A Cellular Goore Game-Based Algorithm for Finding Shortest Path in Stochastic Multi-Layer Graphs

Mohammad Mehdi Daliri Khomami
Soft computing laboratory, Department of Computer Engineering, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
m.daliri@aut.ac.ir

Mohammad Reza Meybodi
Soft computing laboratory, Department of Computer Engineering, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
mmeybodi@aut.ac.ir

Alireza Rezvanian
Department of Computer Engineering, University of Science and Culture, Tehran, Iran
rezvanian@usc.ac.ir

Abstract: The shortest path problem in stochastic graphs has been extensively studied, with numerous algorithms proposed using various learning automata models. However, the dynamic nature, diverse individual characteristics, and inherent uncertainties of social interactions necessitate the adoption of stochastic multi-layer social network modeling. This approach provides deeper insights into the complex relationships within social networks. When formulated as a stochastic multi-layer graph, key elements such as the shortest path require redefinition to account for these complexities. This paper explores the shortest path problem in stochastic multi-layer graphs and introduces a novel algorithm based on the Cellular Goore Game (CGG) to address this challenge. The proposed CGG-based algorithm leverages learning automata and extensive edge sampling to determine the optimal path efficiently. By integrating learning automata and selectively sampling from relevant sections of the graph, the algorithm significantly reduces computational complexity. Experimental results on stochastic multi-layer graphs highlight the effectiveness of the proposed algorithm, demonstrating substantial improvements across multiple metrics, including sampling ratio, shortest path ratio, average iterations, and convergence rate.

Keywords— ***Stochastic Multi-layer Graphs, multi-layer shortest path problem, Sampling Method, Cellular Goore Game, Learning Automata.***

1 INTRODUCTION

Social networks provide platforms for individuals to connect, communicate, and share information with others [1]. They facilitate the formation and maintenance of personal and professional relationships, enabling people to stay connected across distances and share ideas, experiences, and resources. Moreover, social networks serve as sources of information and knowledge [2]. Through social media platforms, online communities, and professional networks, individuals can access news, expertise, and insights on various topics [3]. Social networks also enable the rapid dissemination of information, making them valuable tools for spreading awareness, mobilizing

support, and sharing important messages. The analysis of social networks has been studied by single-layer graphs [4].

Studying social networks as single-layer graphs is a common approach in network analysis and social network research. In this context, each individual in the social network is represented as a node and the relationships or connections between individuals are represented as edges. Studying social networks as a single-layer graph model provides a simplified representation of the social structure [5]. By employing single-layer graph modeling for social networks, the analysis of various patterns, such as connections and interactions within the network, becomes more accessible. However, representing social networks as a single-layer graph may bring some drawbacks. The main drawback of representing social networks as single-layer graph modeling is that it oversimplifies the complexity and richness of real-world social interactions[6]. Social networks are often multidimensional, comprising various types of relationships and contexts, which cannot be fully captured by a single-layer graph modeling [7]. Condensing all relationships into a single layer may result in the loss of important characteristics and details related to different types of connections. This oversight can lead to an incorrect analysis of the networks under study. For example, in a social network, individuals may have diverse relationships like family ties, friendships, professional collaborations, or online interactions. Treating all these relationships as equal and representing them through a single-layer graph model may fall short of capturing the nuanced variations in strengths, dynamics, and contexts inherent in these connections [8]. Real-world social interactions occur within specific contexts such as work, school, online communities, or geographical locations [9], [10], [11]. Each context can have its own set of relationships and dynamics. Representing social networks as single-layer graphs overlooks the crucial aspect of context, thereby constraining our comprehension of how social relationships exhibit variations across diverse environments. This simplification undermines the depth of understanding needed to grasp the intricate dynamics and nuances inherent in social interactions within varying environments. Moreover, social networks are dynamic, with relationships evolving [12]. Single-layer graph representations typically assume static networks, neglecting the temporal dimension inherent in dynamic social structures. This oversight limits our ability to capture the evolving nature of social interactions over time, thereby hindering a comprehensive understanding of the temporal dynamics shaping network behavior [13]. However, the time-varying nature and different types of relationships can significantly influence the structural configuration and operational dynamics of the network. Therefore, a new phenomenon known as multi-layer social networks is emerging, which represents an opportunity to analyze social networks in-depth [5].

