield higher rewards. There is a wide array of networks of learning automata (LAs) documented in the literature, including DLA [21], eDLA [22], WCLA [23], Adaptive Petri Net [24], and CLA [2], among other structures highlighted in [25]. These publications focus on diverse designs for networks of LAs. We have thoroughly studied all these to underscore the unique stance of our model. The CLA has previously been introduced as a hybrid model in [1] because it encompasses a specific structure of LA that emphasizes the local interactions of cells—a trait common to some cellular automata (CA) systems—and operates under a local rule derived from the theories of LAs and CAs. For more clarification, in the revised version for the paper, Fig. 1 give a classification on CLAs that has been reported in the literature.

Cellular learning automata (CLAs) emerge from the fusion of Cellular Automata (CAs) and learning automata (LAs) [2]. CAs are models composed of numerous cells arranged in a lattice structure. In a CA, each cell adopts a state from a finite set of states. The subsequent state of each cell is determined by the prior states of a

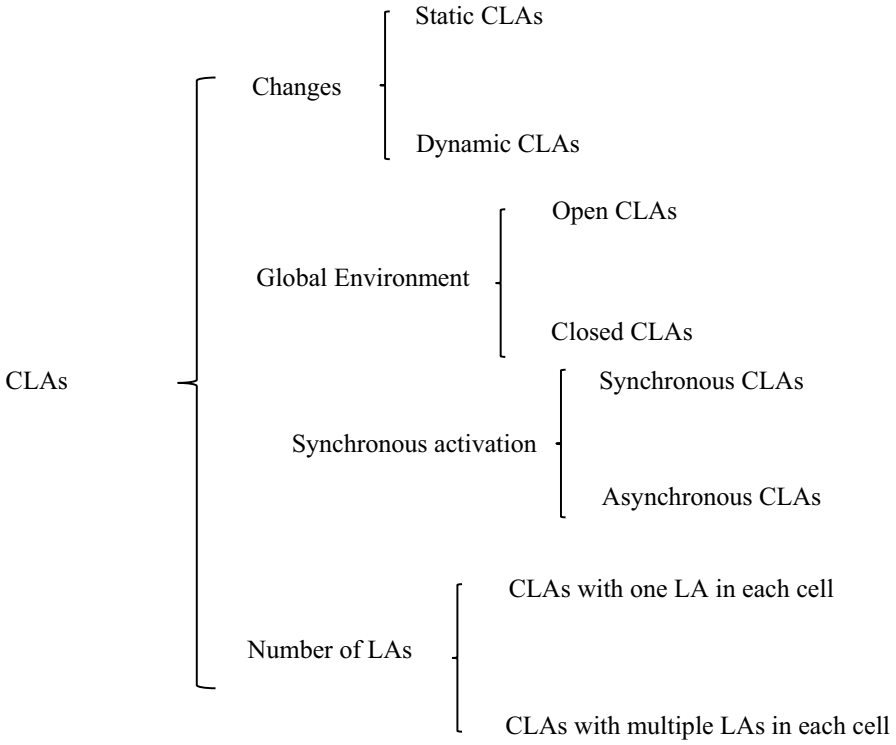


Fig. 1 A classification of cellular learning automata

designated set of cells, encompassing the cell itself and its neighboring cells [26]. In a CLA, every cell within the CA is endowed with a learning automaton to govern its state selection. Within the CLA framework, a local rule, termed as the local rule, dictates the response of each cell's learning automaton. Each cell employs this local rule to elicit the response of its learning automaton, taking into account information pertaining to itself and its neighboring cells. Reported models concerning cellular learning automata (CLAs) can be categorized into two distinct classes, as elucidated as follows:

Cellular learning automata (CLAs) with one LA in each cell, constitute a distinct class, wherein the cellular structure of the CLA comprises one LA per cell. Many CLAs, including those documented in [10, 27–29], fall into this category. CLAs with one LA can be further categorized as closed or open. In closed CLAs, the action of each LA relies on neighboring cells, while in open CLAs, it depends on neighboring cells, a global environment, and an exclusive environment. CLAs with one LA can also be synchronous or asynchronous. In synchronous CLAs, all cells employ their local rules simultaneously, assuming the existence of an external clock that synchronizes cellular events [1]. Conversely, in an asynchronous CLA, only some cells are activated at a given time, while the rest maintain their states unchanged [5]. In an Irregular CLA with one LA in each cell, the assumption of structural regularity is discarded, facilitating its application in solving problems modeled by graphs with irregular structures, as demonstrated in [33]. Dynamic cellular learning automata (DCLAs) form another variation in which using one LA in each cell, where one or more aspects of the CLA, such as its structure, local rule, or neighborhood, may undergo changes over time. DCLAs can be further divided into closed DCLAs [30, 31] or open DCLAs, based on their operational characteristics. Moreover, DCLAs can be classified as either synchronous or asynchronous. Notably, all documented DCLAs are characterized as closed and asynchronous [30, 32].

A model of CLA featuring multiple LAs within each cell was outlined in [8]. Depending on its structure, CLA with multiple LAs in each cell can also be classified as regular or irregular [33, 34]. Cellular learning automata (CLA) with multiple learning automata (LA) in each cell represent a variation where multiple LA entities coexist within a single cell. This approach allows for more complex decision-making processes and enhanced adaptability within cellular networks. With multiple LAs, each cell can simultaneously consider and evaluate multiple actions, leading to a richer exploration of the solution space. The interactions between multiple LAs within a cell can introduce dynamics and emergent behaviors that are not present in CLAs with a single LA per cell. This approach has been explored in various contexts, including optimization, network routing, and pattern recognition, where the parallel decision-making capabilities of multiple LAs can lead to more efficient and effective solutions.

In recent years, notable progress has been achieved in learning models and the detection of multilayer overlapping communities. The identification of communities within intricate social networks has garnered substantial scholarly attention owing to its diverse applications. Traditional algorithms like modularity optimization, spectral clustering, or hierarchical clustering can be extended to handle overlapping communities in multilayer networks. Evaluation metrics such as modularity,

normalized mutual information (NMI), or F1 score are used to assess the quality of detected communities, comparing them with known ground truth or evaluating internal cohesion and external separation. Post-processing and visualization techniques refine identified communities and remove potential noise, aiding in interpreting and visualizing resulting overlapping communities in multilayer social networks. Mucha et al. [35] introduce Infomap to detect overlapping communities in multilayer networks, revealing dynamic patterns of community organization over time. Jia et al. [36] proposed an algorithm that utilizes trust relationships to transform multilayer social networks into a single-layer trust network by merging overlapping communities. Their experimental results demonstrated improvements in modularity and harmonic mean. In [37], an algorithm employing non-negative matrix tri-factorization is presented to detect communities in multiple heterogeneous networks. This method leverages alignment matrices, yielding enhanced performance on datasets such as Twitter and Instagram. Wang et al. [38] introduced a community detection method for Social Internet of Things (IoT) systems, addressing multiple relationship layers. Their algorithm aggregates communities, resulting in improved success rates in experiments. Similarly, Rani et al. [39] proposed a community detection approach for Social IoT systems, focusing on multilayer relationships. The algorithm aggregates communities and enhances experimental success rates. In [40], a hierarchical algorithm for overlapping community detection using weighted aggregation is developed, achieving higher accuracy and reduced computational costs. Finally, Zhang et al. [41] applied matrix tri-factorization to detect communities in multiplex networks, effectively identifying coherent communities across layers. Huang et al. [42] offers an exemplary review of community detection techniques in multilayer social networks and their corresponding applications. They begin by discussing various formats of multilayer networks and introduces two fundamental mathematical models. Then, it reviews quality evaluation measures and several common community detection algorithms, such as label propagation-based algorithms, non-negative matrix factorization, random walk methods, and multi-objective optimization methods. The analysis suggests that existing methods are primarily designed for multiplex networks, where nodes in each layer are aligned, limiting research on more universal multilayer network formats. Additionally, the algorithms tend to have high computational complexity and struggle to produce meaningful partitions in large-scale multilayer networks. They also explore applications of multilayer community detection in diverse fields such as social networks, biological networks, and transportation networks. In [43] proposes a novel community detection method tailored for multilayer networks, which considers both intra-layer and inter-layer similarities to construct a dendrogram capturing the network's unique topology and interplay. By introducing a new community density metric, the dendrogram is partitioned to identify overlapping communities accurately. Experimental results on synthetic and real-world datasets demonstrate the effectiveness of the proposed method in discovering overlapping communities in multilayer networks. Huang et al. [44] discuss the limitations of existing approaches for detecting overlapping community structures in multi-view networks and introduces a novel method called oComm. This approach incorporates a community membership strength vector for each node

in each view to model overlapping community structures and utilizes a network generative model to measure within-view community quality. Additionally, a cross-view community consistency model is established using Jaccard similarity to assess consistency across different views. By solving an objective function through the alternative coordinate gradient ascent method, optimal community membership vectors are generated, allowing for the identification of multi-view overlapping community structures. The effectiveness of the proposed method is demonstrated through comparisons with existing approaches using EEG data from 147 subjects, highlighting its superior performance. In [45] discusses the challenges in community detection within real-life networks, particularly weighted temporal text networks, focusing on modeling edge weights and leveraging temporal information. They stated that existing methods often overlook edge weights or handle them using graph measures like modularity, which lacks scalability. Moreover, temporal information is commonly handled through time discretization, leading to various issues. To address these challenges, the custom temporal community detection (CTCD) method is proposed, which incorporates a probabilistic generative model to encode edge weight and temporal information. CTCD utilizes network, semantic, and temporal data to extract temporal community affiliations, influence strength, and temporal interests. The method offers efficient inference and parallel implementation to handle large datasets, enabling the identification of individual user community shifts and tracking community development over time. Experimental results on large-scale weighted temporal text networks demonstrate CTCD's significant improvement over existing methods across various tasks. Gupta et al. [46] investigated the importance of identifying community structures within growing networks across various domains, emphasizing the need for robust community detection algorithms to reveal disjoint, overlapping, and nested community patterns. To address this, the paper introduces an algorithm for overlapping community detection leveraging granular link information and principles from rough set theory. The algorithm begins by utilizing neighborhood links to create initial link subsets and iteratively computes constrained linkage upper approximations until convergence. These upper approximation subsets are then constrained and merged based on mutual link reciprocity. Experimental results on ten real-world networks validate the effectiveness of the proposed algorithm, as demonstrated through comparative evaluations against state-of-the-art community detection methods. In [47] discusses the complexity of multidimensional networks and the challenges in detecting overlapping communities within them. With the aid of existing methods in which overlook inter-layer interactions and unique topological structures, focusing solely on non-overlapping communities. To address this gap, they proposed a novel approach leverages edge behavior within and between layers to calculate similarities, which are then used to construct a dendrogram capturing both structural and transactional aspects. By introducing a new community density metric, overlapping communities are identified effectively. In [36] introduces a novel algorithm for detecting overlapping communities in multilayer social networks, addressing the issue of lacking user social attribute characteristics during the process. The proposed algorithm utilizes trust relationships by integrating structural trust and social attribute trust, thereby converting the multilayer social network into a single-layer

trust network. Community structure is then obtained using a trust-based community discovery algorithm, followed by the merging of communities with higher overlap. Experimental evaluations conducted on synthetic and real networks demonstrate the algorithm's superior performance in terms of harmonic mean and modularity compared to other algorithms of similar nature. In [48] suggests the significance of multilayer complex networks in representing heterogeneous systems, emphasizing their relevance in the era of big data. It highlights the impact of inter-layer connections on community structures, making community detection in multilayer networks a crucial research area in data mining. They introduce a modified particle competition model tailored for multilayer network community detection, which can handle weighted and directed networks. Additionally, a localized measure is proposed to determine the optimal number of particles for detecting communities. Computer simulations demonstrate the superior performance of the proposed technique compared to existing methods. In [49] proposes an efficient algorithm for Markov Clustering (MCL) based on the principles of belief propagation. It introduces a novel approach to improve the performance of traditional MCL algorithms by incorporating belief dynamics. The method involves iteratively updating the belief of nodes in the network, leading to faster convergence and better clustering results compared to standard MCL algorithms. They demonstrate the effectiveness of the proposed approach through experimental evaluation on various datasets, showing significant improvements in both speed and clustering accuracy. In [50] explores the dynamics of the Prisoner's Dilemma game on signed networks, drawing from the principles of structural balance theory. It investigates how the evolution of cooperation unfolds in social networks characterized by positive and negative links. Through computational simulations and analysis, the study delves into the impact of various network structures and dynamics on the emergence and sustainability of cooperative behavior among individuals. In [51] presents a method for characterizing fuzzy community structures within link graphs by optimizing likelihood. It introduces a novel approach that leverages the concept of likelihood optimization to identify fuzzy community memberships of nodes in a network. By optimizing a likelihood function, the method assigns soft memberships to nodes, allowing them to belong to multiple communities simultaneously with varying degrees of membership strength. The paper demonstrates the effectiveness of the proposed method through experiments on various real-world networks, showcasing its ability to accurately capture the fuzzy nature of community structures in link graphs.

3 Learning automata

The concept of learning automata (LA) revolves around the process of randomly selecting an action from a finite set of actions and subsequently executing that selected action within an unknown and stochastic environment. To accomplish this, a probability distribution is employed to randomly choose an action from the given set, which is then utilized as input within the random environment. As a result of the executed action, the environment generates a reinforcement signal in response. The LA then proceeds to

update its internal state based on the selected action and the received signal, ultimately determining the subsequent action it should undertake.

The primary objective of a learning automaton is to choose the appropriate action from a set of actions in order to minimize the average penalty it receives from the environment. The learning automaton's environment is described as a triple $E = \{\alpha, \beta, c\}$, where $\alpha = \{\alpha_1, \ldots, \alpha_r\}$ denotes the set of inputs (action sets), $\beta = \{\beta_1, \ldots, \beta_m\}$ represents the set of possible output values for the reinforcement signal, and $c = \{c_1, \ldots ., c_m\}$ signifies the set of penalty probabilities determined by the environment response. c_i represents the penalty probability for action α_i . The environment can be categorized into three distinct types, namely P-model, Q-model, and S-model, based on the characteristics of the reinforcement signal. In a P-model, the reinforcement signal exhibits binary behavior with values limited to 0 and 1. In a Q-model, the reinforcement signal is capable of assuming a finite range of values between 0 and 1. In an S-model, the reinforcement signal is characterized as a continuous random variable that spans the interval [0, 1]. If the probabilities of penalties change with time, the random environment is categorized as nonstationary; otherwise, it is classified as stationary. Figure 2 shows the dynamic interplay between the learning automaton and its corresponding environment. It visually captures the intricate relationship and information flow between these two entities throughout the learning process. By analyzing Fig. 2, one can gain a comprehensive understanding of how the learning automaton adapts and responds to the environmental cues and feedback it receives.