Multi-layer social networks, also known as multiplex or multimodal social networks, are a type of social network where individuals or entities are connected through multiple types or layers of relationships[14]. In a single-layer social network, relations are typically represented by a single type of connection, such as friendship or professional connections. However, in multi-layer social networks, individuals can have relationships across different dimensions or contexts. Each layer in a multi-layer social network represents a distinct type of relationship or interaction between individuals [7]. These layers can capture various aspects of social connections, such as friendship[15], collaboration [16], communication [17], co-location[18], or shared interests[19]. As an illustration, within the framework of multi-layer social network modeling, layers may be indicative of online friendships on a social media platform, collaborative partnerships in academic co-authorships, and joint projects within a professional workplace. The various layers of a multi-layer social network are frequently interconnected, enabling the investigation of relationships

across multiple dimensions. This interconnection facilitates a more comprehensive understanding of the intricate interactions and dependencies embedded within a social system [20].
Human behavior, owing to its time-varying and dynamic nature, is a complex and multidimensional phenomenon that cannot be fully captured by a simple snapshot of a multi-layer graph. Consequently, while a multi-layer graph may reveal relationships between users at a specific point in time, it fails to account for the dynamic and evolving nature of human interactions.

On the contrary, a stochastic multi-layer social network offers a more precise and intricate representation of human behavior and provides a more suitable framework for modeling the unpredictable and time-varying nature of social networks [21]. It allows for the representation of dynamic interactions among individuals within the context of a social network[22]. Social networks represent intricate systems characterized by a heightened degree of interconnections, dynamic attributes, unpredictability, and temporal dynamics. Stochastic multi-layer graphs provide such a framework by allowing the representation of multiple layers of connectivity between individuals in a social network. Each layer signifies a distinct aspect of the network, including the frequency and duration of interactions, the strength of ties between individuals, and the content of the communication [23]. This nuanced approach ensures a more comprehensive understanding of the intricate dynamics inherent in social networks.

Through the separate modeling of distinct aspects of the network, stochastic multi-layer graphs can effectively capture the heterogeneity and intricacy inherent in social network activities. Moreover, stochastic multi-layer graphs demonstrate the capability to integrate unpredictability and uncertainty into their modeling frameworks. This is crucial for social network activities inherently exhibit unpredictability and a time-varying nature. In real-world applications, such as influence maximization and budget allocation, which hold critical importance within competitive environments, overlooking the dynamic processes inherent in human behavior within these applications can lead to unintended damage or negative consequences. Therefore, it is imperative to employ a stochastic multi-layer network as a modeling framework to enhance our understanding of human behavior.
The Goore Game (GG), as conceptualized by Tsetlin [24], represents a cooperative game scenario involving a group of players and a referee. In this unique setup, each player faces a binary choice, akin to a "Yes" or "No" decision[25]. The referee, equipped with an unimodal performance evaluation function denoted as G, assesses the collective votes submitted by the players. Specifically, during each iteration, players independently select one of their options, and the referee computes the resulting vote ratio, defined as the number of affirmative votes divided by the total number of votes cast. Subsequently, the referee then probabilistically rewards each player with a dollar based on the outcome of the evaluation function $G(f)$, where f represents the computed vote ratio. Conversely, players may also incur a dollar loss with a complementary probability of $1 - G(f)$. Guided by their gains and losses, players subsequently make independent decisions regarding their votes in the ensuing iterations.
Learning Automata (LAs) serve as models for adaptive decision-making within uncertain and random environments. Each LA possesses a finite set of actions, initially assigned equal probabilities, unbeknownst to the automaton, of being rewarded or penalized by its environment. The primary objective of an LA is to iteratively learn the optimal action, defined as the action most likely to yield a reward. Notably, an optimal action is characterized by its highest probability of garnering rewards from the environment. A remarkable attribute of LAs is their capability to serve

as simple agents for executing complex tasks, thereby facilitating the handling of intricate learning problems [25]. The true potential of LAs is unlocked when multiple automata collaborate as a cohesive team, as exemplified in the Goore game (GG). Such collaborative interactions may manifest in various forms across different game scenarios, encompassing both cooperative and competitive dynamics.