The quadruple $\langle \alpha, \beta, \mathcal{I}, X(t) \rangle$ is defined as a representation of the Learning Automata (LAs), where α indicates a finite set of actions denoted by $\alpha = \{\alpha_1. \ldots .\alpha_r\}$, β stands for the set of all possible inputs or reinforcements signal for the LAs, $\mathcal{J}$ represents the learning algorithm responsible for updating the action probabilities, and $X(t)$ describes the action probability vector at a given time t. The action probability vector is adjusted by the learning algorithm through the use of a recurrence relation.

$$\begin{aligned} X_i(t+1) &= X_i(t) + a\big(1 - X_i(t)\big) \\ X_j(t+1) &= X_j(t) - aX_j(t) \forall j \neq i \end{aligned} \tag{1}$$

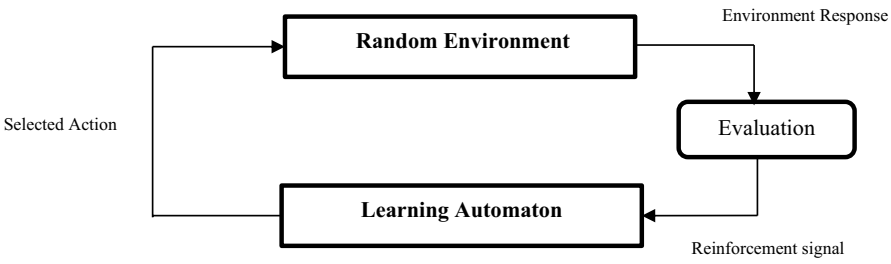


Fig. 2 Learning automata

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

At instance t, a learning automaton selects action α_i from distribution $X(t)$. In a linear learning algorithm, Eq. (1) determines the updating of probability vector $X(t)$ for a favorable response ($\beta = 1$), and Eq. (2) for an unfavorable response ($\beta = 0$). The probability of receiving an unfavorable response for action α_i is represented by $c_i = \Pr[\beta = 0|\alpha_i]$.

The probabilities of receiving a favorable or unfavorable response by the LA are not known. The parameter a determines the extent to which the action probability vector increases, while the parameter b determines the extent to which it decreases. Equations (1) and (2) are referred to as (L_{R-P}) when a equals b, ($L_{R-\varepsilon P}$) when a is much greater than b, and L_{R-I} when b is equal to zero. In L_{R-I}, the action probability vectors remain unchanged despite penalization by the environment. Figure 3 provides the pseudocode of a learning automata and its policy in algorithmic shape.

Learning has become a vital optimization tool in complex and dynamic environments with significant uncertainty or limited environmental knowledge. Numerous instances of successful integration of learning automata can be found across various domains, including graph problems [52], Petri net[53], image processing[54], Internet of Things [55], cloud computing [56], wireless sensor networks[57], p2p networks[58] and complex networks[59].

Input: Action set α
Output: Action probability vector
Assumption
Initialize r-dimensional action set $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ with r actions.
Initialize r-dimensional action probability vector $p(t) = \{p_1, p_2, \ldots, p_r\} = \{\frac{1}{r}, \frac{1}{r}, \ldots, \frac{1}{r}\}$ at instance t
Let i denotes the selected action by the automaton
Let j denotes the current checking action
Begin
Repeat
The learning automata selects an action based on the action probability vector p.
The environment evaluates the selected action and gives the reinforcement signal $\beta \in \{0 \text{ or } 1\}$ to the learning automata
Update the probability vector $p(t)$
For each action $j \in \{1, \ldots, r\}$ **Do**
IF ($\beta = 0$)
The selected action by learning automaton is rewarded based on equation (1)
Else IF $\beta = 1$)
The selected action by learning automaton is penalized based on equation (2)
End IF
End For
Until (automaton does not converge to action)
End Algorithm

Fig. 3 Pseudocode of a learning automaton and its policy

4 Cellular Goore Game with multiple LAs (CGG-MLA)

This section starts by defining the Cellular Goore Game with Multiple Learning Automata (CGG-MLA). It then proceeds to explain the updating process of CGG, followed by a description of two evolution metrics used to analyze this process. Moreover, Table 1 provides notations applied for CGG-MLA.

4.1 The Definition of CGG with multiple LAs

The CGG with multiple LAs in each cell is network of GG in which each cell comprises several LAs. This model can be formally defined by 6 tuples as follows:

Definition 1 A cellular goore game(CGG) with multiple LAs can be defined by 6-tuple.

Table 1 Table of symbols and abbreviations

n_{l_i}	The number of LAs that is adjacent to $cell_i$ and select "Yes" action
α	Action Set
c	penalty probabilities
N	Input network of Cellular Goore Game
V	Represent the set of vertices
E	The set of edges
P	The set of player
R	The set of referee
A	The set of learning automata
F	The set of mapping function
G	The set of uni-modal performance function
$x_{i1}(t)$	The probability of selection of first action by learning automaton i at instance t
$\mathcal{M}$	Multilayer network
$\mathcal{G}$	Set of networks
$\mathcal{C}$	Set of edges
L	Number of layer
β	Reinforcement signal
G_i	Uni-modal performance function
λ_i^l	The function that counts number of LAs that select the first action a_{i1}^l
Q^M	Multilayer modularity
NMI	Normalized Mutual Information
$\theta_i^l(t)$	The density of community i at iteration t
$\zeta_i^l(t)$	The sparsity of community i at iteration t
$K_{i,l}^{in}(g)$	The input degree of node i in layer l to community g
$K_{i,l}^{out}(g)$	The output degree of node i in layer l to community g
F	The set of mapping function where F_i is associated with $cell_i$

CGG = (*N, P, R, A, F, G),* where.

I. The input network, denoted as $N = (V, E)$, represents the structure of CGG. Here, V refers to a set of vertices represented by $\{cell_1.cell_2. \ldots .cell_n\}$, and E represents the set of edges.
II. $P = \{P_1, \ldots, P_n\}$ where P_i is a subset of players assigned to cell_i.
III. $R = \{R_1, \ldots, R_n\}$ comprises as a subset of V playing the role of referees.
IV. $A = \{A_1, A_2, \ldots A_n\}$ is a set of LAs where $A_i = \{LA_{ij} | \{i,j\} \in E\}$ is the set of learning automaton residing in cell_i and each learning automaton has two actions "Yes" or "No".
V. $\text{F} = \{\text{F}_1, \text{F}_2, \ldots \text{F}_\text{n}\}$ is the set of mapping function where F_i is associated with cell_i. The function F_i takes the set of index of neighbors and assign its corresponding learning automata in that cell. For example, if $cell_j$ is the neighbor of the $cell_i$ then LA_{ij} assign to cell_i.
VI. $G = \left(G_1, G_2 \ldots G_n\right)$ determine a set of unimodal performance criteria for the referees, where G_i is the performance criterion for $cell_i$.

A Cellular Goore Game with multiple LAs is a connected structure in which each cell contains multiple LAs and each LA has two action within each cell as similar as GG. In this model each cell contains a set of learning automata. Moreover, each cell has a referee to evaluate the selected action by LA's. The number of LAs which are resided in a cell is equal to the number of neighbors that cell.

Figure 4 illustrates the system configuration of the Cellular Goore Game with multiple learning automata (LAs) in each cell, focusing on the microscopic perspective of cell_1 and its neighborhood structure. In this model each LAs may be activated synchronously or asynchronously. In contrary to original GG, exist a mapping function F to generate LAs for a cell and each referee has its own players to playing GG. Let $cell_j$ is adjacent to $cell_i$, then the mapping function F takes a learning automaton and return a learning automaton LA_{ij} in cell i and a learning automaton LA_{ji} in cell j simultaneously. Based on the output of the mapping function F a set learning automaton is obtained for each cell. Figure 3 shows the initial local structure of cell_1 and its neighborhood cells, which contains a set of learning automata and a referee in each cell. The operation of CGG with multiple LAs takes place in a discrete time steps. Let α_{i1} and α_{i2} are the first and second actions of automaton LA_{ij}, respectively. Moreover, $p_{ij}(t) = \left[p_{i1}(t), p_{i2}(t)\right]$ is the probability of first and second action of LA_{ij}. At round t of the game, each LA_{ij} selects one of its action $\alpha_{ij}(t)$ from two available actions. After the action selection, each LAs send the selected action to its corresponding referee concurrently. Let LA_{ij} is a learning automaton that resided in cell_i, after the action selection is done, the learning automaton send its selected action for the referee R_j. The referee R_j receives the selected actions from all current neighboring cell and computes the fraction of LAs that select its first action to the total number of LAs participate in GG for that cell. Let λ_j is the fraction of LAs that select its first action. Then, based on the value of performance criteria $G_j\left(\lambda_j\right)$ referee R_j, each LAs generates a random number in order to reward or penalized of

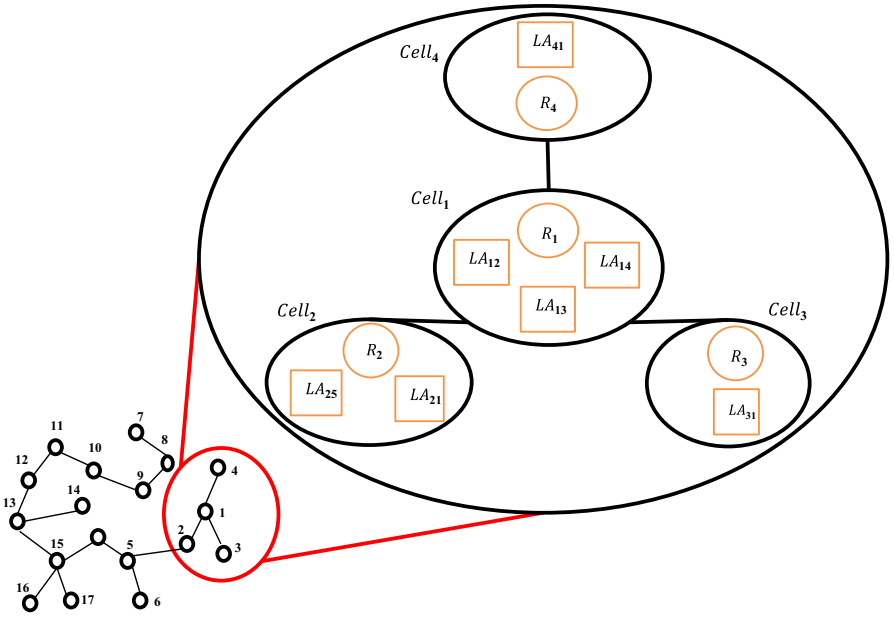


Fig. 4 System model of Cellular Goore Game with multiple LAs in each cell by relying on the microscopic view of the cell_1 and its neighborhood structure

selected action based on learning algorithm. Based on the performance criteria, if each of the learning automata (LAs) selects the first action and the generated random number is less than the performance criteria value, then the first actions of all LAs are rewarded. Likewise, if the selected action is the second one and the random number is less than the performance criteria value, then the second actions of all LAs are rewarded. Figure 5 shows the action selection by LAs that associated with cell_1 and generate reinforcement signal by referee R_1.

Remark 1: The number of LAs resided in each cell corresponds to its direct neighbor degree.

For enhanced clarity, Figure 6 illustrates the flowchart depicting the operation of CGG with multiple LAs.

4.2 Convergence behavior of CGG with multiple LAs

In this section, we assume that the following assumptions hold for each $cell_i$ of the CGG-MLA. Subsequently, we delve into the behavior of the CGG in the remainder of this section.

Definition 1 The configuration of the CGG-MLA with multiple LAs at iteration t is denoted by $\underline{X}(t) = \big(X_1(t), X_2(t), \ldots, X_n(t)\big)$ where $X_i(t)$ represents the action probability vector of learning automaton A_i.

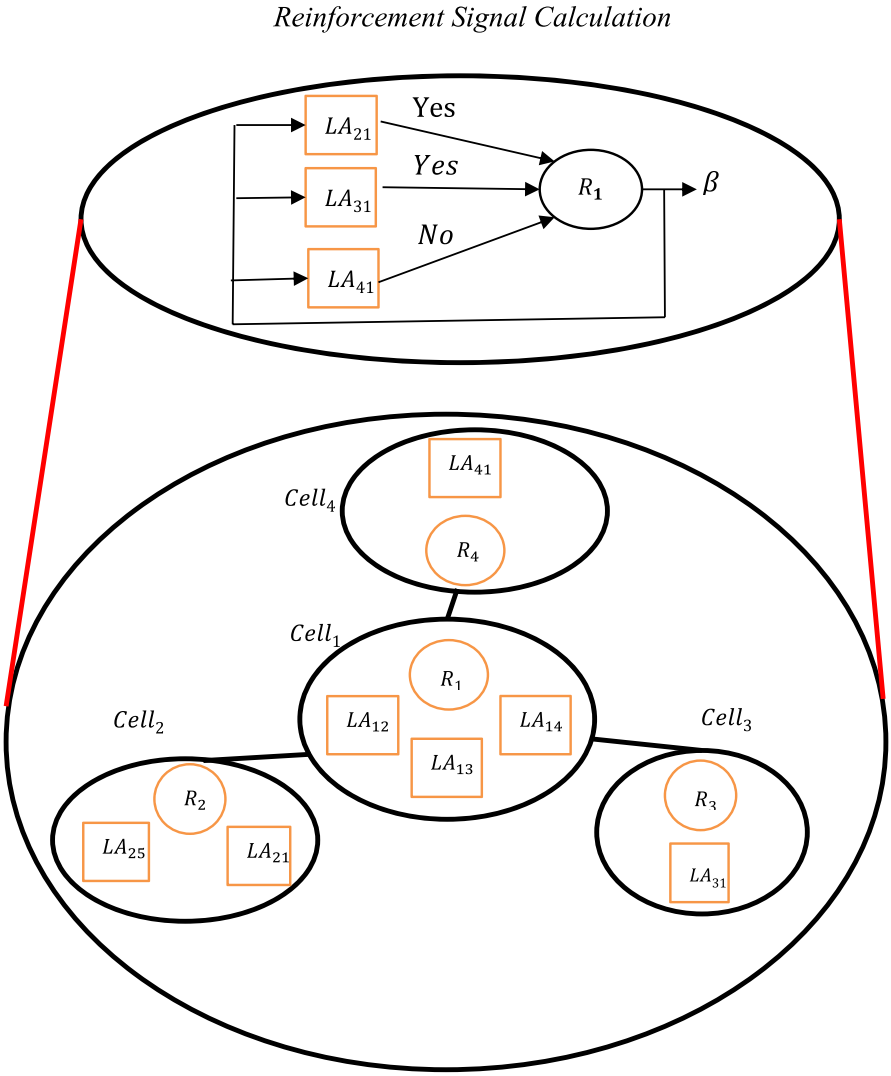


Fig. 5 The structure of Cellular Goore Game with Multiple LAs (CGG-MLA)

Definition 2 A configuration $\underline{X}$ is referred to as deterministic when the action probability vector of all LAs forms a unit vector. Conversely, when the action probability vector deviates from being a unit vector, it is referred to as probabilistic. Hence, we define two sets: $\mathcal{K}^*$, representing the set of all deterministic configurations, and $\mathcal{K}$, representing the set of probabilistic configurations. $\mathcal{K}^*$ is defined as follows:$\mathcal{K}^* = \left\{\underline{X} | x_{iy} \in \{0 \cdot 1\} \forall y \cdot i\right\}$. On the other hand, $\mathcal{K}$ is defined as: $\mathcal{K} = \left\{\underline{X} | x_{iy} \in [0 \cdot 1] \forall y \cdot i\right\}$, where $\sum_{y=1}^{2} x_{iy} = 1$.