The Cellular Goore Game (CGG), introduced by Khomami et al [26], offers a modeling framework suitable for systems composed of simple, identical components engaged in localized interactions aimed at optimizing various criteria. This model holds relevance for systems characterized by extensive collections of straightforward objects interacting locally, hence its designation as "cellular," resembling points within a structural framework. CGG functions as a network of Goore Games (GGs), where each node in the network, or a designated subset, acts as a referee engaging in GGs with neighboring players. Like the GG framework, players independently select actions based on individual gains and losses from adjacent referees, lacking knowledge of others' actions or the reasons behind rewards or penalties. Notably, each node in CGG can simultaneously serve as both player and referee. During each round, nodes act as referees, participating in GGs with neighboring cells as voters, possibly engaging in multiple simultaneous GGs. Each referee follows an unimodal performance criterion, optimized when the fraction of neighboring players casting "Yes" votes matches a predetermined threshold. Following independent and simultaneous voting by players, referees compute the fraction of "Yes" votes from adjacent player cells. Referees then reward players based on this fraction, with players subsequently deciding how to cast votes for the next round based on received rewards.

Studying the shortest path problem in stochastic multilayer graphs has garnered significant interest due to its practical importance in modeling and solving real-world problems characterized by complexity and uncertainty. Unlike traditional single-layer networks, multilayer graphs can represent various types of connections and interactions across different layers, capturing the multifaceted nature of systems such as transportation networks, communication infrastructures, and social interactions. When these graphs incorporate stochastic elements, such as time-dependent edge weights and variable connectivity, they become highly effective for simulating environments where uncertainty is inherent. Identifying the shortest path in such complex structures is crucial for optimizing processes such as routing, resource allocation, and information dissemination. The main challenge lies in balancing computational efficiency with accuracy while accounting for the probabilistic nature of connections and interactions. Addressing this problem can lead to more robust and adaptive solutions, thereby enhancing decision-making capabilities in dynamic and unpredictable settings. Real-world applications, including transportation networks[27], biological networks[28], and social networks[29], often exhibit multiple layers of interactions and uncertainties. Modeling these systems as stochastic multilayer graphs provides a more realistic and accurate representation of problems characterized by their time-varying nature, uncertainties, and unpredictability.
In recent studies [26], [30], the Cellular Goore Game model is a powerful tool for tackling complex problems within the realm of graph theory. The Cellular Goore Game is particularly well-suited for addressing the shortest path problem in stochastic multilayer graphs due to its unique attributes that align with the challenges posed by these complex systems. One of its key strengths is its decentralized problem-solving approach, where local interactions between cells enables the exploration of intricate networks without the need for centralized control. This decentralized