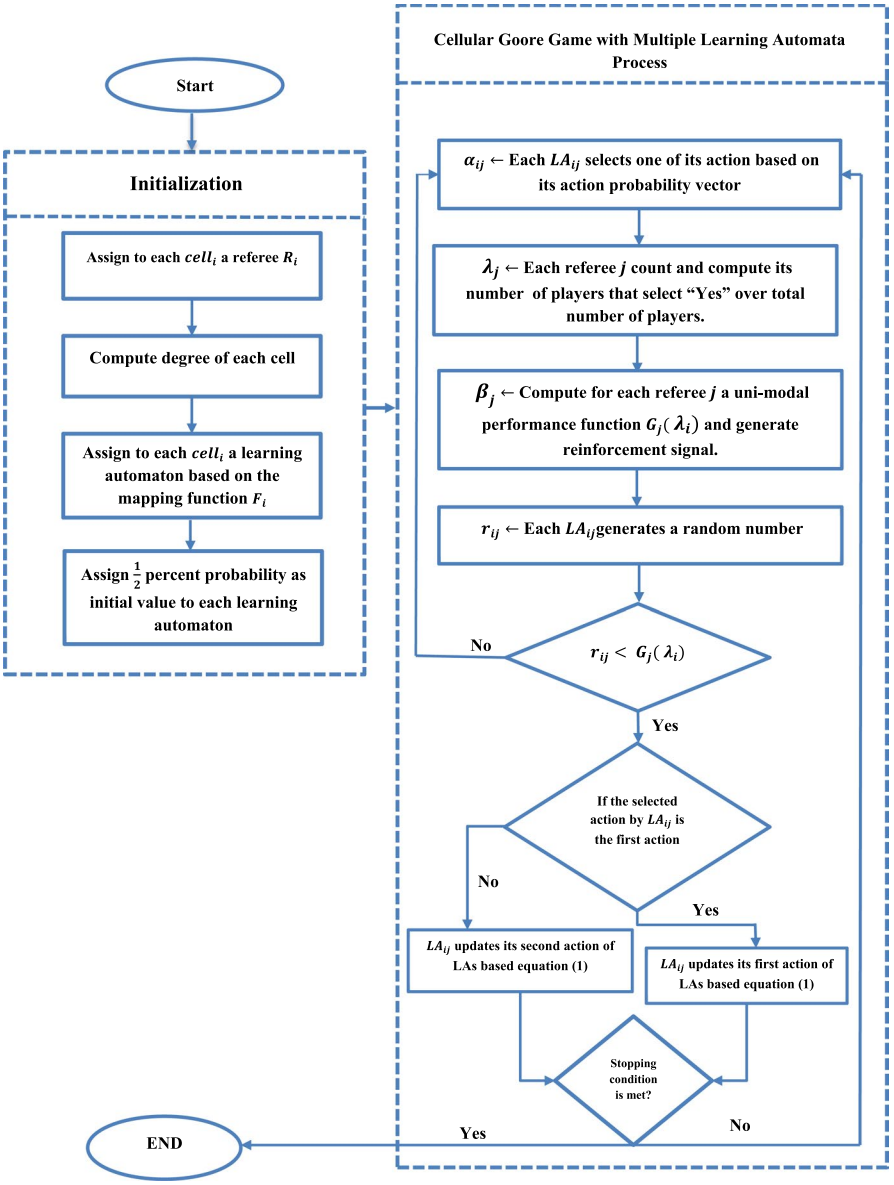


Fig. 6 Flowchart of the CGG-MLA

Definition 3 The dynamics of the CGG are described by a mapping $\mathcal{G} : \mathcal{K} \to \mathcal{K}$, which represents the global behavior of CGG with multiple LAs.

Definition 4 The evolution of the CGG with multiple LAs from a given initial configuration $X(0) \in \mathcal{K}$ is a sequence of interpolated process $\{\underline{X}(t)\}_{t\geq 0}$, such that $\underline{X}(t+1) = a\Theta(\underline{X}(t)) + \underline{X}(t)$.

Our focus will be on investigating the asymptotic behavior of CGG, where the L_{R-I} learning algorithm is employed by all learning automata. The process $\left\{\underline{X}(t)\right\}_{t\geq 0}$, evolutions based on the L_{R-I} learning algorithm can be describe by the following differential equation:

$$\underline{X}(t+1) = \underline{X}(t) + \underline{a} \cdot \Theta\Big(\underline{X}(t) \cdot \underline{\beta}(t)\Big) \tag{3}$$

Definition 5 Let $\Delta X_i(t)$ indicates as the conditional expectation of $X_i(t)$ and be defined as follows:

$$\Delta X_i(t) = E[X_i(t+1) - X_i(t)|X(t)] \tag{4}$$

Equation (4) is applied to study the asymptotic behavior of the $X_i(t)$ in the expected sense.

Definition 6 Let $P(\mathrm{NE}(i).i)$ be the finite sub set of LAs that does not contain i and defined as follows:

$$P(\mathrm{NE}(i).i) = \{LA_j | \forall j : (i.j) \in E\} \tag{5}$$

Additionally, $\psi(P(\mathrm{NE}(i).i))$ indicates the subset node form $P(\mathrm{NE}(i).i)$ LAs and is defined as follows:

$$\psi(P(\mathrm{NE}(i) \cdot i)) = \{W : W \subseteq P(\mathrm{NE}(i) \cdot i)\} \tag{6}$$

Assumption 1 The reinforcement signal $\beta_i(t)$ is non-negative and bounded. For the sake of generality, we assume, $\beta_i \in [0.1]$.

Assumption 2 The expected value of G_i:$[0.1] \rightarrow [0.1]$ is continuous for $\forall i \in N$. This continuity is essential as G_i should not exhibit discontinuities for any given number of players.

Assumption 3 The referee function G_i for $\forall i \in N$ is a unimodal function which is computes by Eq. (7).

$$E\left[\beta_i | n_{l_i}\right] = G_i\left(\frac{n_{l_i}}{|NE_i|}\right) \tag{7}$$

where n_{l_i} is the number of LAs that is adjacent to $cell_i$ and select "Yes" action.

Assumption 4 In CGG with multiple LAs, there are not isolated learning automaton that do not participate in action selections in each iteration.

Assumption 5 In CGG with multiple LAs, all referees use same unimodal performance function to generate reinforcement signals.

Hence, the probability of choosing the first action and the second action k times is given by:

$$\Pr\left(n_k(t)=k|X.\alpha_i(t)=\alpha_{i1}\right)=\sum_{W\in P(\mathrm{NE}(i).i);|W|=k-1}\left(\prod_{m\in W}x_{m1}(t)\prod_{n\in\psi(\mathrm{NE}(i).i)-W}\left(1-x_{n1}(t)\right)\right) \tag{8}$$

And

$$\Pr\left(n_k(t)=k|X.\alpha_i(t)=\alpha_{i2}\right)=\sum_{W\in P(\mathrm{NE}(i).i);|W|=k}\left(\prod_{m\in W}x_{m1}(t)\prod_{n\in\psi(\mathrm{NE}(i).i)-W}\left(1-x_{n1}(t)\right)\right) \tag{9}$$

Since $\{X(t):t>0\}$ is Markovian and $\underline{\beta}(t)$ depends only on $X(t)$ and not on t, then Δx_i can be expressed as follows:

$$\Delta x_i=bH_i(X) \tag{10}$$

Hence from Eq. (10) for each $cell_i$, Δx_i is computed by Eq. (11):

$$\Delta x_i=E\left[x_i(t+1)-x_i(t)|X\right] \tag{11}$$

Assume the independence of $x_i(t+1)$ from $x_i(t)$, Eq. (11) is replaced with Eq. (12):

$$\Delta x_i=E\left[x_i(t+1)|X\right]-x_i(t) \tag{12}$$

Furthermore, by substituting L_{R-I} learning algorithm into Eq. (12) we derive Eq. (13):

$$\Delta x_i=x_{i1}+x_{i1}\left[aE\left[\beta|\alpha_i=\alpha_{i1}.X\right]\right]\left[1-x_{i1}\right]+x_{i2}\left[-aE\left[\beta|\alpha_i=\alpha_{i2}.X\right]x_{i1}\right]-x_{i1} \tag{13}$$

After some mathematical manipulation, Eq. (13) is transformed into Eq. (14)

$$\Delta x_i=x_{i1}+ax_{i1}\left[1-x_{i1}\right]\left[aE\left[\beta|\alpha_i=\alpha_{i1}.X\right]\right]-ax_{i1}\left(1-x_{i1}\right)\left[aE\left[\beta|\alpha_i=\alpha_{i1}.X\right]\right]-x_{i1} \tag{14}$$

Applying assumption (3) to Eq. (14) yields Eq. (15):

$$\Delta x_i=ax_{i1}\left(1-x_{i1}\right)E\left[\beta|\alpha_i=\alpha_{i1}.X\right]-ax_{i1}\left(1-x_{i1}\right)E\left[\beta|\alpha_i=\alpha_{i2}.X\right]=ax_{i1}\left(1-x_{i1}\right)H_i(X(t)) \tag{15}$$

Utilizing assumption (3) in Eq. (14), we substitute $E\left[\beta|\alpha_i=\alpha_{i1}.X\right]$ with Eq. (14) and $H_i(X)$ is calculated using Eq. (16)

$$H_i(X(t))=A_i(X(t))+B_i(X(t)) \tag{16}$$

where $A_i(X(t))$ is obtained by Eq. (17)

$$A_i(X(t))=\sum_{k=0}^{n-1}G_{k+1}\times\left(\sum_{W\in P(\mathrm{NE}(i).i);|W|=k}\left(\prod_{m\in W}x_{m1}(t)\prod_{n\in S(\mathrm{NE}(i).i)-W}\left(1-x_{n1}(t)\right)\right)\right) \tag{17}$$

And $B_i(X)$ is obtained by Eq. (18)

$$B_i(X(t)) = -\sum_{k=0}^{n-1} G_k \times \left(\sum_{W \in P(\mathrm{NE}(i).i);|W|=k} \left(\prod_{m \in W} x_{m1}(t) \prod_{n \in S(\mathrm{NE}(i).i)-W} \left(1 - x_{n1}(t)\right) \right) \right) \tag{18}$$

After some mathematical simplification, Eq. (15) can be derived as Eq. (19):

$$H_i(X(t)) = \underbrace{\sum_{k=0}^{n-1} \left(G_{k+1} - G_k\right)}_{1} \left(\underbrace{\sum_{W \in P(\mathrm{NE}(i).i);|W|=k} \left(\prod_{m \in W} x_{m1}(t) \prod_{n \in S(\mathrm{NE}(i).i)-W} \left(1 - x_{n1}(t)\right) \right)}_{2} \right) \tag{19}$$

In Eq. (17) and (18), when $t \to \infty$, the probability of LAs choosing the first action converges to unity. Therefore, using Eqs. (19), we obtain:

$$\lim_{t \to \infty} x_{m1}(t) = 1 \tag{20}$$

And at the same time

$$\lim_{t \to \infty} x_{n1}(t) = 1 \tag{21}$$

By utilizing the outcomes of Eqs. (18) and (19), the second part of Eq. (19) approaches the sum of the product of the probabilities to 1, leading to the derivation of Eq. (22).

$$H_i(X) = \sum_{k=0}^{n-1} \left(G_{k+1} - G_k\right) \tag{22}$$

Definition 7 Let u_{ij} be initially defined as a real value within the range $u_{ij} \in (0, 1)$ where $j \in \{1, \ldots, |NE_i|\}$ indicates the neighborhoods for $\mathrm{cell_i}$. Moreover, $\mathcal{U}_i$ is defined as a vector $\mathcal{U}_i = \left[u_{ij}\right]_{1 \times |NE_i|}$ for each $\mathrm{cell_i}$. Additionally, for each $\mathrm{cell_i}$, the initial value of $H_i\left(\mathcal{U}_i\right)$ is set to zero because we consider G_{k+1} to be equal to G_k in the first part of Eq. (19), denoted as zero.

Definition 8 Let k represent a combination of solutions for each $\mathrm{cell_i}$ of the CGG, with the matrix $U = \left[\mathcal{U}_1, \ldots, \mathcal{U}_n\right]$. For each $\mathrm{cell_i}$ $\frac{d\mathcal{U}_i}{dk} = \mathscr{A}_i\left(\mathcal{U}_i\right)$, where:

$$\mathscr{A}_i\left(\mathcal{U}_i\right) = u_{ij}\left(1 - u_{ij}\right) H_i(X)\, j \in \left\{1, \ldots, \left|NE_i\right|\right\} \tag{23}$$

Lemma 1 *The maximum value of Eq.* (23) *is attained by solving* $\frac{d\mathscr{A}_i\left(\mathcal{U}_i\right)}{dk} = 0$.

Proof It is an optimization problem, and to finding the maximum value, we need to compute the derivative with respect to k [65]

In CGG with multiple LAs (CGG-MLA), each cell plays GG with its own environment simultaneously. Therefore, to examine the convergence behavior of CGG with multiple LAs in each cell, we analyze the convergence behavior of individual cell.

Theorem 3 *Considering all assumptions for CGG with multiple LAs, Eq.* (24) *is satisfied for each specific cell such as cell$_i$.*

$$MAXF(x_i) = \sum_{i=1}^{NE_i} E[\beta_i | x_i] \tag{24}$$

Subject to:

$$\sum_{i=1}^{NE_i} \sum_{j=1}^{r} x_{ij} = 1 \; r = 1, 2$$
$$i = 1, \ldots n$$

$$0 \leq x_{ij} \leq 1 \; \forall i, j$$

where $F(x_i)$ is the function that takes the probability vector of x_i and generates the reinforcement signal β_i.

Proof To proof this theorem, using lemma 1, the problem reduces to the classic GG, where we have:

$$\frac{d\mathcal{U}_i}{du_{ij}} = u_{ij}(1 - u_{ij}) \frac{\mathcal{U}_i(\mathcal{U}_i)}{du_{ij}} if j \neq i \tag{25}$$

$$\frac{d\mathcal{U}_i}{du_{ij}} = (1 - 2u_{ij}) \mathcal{U}_i(\mathcal{U}_i) if \; j = i \tag{26}$$

The only stable point of CGG is obtained when the eigenvalues of the matrix $\frac{d\mathcal{U}_i}{du_{ij}}$ is negative, where $1 \leq i \leq |NE_i|$ and $1 \leq j \leq |NE_i|$. For each cell$_i$, we consider $|NE_i|$ bit binary strings, with $u_{ij} \in \{0, 1\}$, for all j which are neighborhood of i. Therefore, we have:

$$\frac{d\mathcal{U}_i}{du_{ij}} = 0; for \; all \; j \neq i \tag{27}$$

And for j which is equal to i.

$$\frac{d\mathcal{U}_i}{du_{ij}} = \begin{cases} \mathcal{U}_i(u_{ij}) \; if \; u_{ij} = 0 \\ -\mathcal{U}_i(u_{ij}) \; if \; u_{ij} = 1 \end{cases} \tag{28}$$

Consider a solution string with k ones, with u_{ij} is equal to one when $i = j$. For this combination, it is easy to verify from Eq. (23) that,

$$\frac{d\mathcal{U}_i}{du_{ij}} = -\left(G_k - G_{k-1}\right) \tag{29}$$

Similarly, if there are k ones where u_{ij} equals zero when $i = j$.

$$\frac{d\mathcal{U}_i}{du_{ij}} = -\left(G_{k+1} - G_k\right) \tag{30}$$

Hence for such a string, the matrix $\frac{d\mathcal{U}_i}{du_{ij}}$ becomes a diagonal matrix with $-\left(G_k - G_{k-1}\right)$ for l diagonal entries and $-\left(G_{k+1} - G_k\right)$ for $(n-1)$ diagonal entries. Therefore, only for the value of k as indicated in remark 3, both eigenvalues are negative. Consequently, only those solutions containing k ones are asymptotically stable [60].

Proposition 1 *In CGG with multiple LAs, the maximum expected value of reinforcement signals is obtained using Eq.* (31):

$$\hat{\theta} = argmaxF(X) \tag{31}$$

where $F(X)$ represents the total estimation of reinforcement signals. Based on assumption 3, *in each cell, the referee utilizes the unimodal performance function to generates reinforcement signal. Hence, Eq.* (31) *for each $cell_i$ may be equivalent by Eq.* (32):

$$F(X) = \underset{X_i \in X}{\operatorname{argmax}} F\left(X_i\right)\ i \in \{1, \ldots, n\} \tag{32}$$

Equation (32) *implies that total expected value by CGG-MLA is equivalent to the maximum expectation β_i for each $cell_i$, which can be rewritten as Eq.* (33):

$$F\left(X_i\right) = E\left[\beta_i | X_i\right] \tag{33}$$

Proof Since the proposed model employs the same unimodal functions in each cell, the problem is thereby simplified as the expected maximization for a cell. To clarify further, maximizing the expected value of the proposed algorithm merely requires optimizing the unimodal performance function for a cell. In the following theorem, we provide a proof sketch demonstrating that CGG-MLA converges to a unit vector.

Theorem 4 *In CGG with multiple LAs, if the learning algorithm is L_{R-I}, the entropy, denoted by Z, approaches zero as follows*:

$$\text{Min}Z(t) = \sum_{i=1}^{n} \sum_{j=1}^{|NE_i|} x_{ij}(t) \log_2 x_{ij}(t) \tag{34}$$

Subject to:

$$0 \leq x_{ij}(t) \leq 1 \; i,j = 1,..,N \tag{35}$$

$$\sum_{j=1}^{|NE_i|} \sum_{l=1}^{2} x_{jl}(t) = 1 \; j = 1,..,N \tag{36}$$
$$l = \{1,2\}$$

$$x_{ij}(t) = \{0,1\} \; x_{ij} = \{0,1\} \tag{37}$$

In (34) $Z(t)$ is total entropy of the CGG with multiple LAs, where x_{ij} is the action probability vector of the learning automaton residing in $cell_i$, and $cell_i$ is adjacent to $cell_j$. Moreover, NE_i is defined as the number of adjacent cells for a $cell_i$.

Proof As a result of Eq. (17) and (18), when $t \to \infty$, x_{ij} tends toward a unit vector. Therefore, by taking the limit of $Z(t)$, we obtain Eq. (38)

$$\lim_{t\to\infty} Z(t) = \lim_{t\to\infty} \sum_{i=1}^{n} \sum_{j=1}^{|NE_i|} x_{ij}(t) \log_2 x_{ij}(t) \tag{38}$$

After some mathematical manipulation of Eq. (38), Eq. (39) is obtain:

$$\lim_{t\to\infty} \sum_{i=1}^{|n|} \sum_{j=1}^{|NE_i|} x_{ij}(t) \log_2 x_{ij}(t) = \sum_{i=1}^{n} \lim_{t\to\infty} \sum_{j=1}^{|NE_i|} \left(x_{ij}(t) \log_2 x_{ij}(t) \right) \tag{39}$$

Based on the limit theorem and using the associative property, Eq. (39) is transformed into Eq. (40):

$$\sum_{i=1}^{n} \lim_{t\to\infty} \sum_{j=1}^{|NE_i|} \left(x_{ij}(t) \log_2 x_{ij}(t) \right) = \sum_{i=1}^{|n|} \sum_{j=1}^{|NE_i|} \lim_{t\to\infty} \left(x_{ij}(t) \log_2 x_{ij}(t) \right) \tag{40}$$

When $t \to \infty$, the action probability of LAs falls into two cases. In case 1, the action probability of learning automaton approaches to one; thus, for this scenario, by substituting the value of $1 - \varepsilon$, where ε is close to zero, Eq. (41) is approximated.

$$\lim_{x_{ij}(t)\to 1-\varepsilon} \left(x_{ij}(t) \log_2 x_{ij}(t) \right) = 0 \tag{41}$$

In case 2, the action probability of learning automaton approaches to zero; therefore, by substituting the value of $0 + \varepsilon$, where ε is close to zero, Eq. (42) is approximated.

$$\lim_{x_{ij}(t)\to 0+\varepsilon} \left(x_{ij}(t) \log_2 x_{ij}(t) \right) = 0 \tag{42}$$

From Eq. (39) and (40) the obtained result for theorem 4 is straight forward. □

4.3 Complexity analysis

From a computational complexity perspective and in terms of space complexity, consider a scenario where the maximum and minimum degrees of the graph are both equal to n-1 and 1, respectively. Consequently, in the worst-case scenario, for a graph with n memory cells, a total of $n^2 - n$ LAs are required, whereas in the best case, only n LAs suffice. Given that each automaton possesses a single probability vector containing two actions, it becomes feasible to store this probability vector using a single memory unit and derive the second probability through subtraction from one. Hence, the CGG-MLA algorithm demonstrates a time complexity of $O\left(n^2\right)$ in the worst case and $O(n)$ in the best case. However, when considering time complexity, it primarily revolves around the LAs response to environmental stimuli and the learning rate influencing convergence. Determining the optimal learning rate remains an ongoing challenge in automaton theory[25]. In the context of the CGG-MLA model, the interplay between the behaviors of multiple LAs complicates this matter, rendering it a complex problem without a precise optimal solution. Addressing this challenge is identified as a potential avenue for future research.

5 Applications: an algorithm for overlapping community detection in multilayer social networks.

This section introduces a novel algorithm, CGG-OCD, based on Cellular Goore Game, designed to address overlapping community detection in multilayer social networks. The CGG-OCD algorithm operates locally and independently, with each cell making decisions autonomously. Additionally, the algorithm is fully distributed, ensuring it avoids local optima. The objective of this research is to explore the potential of employing Cellular Goore Game with multiple learning automata per cell for efficient community detection. To achieve this objective, we outline the algorithm's structure.

Definition 9 A multilayer network $\mathcal{M}$ is defined as $\mathcal{M} = (\mathcal{G}, \mathcal{C})$, where $\mathcal{G} = \left\{G_i | i \in \{1, 2, \ldots, L\}\right\}$ represents the set of networks, with |L| denoting the total number of layers. Each, $G_i = \left(V_i, E_i\right)$ denotes a network in layer i, where G_i is termed the layer of M. Furthermore, $\mathcal{C} = \left\{E_{ij} \subseteq V_i \times V_j : i, j \in \{1, 2, \ldots ., L\}, i \neq j\right\}$ represents the set of interconnections between nodes across different layers G_i and G_j, where $i \neq j$. The edges within the E_i set are known as intra-edges. Elements of $\mathcal{C}$ are referred to as crossed layers, while elements of each E_i are termed inter-layer edges of $\mathcal{M}$, in contrast to E_{ij}, which are termed cross-layer edges.

Definition 10 The set of communities, denoted as $S = \left\{C_1, C_2, \ldots, C_m\right\}$, comprises individual communities represented by $C_i = (g, l)$, where g is a subset of G and l is a member of |L|. In order to establish a formal definition for multilayer communities, we have expanded upon the definition provided in [61] for simple networks.

Definition 11 A multilayer strong community refers to a group of nodes where, in each layer, the internal degree of the nodes surpasses the external degree. This condition can be mathematically expressed using Eq. (43):

$$\forall l \in L : i \in g : K_{i,l}^{in}(g) > K_{i,l}^{out}(g) \tag{43}$$

where $K_{i,l}^{in}(g)$ indicates as the input degree of node i in layer l to community g and $K_{i,l}^{out}(g)$ indicates as the output degree of node i in layer l to community g

Definition 12 A multilayer weak community can be defined as a collection of nodes where, in each layer, the total internal degree of the nodes exceeds the total external degree of the nodes. This condition is mathematically determined by the following Eq. (44):

$$\sum_{i \in l} \sum_{j \in g} K_{i,j}^{in}(g) > \sum_{i \in l} \sum_{j \in g} K_{i,j}^{out}(g) \tag{44}$$

where $K_{i,l}^{in}(g)$ indicates as the input degree of node i in layer l to community g and $K_{i,l}^{out}(g)$ indicates as the output degree of node i in layer l to community g

Definition 13 The density of a community i in a multilayer network is the number of all the edges connecting the all members of C_i for all $v_i^l \in C_i$. It is defined as follows:

$$den(C_i) = \left\{ \left(v_i^l, v_j^l \right) \in E_i^l, v_i^l \in C_i and v_j^l \in C_i \right\} \tag{45}$$

Definition 14 The disconnectivity of a community i is define as the number of all the edges outside of community C_i.

The disconnectivity of community C_i is denoted by Sep $\left(C_i\right)$ and defined by Eq. (46):

$$dis(C_i) == \left\{ \left(v_i^l, v_j^{l'} \right) \in C_i, v_i^l \in C_i and v_j^l \notin C_i \right\} \tag{46}$$

The primary objective is to compute a partitioning in which communities exhibit compactness and separability. However, it is important to note that the notion of disconnectivity does not imply complete isolation, as overlaps between communities are allowed. All these concepts are taken into consideration in our objective function, termed the ‘‘Index of connectivity,’’ which is defined as follows.

Definition 15 To determine the quality of connectivity of a community is based on the idea of maximizations of the internal edges to a community. The objective function is defined as the difference among the density and disconnectivity, normalized by the sum of the density and disconnectivity of the considered one as defined as follows:

$$G(C_i) = \frac{den(C_i)}{den(C_i) + dis(C_i)} \tag{47}$$

The objective function $G(C_i)$ is maximal, when the density of community is high and disconnectivity is low. In other words, the vertices included in community have strong connection among them and are loosely connected with nodes outside this community.

Definition 16 Based on definition (10), we denoted a community as $C_i = (g, l)$, where g represents a subset of G, and l is an element of $|M|$. In the context of Multilayer Overlapping Communities detection, the communities $C_1 = (g_1, l_1)$ and $C_2 = (g_2, l_2)$ are identified as multilayer overlapping communities if and only if $\{l_1 = l_2 \wedge g_1 \cap g_2 \neq \emptyset\}$. Hence, it is crucial to note that even if two communities share common nodes across different layers, they do not qualify as overlapping communities due to this conceptual distinction. Two instances of simple overlapping communities within a multilayer network are depicted in Fig. 7 as follows.

6 Proposed algorithm

This section presents the CGG-based algorithm for detecting overlapping communities in multilayer social networks named CGG-OV. The CGG-OV algorithm operates locally and independently in each cell to identify communities with overlapping structure within these networks. We delve deeper into the process of detecting overlapping communities using the Cellular Goore Game (CGG) with multiple Learning Automata (LAs) in each cell. To elucidate this objective, we initially outline the algorithm's whole structure in brief. The CGG-OV

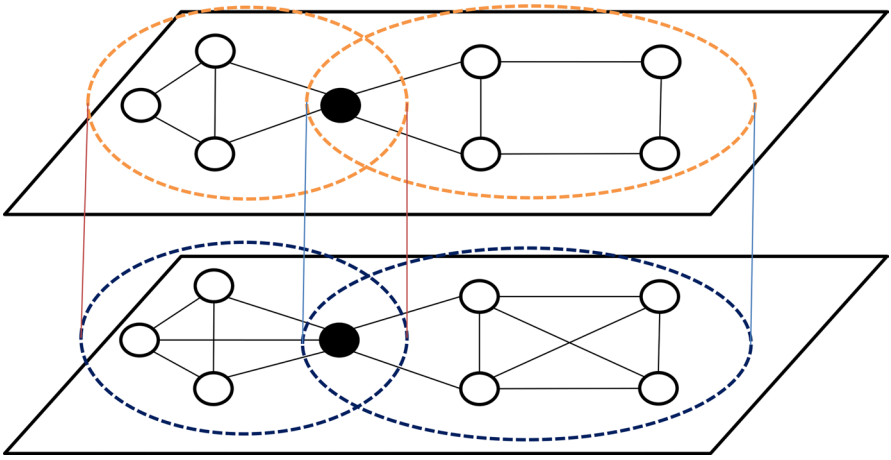


Fig. 7 An illustration of multilayer overlapping communities depicts subsets of nodes characterized by dense interconnections within the same subset of layers and sharing common node membership

algorithm comprises of five steps. Let $\mathcal{M} = (\mathcal{G}, \mathcal{C})$ denote a multilayer network, where $\mathcal{G} = \{G_i | i \in \{1, 2, \ldots, L\}\}$ represents the set of networks and L is total number of layers. Each network $G_i \in \mathcal{G}$, defined as $G_i = (V_i, E_i)$ consist of a node set $V_i = \{v_{i1}, \ldots, .v_{in}\}$ and an edge set $E_i \subseteq V_i \times V_i$, respectively. We note that, if $C_i = (V_i, l_i)$ and $C_j = (V_j, l_j)$ are two multilayer overlapping communities, then $l_i = l_j$ and $V_i \cap V_j \neq \emptyset$. Initially, a CGG with multiple LAs is isomorphic to each cell of the multilayer networks. Subsequently, CGG-OV utilizes leaning automata tries to find candidate dense communities. In the subsequence step, these candidate communities are evaluated by uni-modal performance function, referred to as F, employed by a referee. Based on the referee's evaluation of the communities, reinforcement signals are generated for each neighborhood LA, and each LA updates its action accordingly. This process of community identification, evaluation by the referee, and action probability vector update continues until a predefined criterion is met or no changes occur in the action probability.

6.1 Initialization

To initialize the algorithm, a CGG with multiple LAs in each cell is created, which is isomorphic to the input network. This resulting network can be described as $\mathrm{N} = (\underline{\mathrm{A}}, \underline{a})$, where $\underline{\mathrm{A}} = \{A_i | i \in \{1, 2, \ldots, M\}\}$ and $A_i = \{A_i^1, A_i^2, \ldots, A_i^n\}$ denote the set of learning automata assigned to network in layer i, and A_i^j represents the learning automaton assigned to cell j in layer i. Moreover, $\underline{a} = \{\underline{a_1}, \underline{a_2}, \ldots, \underline{a_M}\}$ represents the set of action which is assign to LAs where $\underline{a_i} = \{a_i^1, a_i^2, \ldots, a_i^n\}$ indicate the set of action of automaton in layer i in which a_i^j represent the set of actions that is taken by automaton j in layer i. To assign LAs to each cell of this network, a mapping function exists which maps each learning automaton to the cells of the input network based on following assignment. Let $G_i = (V_i, E_i)$ where v_k^i, $v_q^i \subseteq V_i$ and $(v_k^i, v_q^i) \in E_i$ and $i \in L$ then LA_{kq}^i is assign to cell v_k and LA_{qk}^i to cell v_q in layer i. Additionally, if $\mathcal{C} = \{E_{ij} \subseteq V_i \times V_j : i,j \in \{1, 2, \ldots ., L\}, i \neq j\})$ where $v_k^i \subseteq V_i$, $v_j^j \subseteq V_j$ and $(v_i^i, v_j^j) \in E_{i,j}$ for each $i < j$ then LA_{ij}^i is assign to cell v_i^i in layer i and LA_{ji}^j is assign to cell v_j^j in layer j. Moreover, each LA_{ij}^i has two actions corresponding to α_1^i, α_2^i, representing "Yes" and "NO", respectively. At the initial step, the value of each action has equal probability. Additionally, in each cell $v_i^i \subseteq V_i$, a referee R_i^i resides, which counts the fraction of "Yes" over total number of LAs participating in this game.

6.2 Community formation

In this step, each cell simultaneously constructs its own local communities based on internal density and external disconnectivity found by the algorithm. To create such local communities, each learning automata LA_{ij}^i, which is assigned to cell v_i^i in layer i, selects one of its possible actions. Let a_{i1}^j be the first action corresponding to "Yes"

selected by LA_{ij}^{i}. Then, each referee R_j^l residing inside cell j in layer i counts the fraction of "Yes" of its corresponding learning automata in the neighborhood of the adjacent cells to total LAs that participate for local communities formation. Let λ_i^l be the function that counts number of LAs that select its first action a_{i1}^l in layer l, defined by Eq. (48):

$$\lambda_i^l(t) = \sum_{l=1}^{|L|} \sum_{j=1}^{d_{ij}^l} I\left\{ \alpha_i^j(t) = \alpha_{i1} \right\} \tag{48}$$

where $d_{ij}^l\left(v_i^i\right)$ is the number of LAs that are resided in the adjacent cell of v_i and I stand stands for the indictor function.

6.3 Community evaluation

In this step, the candidate communities are evaluated based on reinforcement signal generated by the algorithm. Let θ_i^l and ζ_i^l denote the density and disconnectivity of community C_i in layer l, respectively. Then, the expected value $E[\beta_i^l|\theta_i^l, \zeta_i^l]$is defined by Eq. (49):

$$E\left[\beta_i^l(t)|\theta_i^l(t), \zeta_i^l(t)\right] = \phi_i\left(\frac{den\left(C_i(t)\right)}{den\left(C_i(t)\right) + dis\left(C_i(t)\right)} \right) \tag{49}$$

where ϕ is defined by Eq. (50)

$$\phi_i\left(\theta_i^l(t), \zeta_i^l(t)\right) = 0.2 + 0.8e^{0.002\left(\left(|\theta_i^l(t)|-|\theta^*|\right)^2\left(|\zeta_i^l(t)|-|\zeta^*|\right)^2\right)} \tag{50}$$

where $|\theta_i^l(t)|$ and $|\zeta_i^l(t)|$ represent the cardinality of density and cardinality of sparsity of the community C_i at round t, respectively. Additionally, $|\theta^*|$ and $|\zeta^*|$ are the cardinality of complete and sparse community, respectively. It should be noted that, due to the nature of the GG, the function ϕ_i is a uni-modal performance function which is optimized when $\left|\theta_i^l(t)\right|$ is equal to $|\theta^*|$ and $\left|\zeta_i^l(t)\right| - |\zeta^*|$.

6.4 Update action probability vector

In this step, for each cell i, if the selected action by LAs forms dense community, the value of the function ϕ_i is computed to generate reinforcement signals based on Eq. (49). Let $\beta_i^l(t)$ denote the reinforcement signal generated by cell i in layer l at iteration t. Then, each LA associated with this cells updates its internal states based on L_{R-I} learning algorithm. The updating of internal states are occurs based on the value of $\beta_i^l(t)$ and ϕ_i for each cell i. In this step, each LAs generates its own random number independently to update its action probability vector. If the selected action by LAs is equal to a_{i1}^l and the random number generated by each learning automaton is less than the value of ϕ_i, then the probability vector of the associated LA updates based on Eq. (51):

$$\begin{aligned} x_{i1}^{l}(t+1) &= x_{i1}^{l}(t) + a\left(1 - x_{i1}^{l}(t)\right) \\ x_{i2}^{l}(t+1) &= x_{i2}^{l}(t) - a x_{i2}^{l}(t) \end{aligned} \tag{51}$$

where a determines the learning rate of the learning algorithm. Also, if the selected action by LAs is equal to a_{i2}^{l} and the random number generated by each learning automaton is greater than the value of ϕ_i, then the probability vector of the associated LA updates based on Eq. (52):

$$\begin{aligned} x_{i2}^{l}(t+1) &= x_{i2}^{l}(t) + a\left(1 - x_{i2}^{l}(t)\right) \\ x_{i1}^{l}(t+1) &= x_{i1}^{l}(t) - a x_{i1}^{l}(t) \end{aligned} \tag{52}$$

If the conditions are not satisfied for updating based on Eq. (51) and (52), then algorithm does not perform any updates and next iteration starts.

6.5 Stopping condition

The proposed CGG-OV stops when the number of iterations exceeds a predefine maximum iteration T_{max} or when the value of entropy defined by Eq. (53) reaches a predefine value.

$$EN\left(x_{ij}^{l}\right) = -\sum_{j=1}^{2} x_{ij}^{l} \log x_{ij}^{l} \tag{53}$$

where x_{ij}^{l} is the probability of selected action j in cell i in layer l and $EN\left(x_{ij}^{l}\right)$ is the entropy value of its counterpart. The pseudo-code of the CGG-OV algorithm is demonstrates in Fig. 8.

7 Evaluation metric

A comparison of the proposed CGG-OV algorithm with different methods is made on both synthetic and real network datasets in order to evaluate its effectiveness (Figs. 9 and 10).

7.1 Normalized mutual information

The Normalized Mutual Information (NMI) [62] is a measurement that assesses the extent to which nodes in two layers, α and β, of a multiplex belong to the same community. It quantifies the similarity between the community structures of these two layers. When the NMI value is high, it indicates that the community structures of the two layers are quite similar. In other words, if the NMI value is high, it suggests that both layers have a comparable distribution of communities, meaning that they possess similar community structures. This information is valuable for researchers interested in understanding how different layers interact and the similarities or differences in

Cellular Goore Game with Multiple LAs in each Cell for Overlapping Community Detection

Inputs: The input multilayer Graph $\mathcal{M} = (\mathcal{G}, \mathcal{C})$, T_{max} maximum iteration number, Min_{EN} minimum value of entropy.
Output: The set of overlapping communities
Initialization
Let N = $(\underline{A}, \underline{a})$ is the networks of learning automata Which is isomorph to the input multilayer network.
Let $\underline{A} = \{A_i | i \in \{1,2,\dots,M\}\}$ is the set of learning automaton that is assign to network in layer i.
Let $A_i = \{A_i^1, A_i^2, \dots, A_i^n\}$ indicates as the set of learning automaton that is assign to network associated with layer i and A_i^j is the learning automaton that is assign to cell j in layer i.
Let $\underline{a} = \{\underline{a_1}, \underline{a_2}, \dots, \underline{a_M}\}$ denotes the set of action which is assign to LAs associated with its LAs.
Let $\underline{a_i}$=$\{a_i^1, a_i^2, \dots, a_i^n\}$ indicate the set of action of automaton in layer i in which a_i^j represent the set of actions that is taken by automaton j in layer i.
Let x_{ij}^l is the probability of selected action j in cell i into layer l and the initial value is 0.5.
Begin
Let t is the iteration number and initially set to 1.
Let $EN(x_{ij}^l)$ is the entropy value of action j in cell i that associated with layer l and the initial value is 1.
Repeat
  For each $cell_i^l$ in $\mathcal{G}$ **do** in parallel
    LA_i^l selects one of its action α_{ij}^l from two available actions "Yes" or "NO" based on its action probability vector $x_i^l = (x_{i1}^l, x_{i2}^l)$.
  End For
  For each $cell_i^l$ in R_i^l **do** in parallel
    Referee R_i^l count the fraction of "Yes" to determine its own communities based on neighborhood LAs.
    IF the number of "Yes" forms community based on equation (44) **then**
      Compute density of community C_i^l based on equation (45)
      Compute dis-connectivity of community C_i^l based on equation (46)
    End IF
  End For
  For each $cell_i^l$ in $\mathcal{G}$ **do** in parallel
    Let $\phi_i\left(\theta_i^l(t), \zeta_i^l(t)\right)$ is defined as reward probability at iteration t that is computed by referee R_i^l
    IF $(rnd_i^l(t) < \phi_i(\theta_i^l(t), \zeta_i^l(t))$ and select action is α_{i1}^j **then**
      LA_i^j update its action probability vector based on equation (51)
    End IF
    IF $(rnd_i^l(t) > \phi_i(\theta_i^l(t), \zeta_i^l(t))$ and select action is α_{i1}^j **then**
      LA_i^j update its action probability vector based on equation (52)
    End IF
  Calculate information entropy
  Set t= $t + 1$
Until t > T_{max} and $EN(x_{ij}^l) > Min_{EN}$
end

Fig. 8 The pseudo-code of CGG-OV is outlined

their structures. By comparing the NMI values of various layers, researchers can gain insights into the relationships between these layers and their community structures. Let $g_i^{[k]}$ and,$\overline{g}_i^{[q]}$ represent two community assignments, indicating the communities $\sigma = g_i^{[k]}$ to which the replica node (i,k) belongs $\sigma = 1, 2, \dots, P^{[k]}$, and the communities $\sigma = \overline{g}_i^{[q]}$ to which the replica node (i,q) belongs, with $\sigma' = 1, 2, \dots, P^{[k]}$. The community structure can be captured using the NMI, as given by Eq. (54).

$$NMI\left(g_i^{[k]}, \overline{g}_i^{[q]}\right) = \frac{-2\sum_{\sigma=1}^{P[k]}\sum_{\sigma'=1}^{P[q]} N_{\sigma,\sigma'}^{[k,q]} \log\left(\frac{N_{\sigma,\sigma'}^{[k,q]} N}{N_\sigma^{[k]} N_{\sigma'}^{[q]}}\right)}{\sum_{\sigma=1}^{P[k]} N_\sigma^{[k]} \log\left(\frac{N_\sigma^{[k]}}{N}\right) + \sum_{\sigma'=1}^{P[q]} N_{\sigma'}^{[q]} \log\left(\frac{N_{\sigma'}^{[q]}}{N}\right)} \tag{54}$$

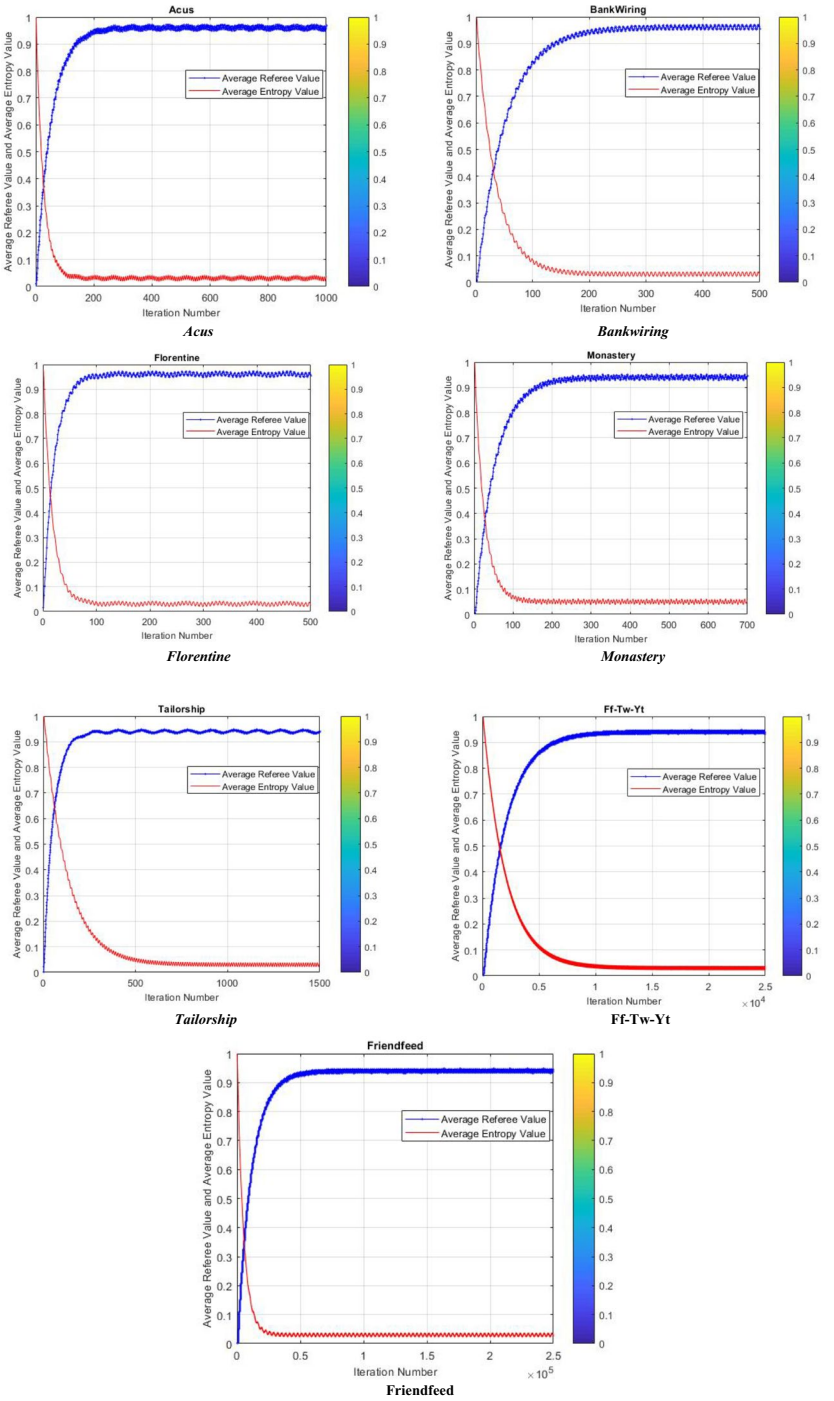


Fig. 9 The plot of average entropy value versus iteration number for different learning rate

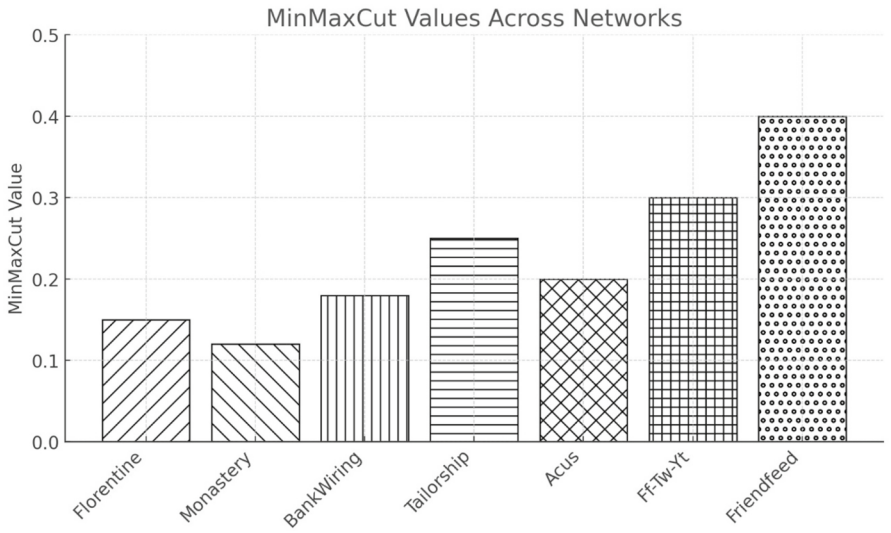


Fig. 10 The results for MinMax Cut for real multilayer networks

where $N_{\sigma}^{[k]}$ and $N_{\sigma'}^{[q]}$ determine the number of nodes belong to community σ in layer k and in the community σ' in layer q, respectively. Moreover, $N_{\sigma,\sigma'}^{[k,q]}$ shows how many nodes have replicas in layer k in community σ and their replica node in layer q in the community σ'.

7.2 Modularity

The generalized modularity quantifies the quality of the multilayer community assignment by measuring how tightly connected the multilayer communities are with respect to the random hypothesis. Specifically, the generalized modularity Q^M is determined by equation:

$$Q^M = \frac{1}{\mu} \sum_{i,j,k,q} \left\{ \left(A_{ik,jq} - \gamma^{[k]} \frac{k_i^{[k]} k_j^{[q]}}{k^{[k]} N} \right) \delta_{k.q} + \omega A_{ik,jq} \delta_{ij} \right\} \delta\left(g_i^{[k]}, g_j^{[q]} \right) \tag{55}$$

where $A_{ik,jq}$ is the supra-adjacency matrix, $\mu = \sum_{i,j,k} A_{ik,jk} + \omega \sum_{i,j,q} A_{iq,jq}$ and $\delta(x, y)$ indicates as the kronecker delta.

7.3 MinMaxCut

In the context of overlapping community detection in a multilayer network with L layers, the concept of cuts between communities' C_i and C_j defines the edges connecting these communities. The MinMaxCut of a partition P of the multilayer

graph is the ratio of overlapping inter-community edges to the total number of edges associated with the different communities separately, mathematically expressed as:

$$MinMaxCut(P) = \sum_{i=1}^{K} \frac{E_{oE}\prime}{E_i} \tag{56}$$

where $E_{oE}\prime$ is the number of overlapping edges between community C_i and other communities, and E_i is the number of edges of community C_i. Minimizing the MinMaxCut, as shown in Eq. (56), results in well-separated and dense communities.

7.4 Coverage

The coverage of a partition P for overlapping community detection in a multilayer network with L layers is defined as the ratio of interior edges within the communities to the total number of edges, mathematically defined as:

$$Coverage(P) = \sum_{i=1}^{K} \frac{E_{oi}}{E} \tag{57}$$

where E_{oi} is the number of edges community of C_i. Maximizing the coverage, as shown in Eq. (57), results in dense communities.

7.5 Adjusted rand index

The adjusted rand index (ARI) is a widely used metric to measure the similarity between two clustering results while correcting for random chance. In the context of multilayer social networks, ARI can be adapted to compare the detected community structure across multiple layers with a given ground truth or another benchmark clustering. Given two partitions C (the detected communities) and C′ (the ground truth or benchmark clustering), ARI is defined as:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left(\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right) / \binom{N}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] / \binom{N}{2}} \tag{58}$$

where N is the total number of nodes, n_{ij} is the number of nodes common to both community i in C and community j in C′, a_i is the sum of all nodes in community i of C, b_j is the sum of all nodes in community j of C′ and $\binom{a_i}{2} = \frac{a_i(a_i-1)}{2}$ is the binomial coefficient counting the number of ways to choose two elements from a_i. For multilayer networks, ARI can be adapted as follows:

$$ARI_{multi} = \frac{\sum_{l=1}^{L} w_l . ARI_l}{\sum_{l=1}^{L} w_l} \tag{59}$$

where w_l is the weight of layer l based on edge density or significance.

8 Experiment setting

In order to assess the effectiveness and efficiency of the CGG-OV algorithm in detecting overlapping community structures, a series of experiments is conducted on both real networks (Florentine, Monastery, Bankwiring, Tailorship) as presented in Table 2, and synthetic networks (MLFR benchmark [14]). In the MLFR networks, various parameters are considered for characterization: "$-Orig$" represents the number of nodes in the original network, "L" denotes the number of layers, "N" signifies the number of nodes, "K" indicates the average degree, "$MaxK$" represents the maximum degree of nodes, "μ" relates to the mixing parameter, "τ_1" represents the negative exponent for the degree sequence, "τ_2" signifies the negative exponent for the community size distribution, "$MinC$" denotes the minimum community size, "$MaxC$" represents the maximum community size, "ON" refers to the number of overlapping nodes, and "OM" represents the number of memberships assigned to overlapping nodes.

Additionally, the performance evaluation of the CGG-OV algorithm includes the utilization of the well-known computer-generated network, MLFR [63]. For the synthetic modular network, specific parameters are set according to the MLFR benchmark guidelines. These parameters include "$-Orig = 1$" to specify the number of the original network, "$L = 3$" indicating the number of layers, "$\mathrm{N} = 1000$" representing the total number of nodes, "$K = 1000$" signifying the average degree, "$MaxK = 50$" denoting the maximum degree, "$\mu = \{0 - 0.8\}$" specifying the range of the mixing parameter, "$\tau_1 = 2$" representing the

Table 2 Characteristic of multilayer networks

Networks	Nodes	Edges	Layer	Description
Florentine	15	35	2	Network of politically prominent families in Florence
Monastery	18	510	10	Network of three choices of a group of monks on four pairs of relations
Bankwiring	14	110	6	Network of employees that work in the bank wiring room
Tailorship	39	552	4	Network of work and friendship interactions among workers in a tailor ship
Acus	61	620	5	Network of relationships and interactions among professors and other staff
Ff-Tw-Yt	6407	74,862	3	A network of anonymized users from Facebook, Twitter, and YouTube
Friendfeed	510,896	20,330,701	3	Network of social interaction with commenting, liking, and following

negative exponent for the degree sequence, "$\tau_2 = 1$" indicating the negative exponent for the community size distribution, "MinC = 20" representing the minimum community size, "$MaxC = 50$" denoting the maximum community size, "$ON = \{0 - 200\}$" specifying the range of overlapping nodes, and "$OM = \{1 - 4\}$" signifying the range of membership values for overlapping nodes. By performing experiments on both real and synthetic networks while considering a wide range of parameter settings, a comprehensive evaluation of the CGG-OV algorithm's capabilities in detecting overlapping communities is achieved. This ensures a thorough analysis of its performance under various network configurations and enhances our understanding of its efficiency and applicability.

To evaluate our proposed algorithm, our first experiment conducted on synthetic networks.

8.1 Experiment results

8.1.1 Experiment I

This experiment provides the impact of varying learning rates on modularity across different network structures and reveals several key observations. A consistent trend is observed across all networks, where an increase in the learning rate leads to a decline in modularity scores. This suggests that higher learning rates may introduce instability in the learning automata, ultimately affecting the quality of detected community structures. Notably, smaller networks such as the Monastery and Tailorship networks exhibit a steeper decline in modularity compared to larger and more complex networks like Acus and Ff-Tw-Yt. This implies that smaller networks are more sensitive to variations in learning rates, likely due to their limited connectivity and fewer well-defined clusters, whereas larger networks maintain relatively stable modularity values despite an increase in the learning rate. The Acus network, which initially demonstrates the highest modularity score, experiences a decline similar to other networks but remains more resilient, suggesting that denser connectivity structures help preserve community integrity even at moderate learning rates. In terms of parameter selection, the results indicate that lower learning rates (≤ 0.03) generally yield better modularity performance across all networks, ensuring that the learning automata can effectively capture community structures without excessive parameter fluctuation. Moderate learning rates (0.04–0.06) present a balance between modularity retention and convergence speed, whereas higher learning rates (≥ 0.07) result in significant degradation of modularity, indicating that aggressive updates may disrupt the stability of learned partitions. Consequently, an adaptive or network-specific learning rate adjustment strategy is recommended to optimize modularity while maintaining convergence efficiency. In the following experiments, we used 0.01 as the learning rate for the automata. The results are shown in Table 3.

Table 3 Impact of varying learning rates on modularity across different multilayer social network structures

Learning rate	*Florentine*	*Monastery*	*Bankwiring*	*Tailorship*	*Acus*	*Ff-Tw-Yt*	*Friendfeed*
0.01	*0.592*	*0.419*	*0.526*	*0.515*	*0.801*	*0.498*	*0.586*
0.02	0.578	0.402	0.512	0.503	0.788	0.485	0.572
0.03	0.563	0.390	0.500	0.491	0.775	0.472	0.560
0.04	0.550	0.375	0.487	0.475	0.760	0.460	0.547
0.05	0.537	0.360	0.475	0.466	0.745	0.448	0.535
0.06	0.523	0.345	0.462	0.453	0.730	0.435	0.522
0.07	0.510	0.330	0.450	0.440	0.715	0.422	0.510
0.08	0.495	0.315	0.438	0.428	0.700	0.410	0.498
0.09	0.480	0.300	0.425	0.415	0.685	0.397	0.485
0.1	0.465	0.285	0.412	0.402	0.670	0.385	0.473

8.1.2 Experiment II

The experimental analysis of the proposed CGG-based overlapping community detection algorithm demonstrates its effectiveness across different multilayer social network datasets, including Acus, BankWiring, Florentine, and Friendfeed. The results indicate that in networks with well-defined community structures, such as Acus, BankWiring, and Florentine, the algorithm converges rapidly, with the Average Referee Value (ARV) stabilizing close to 1 within approximately 200 iterations, while the Average Entropy Value (AEV) declines swiftly, indicating minimal uncertainty in community assignments. In contrast, the Friendfeed network, characterized by its larger size and potentially overlapping communities, requires significantly more iterations (over 200,000) to stabilize, suggesting a more complex underlying structure.

Similarly, for Monastery (small-scale), Tailorship (medium-scale), and Ff-Tw-Yt (large-scale) networks, the ARV increases monotonically and stabilizes near 1, while the AEV declines rapidly, approaching zero, signifying effective community optimization. The Monastery network exhibits the fastest convergence, stabilizing within ~ 100 iterations, implying well-defined communities with minimal uncertainty. The Tailorship network follows a similar trend but requires approximately 500 iterations, reflecting a moderately complex structure with minor fluctuations in community definition. The Ff-Tw-Yt network, being significantly larger, demands ~ 20,000 iterations to achieve stability, with entropy decreasing more gradually due to its high sparsity and intricate interactions.

These trends indicate that smaller networks converge faster, whereas larger networks require more computational effort due to increased structural complexity. The consistent decline in AEV across all datasets validates the algorithm's ability to minimize uncertainty and refine community assignments over time. Furthermore, the near-optimal ARV suggests strong intra-community connectivity across different network topologies. Therefore, these findings confirm the effectiveness and scalability of the CGG-based algorithm in detecting communities, though

larger and more intricate networks necessitate extended learning phases for optimal performance.

8.1.3 Experiment III

The experiment was conducted to compare the MinMaxCut values, Coverage, and adjusted rand index (ARI) across multiple multilayer networks. The MinMaxCut metric, which quantifies the quality of partitioning by minimizing inter-community edges while maximizing intra-community connectivity, produces lower values for smaller datasets such as Florentine and BankWiring, indicating well-defined community structures. In contrast, large-scale networks such as Friendfeed exhibit relatively higher MinMaxCut values, likely due to the complexity of their dense and diverse interactions.

The Coverage metric, which measures the proportion of correctly clustered edges, remains consistently high across all datasets, signifying the effectiveness of the learning automata-based approach in capturing network structures. Notably, simpler social structures, such as Acus and BankWiring, achieve near-optimal coverage, whereas massive networks, such as Friendfeed, display marginally reduced coverage due to their inherent sparsity and noise. Moreover, the Adjusted Rand Index (ARI), which evaluates clustering accuracy relative to a ground truth, follows a similar trend: small to medium-sized datasets (e.g., Florentine and Tailorship) exhibit near-perfect ARI, while large-scale and highly complex datasets, such as Friendfeed, show comparatively lower ARI values. This suggests that the proposed approach effectively identifies communities in structured networks but may be affected by the increasing complexity and sparsity of larger networks. Thus, the results validate the efficacy of the learning automata model in detecting community structures, particularly in networks with well-defined relationships and moderate inter-layer dependencies.

8.1.4 Experiment V

This experiment investigates the convergence behavior of the proposed CGG-OV algorithm compared to other algorithms in terms of Normalized Mutual Information for multilayer social networks (MNMI). To conduct this experiment, we explored the impact of different mixing parameters (μ) generated for multilayer LFR networks and their corresponding NMI values. Additionally, we studied the influence of the number of memberships for the overlapping nodes, which varied from 1 to 4 in these networks. The obtained results are depicted in Fig. 11. From the results, it is observed that by varying the mixing parameters, the proposed algorithm successfully identifies overlapping communities for communities with lower mixing parameter rates. However, as the mixing parameter rate increases, leading to less clarity in the communities, all algorithms exhibit similar NMI values. The CGG-OV algorithm outperforms other algorithms due to its utilization of learning mechanisms. In other words, by employing learning automata and leveraging the objective function, the proposed algorithm effectively guides the discovery of overlapping communities in the networks (Figs. 12 and 13).

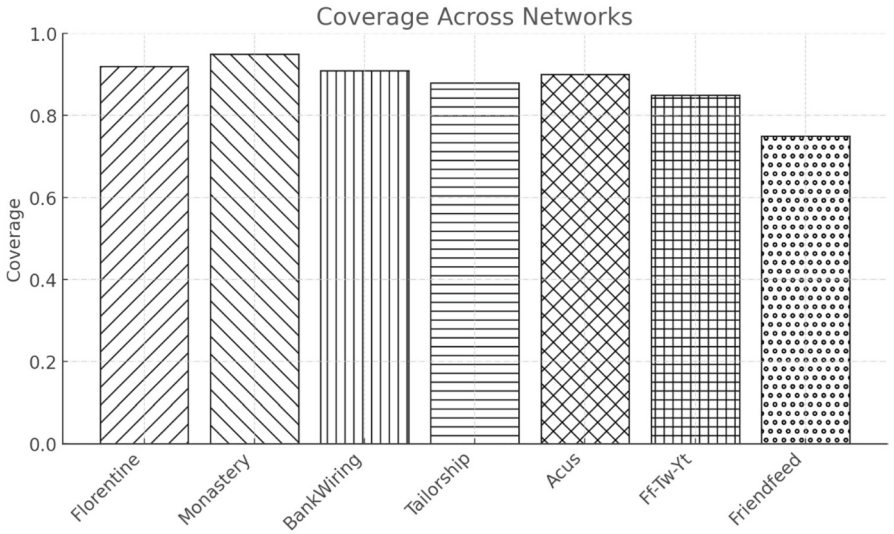


Fig. 11 The results for Coverage for real multilayer networks

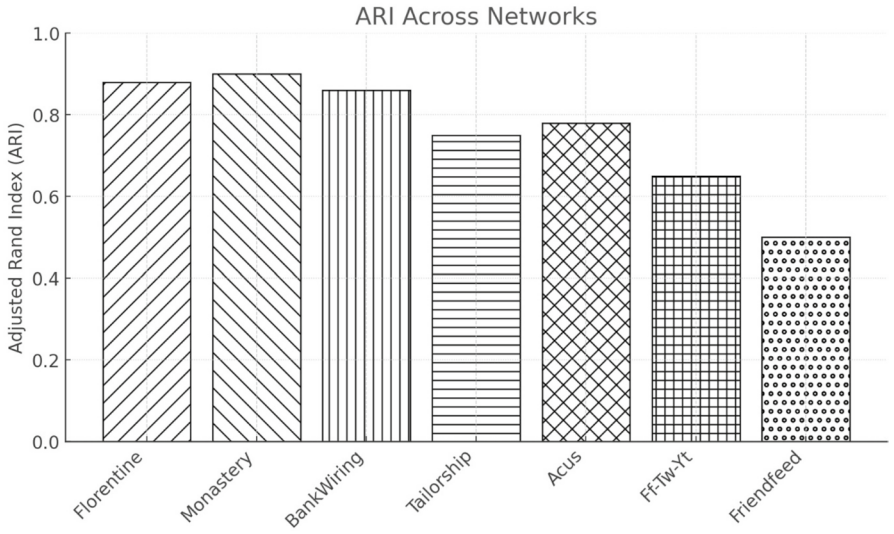
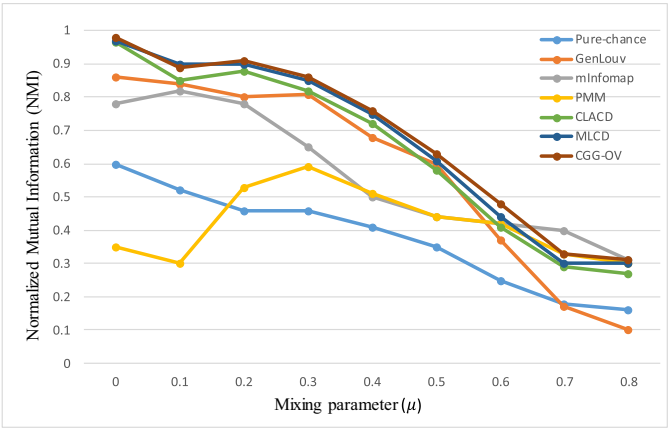


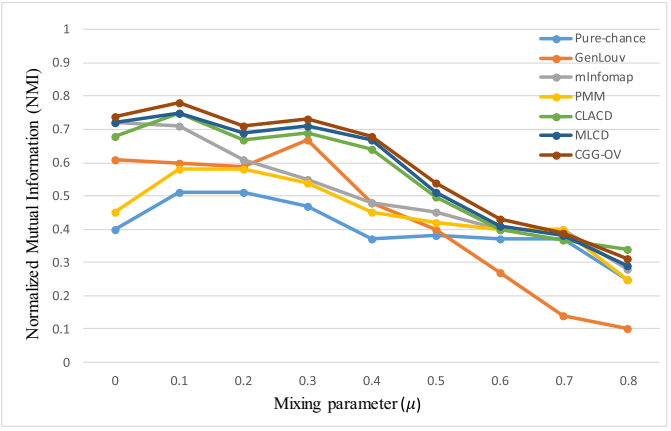
Fig. 12 The results for Adjusted Rand Index for real multilayer networks

8.1.5 Experiment VI

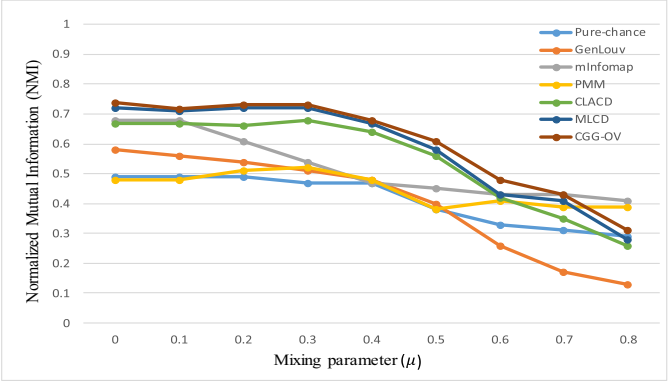
The aim of this experiment is to assess the performance of the CGG-OV algorithm in detecting overlapping communities based on modularity. The obtained results are compared with those of well-known algorithms for overlapping community detection, such as EMCD [64], GenLouv[65], mInfomap[66], PMM[67], CLACD



OM=1

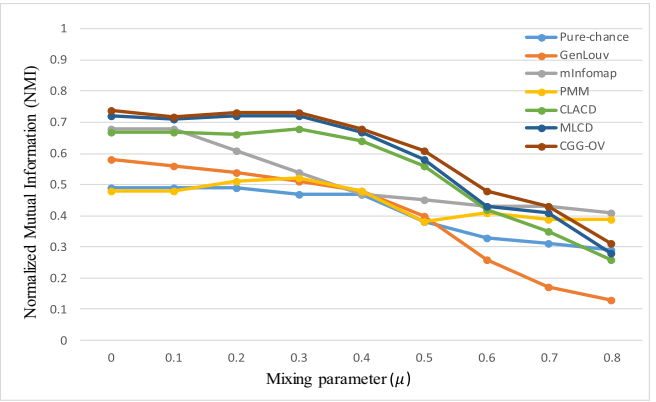


OM=2

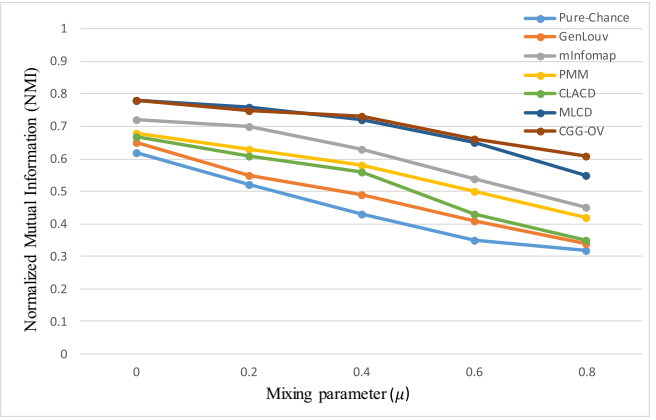


OM=3

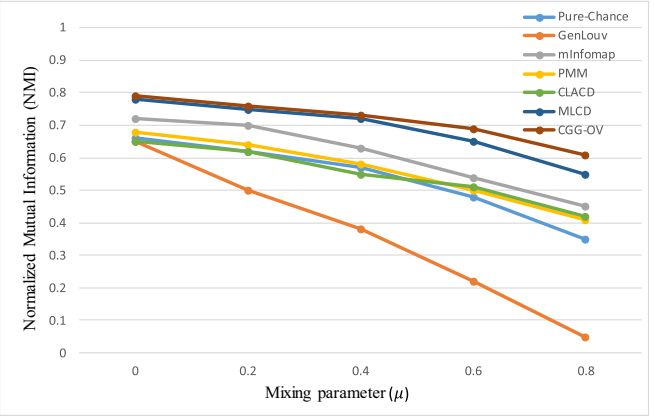
Fig. 13 Comparison of quality of the obtained results in terms of NMI on synthetic multilayer networks by varying number of memberships nodes OM = {1–6} in different mixing parameters



OM=4



OM=5



OM=6

Fig. 13 (continued)

[10] and MLCD [68]. The evaluation presented in Table 4 demonstrates the effectiveness of various algorithms in maximizing modularity (Q) across real-world networks, with the performance assessed based on maximum (Max) and average (Avg) modularity. Among the algorithms, CGG-OV consistently outperforms others, achieving the highest Max modularity across most networks, including Florentine, Monastery, Bankwiring, Tailorship, and Acus. This indicates its strong capability in optimizing community structures. Additionally, CGG-OV exhibits robust Avg modularity scores, often matching or exceeding competitors like GenLouv and MLCD, highlighting its stability and reliability. In comparison, the baseline Pure-Chance (CGG) algorithm achieves significantly lower modularity values, emphasizing the improvements introduced by CGG-OV. GenLouv performs well in Avg modularity and is competitive in Max modularity but is slightly outperformed by CGG-OV in key networks. On the other hand, M-Infomap struggles with both Max and Avg modularity, particularly in networks like Acus and FF-Tw-Yt, where its values are notably lower than other algorithms. PMM, CLACD, and MLCD show mixed performance, with MLCD being competitive in some networks but generally unable to surpass CGG-OV. In network-specific trends, CGG-OV excels in larger, complex networks such as Friendfeed and FF-Tw-Yt, achieving the highest modularity scores, while maintaining consistent results across simpler networks like Florentine and Monastery. These findings suggest that CGG-OV is not only an improvement over its baseline but also a robust and effective alternative to other state-of-the-art methods. The CGG-OV algorithm provides superior performance, particularly its ability to maximize modularity across diverse networks, and position it as a reliable solution for community detection tasks. Moreover, the table provides

Table 4 Evaluation of CGG-OV algorithms on real-world networks based on maximum (Max) and average (Avg.) modularity

Networks	Q	Pure-Chance(CGG)	GenLouv	M-Infomap	PMM	CLACD	MLCD	CGG-OV
Florentine	MAX	0.411	0.530	0.548	0.456	0.546	0.591	**0.592**
	AVG	0.405	0.527	0.542	0.452	0.542	0.589	0.591
Monastery	MAX	0.211	0.411	0.241	0.241	0.398	0.416	**0.419**
	AVG	0.204	0.402	0.238	0.238	0.397	0.413	0.418
Bankwiring	MAX	0.371	0.526	**0.919**	0.40	0.512	0.526	0.526
	AVG	0.371	0.521	0.908	0.394	0.511	0.522	0.526
Tailorship	MAX	0.141	0.358	0.174	0.174	0.491	0.358	**0.515**
	AVG	0.135	0.348	0.168	0.172	0.489	0.353	0.514
Acus	MAX	0.324	0.633	**0.826**	0.372	0.799	0.633	0.801
	AVG	0.319	0.621	0.824	0.368	0.792	0.631	0.801
Ff-Tw-Yt	MAX	0.390	0.497	0.286	0.238	0.488	0.497	**0.498**
	AVG	0.357	0.478	0.282	0.234	0.487	0.493	0.496
Friendfeed	MAX	0.529	0.582	0.482	-	0.512	0.582	**0.586**
	AVG	0.523	0.578	0.478	-	0.508	0.581	0.581

The best-performing values for each metric or comparison row

evidence of the challenges faced by algorithms like M-Infomap and PMM, which tend to underperform in modularity metrics, further underscoring CGG-OV's advantages in this domain (Tables 3 and 5).

8.1.6 Experiment VII

The statistical analysis of community detection algorithms using the t-test reveals significant variations in performance across different methods. Among the evaluated algorithms, GenLouv ($p=0.002$), CLACD ($p=0.020$), MLCD ($p=0.001$), and CGG-OV ($p=0.008$) exhibit statistically significant results, indicating their effectiveness in capturing community structures. Notably, MLCD achieves the highest statistical significance ($T=5.636$, $p=0.001$), suggesting superior performance in the given experimental conditions. Conversely, M-Infomap ($p=0.165$) and PMM ($p=0.869$) fail to achieve significance, implying that their performance may be more variable or less distinguishable from the null hypothesis. The results indicate that modularity-driven and learning-based approaches, such as MLCD and GenLouv, tend to outperform other methods, while certain algorithms may require more favorable conditions to exhibit meaningful results. Then, the findings validate the effectiveness of specific community detection techniques while highlighting the potential limitations of others, guiding future refinements in multilayer network analysis.

9 Discussion

A significant challenge associated with the proposed model lies in its extensive parameterization, particularly the integration of multiple learning automata with independent learning rates. While this level of flexibility enhances adaptability and enables fine-grained decision-making, it simultaneously introduces substantial difficulties in parameter tuning. The process of selecting optimal parameters, such as learning rates and automata configurations, is inherently dependent on the structure of the underlying network, making it difficult to establish a universal guideline applicable across various network types. In practical implementations, the optimal parameter settings are influenced by multiple factors, including network topology,

Table 5 Statistical significance analysis of community detection algorithms using t-Test

Algorithms	T-Statistic	P Value	Significance ($\alpha = 0.05$)
GenLouv	**5.192**	**0.002**	**Significant**
M-Infomap	1.582	0.165	Not Significant
PMM	0.174	0.869	Not Significant
CLACD	**3.136**	**0.020**	**Significant**
MLCD	**5.636**	**0.001**	**Significant**
CGG-OV	**3.935**	**0.008**	**Significant**

The best-performing values for each metric or comparison row

connectivity patterns, and the dynamic behavior of agents. The combinatorial nature of the parameter space further exacerbates the complexity of optimization, rendering it computationally demanding.

From a theoretical perspective, determining the optimal parameter values constitutes an NP-hard problem, as it necessitates exploring an exponentially large space of potential configurations. This complexity arises due to the intricate interplay among multiple interdependent factors that influence the convergence properties and whole performance of the learning automata. Furthermore, the inherent nonlinearity of the learning process complicates direct analytical derivation of optimal parameter settings. While heuristic approaches, such as grid search and metaheuristic optimization techniques (e.g., genetic algorithms and particle swarm optimization), offer potential solutions for identifying near-optimal parameter values, these methods often entail significant computational overhead and may not generalize effectively across diverse network structures.

Given these challenges, a promising avenue for future research involves the development of adaptive or self-tuning mechanisms capable of dynamically adjusting learning parameters based on observed performance metrics. Machine learning-based meta-optimization techniques, such as reinforcement learning and Bayesian optimization, could be employed to automate the parameter selection process, thereby mitigating the reliance on manual tuning. Additionally, a rigorous investigation into theoretical bounds on parameter sensitivity across different network classes could provide valuable insights into the design of more robust and efficient learning automata frameworks. Addressing these challenges would not only enhance the practical applicability of the proposed model but also contribute to the broader theoretical understanding of parameter selection in learning automata-based systems.

10 Conclusion

In this paper, we propose a new model of Goore Game called Cellular Goore Game with multiple learning automaton in each cell (CGG). We extended the primary definitions of covergence in the theory of learning automata for Cellular Goore Game with multiple LAs and therefore based on the new definitions some studied were conducted. The analytical studies confirmed that the CGG with multiple LAs aid of L_{R-I} learning algorithm is converges. Moreover, we have provided an entropy metric to study the players' behavior in each cell of the models and show that as the algorithm processed, the total entropy of the player's approach to zero. The numerical experiments were provided to support the theoretical results. To show the potential of this model in designing algorithms, an overlapping community detection problem called CGG-OV has been studied. Moreover, to show the superiority of the CGG-OV, the algorithm compared with six different algorithms which are two different learning automata based algorithms CLACD, Pure-chance and the obtained results shows the superiority of the proposed model in terms of Normalized mutual information and Modularity value.

The proposed model aims to tackle complex learning scenarios within specific application domains, resulting in inherently greater complexity compared to naïve cellular Goore Game model. However, a critical concern in the CGG-MLA model is its fully-parameterized system, stemming from multiple automata and independent learning rates. This aspect is particularly significant in convergence performance, an area that remains relatively unexplored within this context. The presence of learning automata in the CGG-MLA model suggests that a low learning rate may slow convergence, while a high rate can lead to local minima. Therefore, future work will focus on investigating new policies for tuning the learning rate of CGG-MLA through mathematical analysis.

Author contribution A. Mohammad Mehdi Daliri Khomami: Conceptualization, Methodology, Software, Formal analysis, Funding acquisition, Investigation, Visualization, Writing, Review, and editing – original draft, Writing – review & editing. B. Ali Mohammad Saghiri: Conceptualization, Editing, and review. A.B. Mohammad Mehdi Daliri Khomami and Ali Mohammad Saghiri Editing, and review. C. Reza Meybodi: Conceptualization, Methodology, Supervision, Review, and editing.

Data availability The data underlying this article are available in the article and in its online supplementary material.

Declarations

Conflict of interests The authors declare no competing interests.

References

1. Schönfisch B, de Roos A (1999) Synchronous and asynchronous updating in cellular automata. Biosystems 51(3):123–143
2. Beigy H, Meybodi MR (2004) A mathematical framework for cellular learning automata. Adv Complex Syst 7(03n04):295–319
3. Kari J (1990) Reversibility of 2D cellular automata is undecidable. Physica D 45(1–3):379–385
4. Lakshmivarahan S, Thathachar MAL (1976) Bounds on the convergence probabilities of learning automata. IEEE Trans Syst, Man, Cybern-Part A: Syst Humans 6(11):756–763
5. Beigy H, Meybodi MR (2008) Asynchronous cellular learning automata. Automatica 44(5):1350–1357
6. Rezvanian A, Saghiri AM, Vahidipour SM, Esnaashari M, and Meybodi MR (2018) *Recent Advances in Learning Automata*, vol. 754. in Studies in Computational Intelligence, vol. 754. Cham: Springer International Publishing, https://doi.org/10.1007/978-3-319-72428-7.
7. Beigy H, Meybodi MR (2007) Open synchronous cellular learning automata. Adv Complex Syst 10(04):527–556
8. Beigy H, Meybodi MR (2009) Cellular learning automata with multiple learning automata in each cell and its applications. IEEE Trans Syst, Man, Cybern Part B (Cybernetics) 40(1):54–65
9. Esnaashari M, Meybodi MR (2014) Irregular cellular learning automata. IEEE Trans Cybern 45(8):1622–1632
10. 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. https://doi.org/10.1016/j.jocs.2017.10.009
11. Esnaashari M, Meybodi MR (2013) Deployment of a mobile wireless sensor network with k-coverage constraint: a cellular learning automata approach. Wireless Netw 19(5):945–968. https://doi.org/10.1007/s11276-012-0511-7

12. 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
13. Saghiri AM, Meybodi MR (2016) An approach for designing cognitive engines in cognitive peer-to-peer networks. J Netw Comput Appl 70:17–40
14. Saghiri AM, Meybodi MR (2018) On expediency of closed asynchronous dynamic cellular learning automata. J Comput Sci 24:371–378
15. Oommen BJ and Granmo OC (2009) Learning automata-based solutions to the Goore game and its applications. *Game Theory: Strategies, Equilibria, and Theorems*, p. 183
16. Khomami MMD, Meybodi MR, Ameri R (2022) Cellular goore game with application to finding maximum clique in social networks. J Comput Des Eng 9(3):966–991
17. Ameri R, Meybodi MR, Daliri Khomami MM (2022) Cellular Goore Game and its application to quality-of-service control in wireless sensor networks. J Supercomput 78(13):15181–15228
18. Ameri R, Meybodi MR (2024) An improved cellular goore game-based consensus protocol for blockchain. Cluster Comput 27(5):6843–6868
19. Khomami MMD, Meybodi MR, Rezvanian A (2024) Exploring social networks through stochastic multilayer graph modeling. Chaos, Solitons Fractals 182:114764
20. Khomami MMD, Meybodi MR, Rezvanian A (2025) A cellular goore game-based algorithm for finding the shortest path in stochastic multi-layer graphs. J Supercomput 81(1):1–50
21. Beigy H, Meybodi MR (2006) Utilizing distributed learning automata to solve stochastic shortest path problems. Intern J Uncertain Fuzziness Knowl-Based Syst 14(05):591–615
22. Mollakhalili Meybodi MR, Meybodi MR (2014) Extended distributed learning automata: an automata-based framework for solving stochastic graph optimization problems. Appl Intell 41(3):923–940. https://doi.org/10.1007/s10489-014-0577-2
23. Moradabadi B, Meybodi MR (2018) Wavefront cellular learning automata. Chaos: An Interdiscip J Nonlinear Sci 28(2):021101
24. Vahidipour SM, Meybodi MR, Esnaashari M (2017) Cellular adaptive Petri net based on learning automata and its application to the vertex coloring problem. Discret Event Dyn Syst 27(4):609–640
25. Narendra KS and Thathachar MA (2012) *Learning automata: an introduction*. Courier Corporation.
26. Rezvanian A, Saghiri AM, Vahidipour SM, Esnaashari M, and Meybodi MR (2018) *Recent Advances in Learning Automata*, vol. 754. in Studies in Computational Intelligence, vol. 754. Cham: Springer International Publishing, pp. 21–88. https://doi.org/10.1007/978-3-319-72428-7.
27. Zhao Y, Jiang W, Li S, Ma Y, Su G, Lin X (2015) A cellular learning automata based algorithm for detecting community structure in complex networks. Neurocomputing 151:1216–1226. https://doi.org/10.1016/j.neucom.2014.04.087
28. Mozafari M, Shiri ME, Beigy H (2015) A cooperative learning method based on cellular learning automata and its application in optimization problems. J Comput Sci 11:279–288
29. Esnaashari M, Meybodi MR (2008) A cellular learning automata based clustering algorithm for wireless sensor networks. Sens Lett 6(5):723–735
30. Saghiri AM, Meybodi MR (2018) Open asynchronous dynamic cellular learning automata and its application to allocation hub location problem. Knowl-Based Syst 139:149–169
31. Esnaashari M, Meybodi MR (2018) Dynamic irregular cellular learning automata. J Comput Sci 24:358–370
32. Saghiri AM, Meybodi MR (2017) A closed asynchronous dynamic model of cellular learning automata and its application to peer-to-peer networks. Genet Program Evolvable Mach 18(3):313–349. https://doi.org/10.1007/s10710-017-9299-7
33. Vafashoar R, Meybodi MR (2019) Reinforcement learning in learning automata and cellular learning automata via multiple reinforcement signals. Knowl-Based Syst 169:1–27
34. Vafashoar R, Morshedlou H, Rezvanian A, and Meybodi MR (2021) Learning from Multiple Reinforcements in Cellular Learning Automata. In: *Cellular Learning Automata: Theory and Applications*, vol. 307, in Studies in Systems, Decision and Control, vol. 307, Cham: Springer International Publishing, pp. 111–156. https://doi.org/10.1007/978-3-030-53141-6_3.
35. Mucha PJ, Richardson T, Macon K, Porter MA, Onnela JP (2010) Community structure in time-dependent, multiscale, and multiplex networks. Science 328(5980):876–878
36. Jia J, Liu P, Du X, Zhang Y (2021) Multilayer social network overlapping community detection algorithm based on trust relationship. Wirel Commun Mob Comput 2021:1–14
37. Zhu Z, Yuan G, Zhou T, and Cao J (2025) Community Detection for Heterogeneous Multiple Social Networks," *IEEE Transactions on Computational Social Systems*, 2024, Accessed: Jan. 22, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10552103/

38. Wang P and Chan WKV (2025) A multilayer community detection algorithm based on aggregation in social Internet of things. In: *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, IEEE, 2022, pp. 610–614. Accessed: Jan. 22, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9778307/
39. Rani S, Kumar M (2022) Ranking community detection algorithms for complex social networks using multilayer network design approach. Int J Web Inf Syst 18(5/6):310–341
40. Kashef R (2021) Detecting Overlapping Communities in Social Networks Using A Modified Segmentation by Weighted Aggregation Approach. In: *International Conference on Data Science, E-learning and Information Systems 2021*, Ma'an Jordan: ACM, Apr. 2021, pp. 60–69. https://doi.org/10.1145/3460620.3460632.
41. Zhang J, Wang F, Zhou J (2024) Community detection based on nonnegative matrix tri-factorization for multiplex social networks. J Complex Netw 12(2):cnae012
42. Huang X, Chen D, Ren T, Wang D (2021) A survey of community detection methods in multilayer networks. Data Min Knowl Disc 35(1):1–45. https://doi.org/10.1007/s10618-020-00716-6
43. Liu W, Suzumura T, Ji H, Hu G (2018) Finding overlapping communities in multilayer networks. PLoS ONE 13(4):e0188747
44. Huang L, Wang C-D, Chao H-Y (2019) oComm: overlapping community detection in multi-view brain network. IEEE/ACM Trans Comput Biol Bioinf 18(4):1582–1595
45. Dong R, Yang J, Chen Y (2020) Overlapping community detection in weighted temporal text networks. IEEE Access 8:58118–58129
46. Gupta S, Kumar P (2020) An overlapping community detection algorithm based on rough clustering of links. Data Knowl Eng 125:101777
47. Yang X, Cui L, and Liu Y (2024) Application Analysis of Overlapping Community Detection Algorithms for Multidimensional Network Big Data and IoT," *Wireless Communications and Mobile Computing*, vol. 2022, 2022, Accessed: Feb. 11, [Online]. Available: https://www.hindawi.com/journals/wcmc/2022/1172186/
48. Gao X, Zheng Q, Verri FAN, Rodrigues RD, and Zhao L (2019) Particle Competition for Multilayer Network Community Detection. In: *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, Zhuhai China: ACM, Feb. pp. 75–80. https://doi.org/10.1145/3318299.3318320.
49. Li H, Xu W, Qiu C, and Pei J (2024) Fast Markov clustering algorithm based on belief dynamics. *IEEE Transactions on Cybernetics*, Accessed: Feb. 09. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9693335/
50. Song S, Feng Y, Xu W, Li H-J, Wang Z (2022) Evolutionary prisoner's dilemma game on signed networks based on structural balance theory. Chaos, Solitons Fractals 164:112706
51. Li H-J et al (2022) Characterizing the fuzzy community structure in link graph via the likelihood optimization. Neurocomputing 512:482–493
52. Beigy H and Meybodi MR (2002) A New Distributed Learning Automata Based Algorithm For Solving Stochastic Shortest Path Problem. In *JCIS*, Citeseer, pp. 339–343.
53. Vahidipour SM, Meybodi MR, Esnaashari M (2017) Finding the shortest path in stochastic graphs using learning automata and adaptive stochastic petri nets. Intern J Uncertain Fuzziness Knowl-Based Syst 25(03):427–455
54. 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
55. 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. Phil Trans R Soc A 378(2182):20190593
56. 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
57. Torkestani JA, Meybodi MR (2010) An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata. Comput Netw 54(5):826–843
58. 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
59. Rezvanian A, Meybodi MR (2015) Finding minimum vertex covering in stochastic graphs: a learning automata approach. Cybern Syst 46(8):698–727
60. Thathachar MAL, Arvind MT (2013) Solution of Goore game using modules of stochastic learning automata. J Indian Inst Sci 77(1):47
61. Rhouma D, Romdhane LB (2014) An efficient algorithm for community mining with overlap in social networks. Expert Syst Appl 41(9):4309–4321

62. Zhang Y, Fitch P, Vilas MP, Thorburn PJ (2019) Applying multi-layer artificial neural network and mutual information to the prediction of trends in dissolved oxygen. Front Environ Sci 7:46
63. De Domenico M, Porter MA, Arenas A (2015) MuxViz: a tool for multilayer analysis and visualization of networks. J Complex Netw 3(2):159–176
64. Interdonato R, Tagarelli A, Ienco D, Sallaberry A, Poncelet P (2017) Local community detection in multilayer networks. Data Min Knowl Disc 31(5):1444–1479
65. Jutla IS, Jeub LG, and Mucha PJ (2011) A generalized Louvain method for community detection implemented in MATLAB. URL http://netwiki. amath. *unc. edu/GenLouvain*
66. De Domenico M, Lancichinetti A, Arenas A, Rosvall M (2015) Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. Phys Rev X 5(1):011027
67. Tang L, Wang X, and Liu H (2009) Uncoverning groups via heterogeneous interaction analysis. In: *2009 Ninth IEEE International Conference on Data Mining*, IEEE, 2009, pp. 503–512.
68. Shahmoradi MR, Ebrahimi M, Heshmati Z, Salehi M (2019) Multilayer overlapping community detection using multi-objective optimization. Futur Gener Comput Syst 101:221–235