mechanism is especially valuable in stochastic multilayer graphs, as it allows for effective management and adaptation to the varying conditions across different layers. Furthermore, the adaptability to dynamic environments inherent in the Cellular Goore Game makes it well-equipped to handle the randomness and time-dependent changes found in such graphs, including fluctuating edge weights and connectivity. Its iterative and adaptive nature facilitates real-time recalibration of paths as network conditions evolve. Another advantage is its parallel processing capability, which is a direct result of the cellular automaton framework. This parallelism allows for the simultaneous exploration and updating of multiple potential paths, significantly improving computational efficiency and enabling the identification of optimal or near-optimal paths even in large and complex multilayer graphs. Additionally, the Cellular Goore Game's structure can be adapted to model inter-layer interactions, capturing the dependencies between different layers in a multilayer graph. By defining rules that allow cells to exchange information across layers, the game can facilitate a comprehensive analysis of potential paths that account for cross-layer influences. The game's ability to handle uncertainty and stochastic behavior further enhances its applicability, as it can explore multiple potential solutions and adapt to probabilistic changes in the graph. This results in robust pathfinding strategies that effectively manage uncertainty. Lastly, the game can be designed to balance exploration and exploitation, which is essential in stochastic environments where both discovering new paths and optimizing known ones are crucial for finding the shortest and most reliable routes. In summary, the Cellular Goore Game offers the flexibility, adaptability, and efficiency needed to navigate the complex, interconnected, and uncertain nature of stochastic multilayer graphs, making it an effective tool for shortest path determination in such environments. In the context of relationship analysis within social networks, shortest-path algorithms play a crucial role in assessing the strength or proximity of connections between individuals. These algorithms help quantify the closeness of relationships, providing insights into how information, influence, or interactions propagate throughout the network. Finding the shortest path between two individuals, the algorithm can provide insights into how connected they are and the number of intermediaries between them [31]. Moreover, for influence flow, the shortest path helps to identify influential individuals or entities within a social network. By calculating the shortest paths from influential sources (e.g., celebrities, opinion leaders) to other individuals, we can understand how information or influence spreads through the network [32]. Moreover, in the context of community detection, the utilization of the shortest path can facilitate the identification of communities or clusters within a social network [33]. Through the examination of shortest paths as potential communities between nodes, one can discern groups of closely interconnected individuals who share common interests, affiliations, or characteristics. Hence, this paper introduces a novel algorithm for finding the shortest path in stochastic multi-layer graphs using the Cellular Goore Game (CGG) for the first time. The proposed algorithm utilizes the stochastic multi-layer graph as the random environment for the CGG. In this context, the environment's response is determined based on the sampled length of a specific path. By incorporating the sampled path length, the environment effectively selects an appropriate action or output in response to the chosen action. This response, whether positive or negative, subsequently leads to the rewarding or penalizing of actions taken along the traversed path. The evaluation of these actions establishes a direct association between the response outcome and the repercussions linked to the selected action. The proposed algorithm not only enhances the efficiency and accuracy of shortest path computations but also opens up new horizons for advancing research in network optimization and analysis.

The main contributions of this paper are summarized as follows:

- The paper introduces a novel algorithm, employing the Cellular Goore Game (CGG) framework, to address the shortest path problem in stochastic multilayer graphs for the first time.
- The proposed algorithm incorporates learning automata to enhance path determination through extensive edge sampling. By integrating learning automata into the CGG-based algorithm and strategically sampling from relevant sections of the stochastic multi-layer graph, the proposed approach effectively mitigates algorithmic complexity, offering a more efficient solution for identifying optimal paths.
- Experimental evaluations conducted on stochastic multi-layer graphs demonstrate the effectiveness of the proposed algorithm across various evaluation metrics, including sampling ratio, shortest path ratio, average number of iterations, and convergence proportion, thereby validating its practical utility and advantages over existing approaches.
- The paper provides new definitions for open research areas aimed at extending the study of shortest paths in stochastic multilayer graphs as future work.

The remainder of the paper is organized as follows: Section 2 provides a brief overview of recent related works. Section 3 offers some preliminaries about multi-layer graphs and shortest paths. Section 4 delves into the concepts of learning automata, Goore Game (GG), and Cellular Goore Game (CGG). The notions of stochastic multilayer graphs and shortest paths in stochastic multi-layer graphs are introduced in Section 5. Section 6 presents the CGG-based algorithm for finding the shortest path, while Section 7 introduces some enhancements to the proposed algorithm. In Section 8, the complexity analysis is discussed. Section 9 presents the simulation results, and Section 10 concludes the paper.

2 Related Works

The exploration of shortest-path algorithms in multilayer networks has garnered substantial interest due to their critical role in complex systems, including transportation, communication, and social networks. This section provides a short review of existing research focused on algorithms that solve the shortest path problem, detailing significant advancements, methodologies, and innovative approaches aimed at optimizing pathfinding across various graph structures and network topologies.

Dijkstra's algorithm remains a foundational approach for determining the shortest path between two nodes in a graph, utilizing an iterative method to update distances from the starting node until reaching the destination. The Bellman-Ford algorithm provides an alternative, capable of handling graphs with negative edge weights by iteratively relaxing edges to find the shortest path [34]. The A* algorithm extends these concepts through heuristic functions, combining estimated and actual distances for efficient pathfinding [35]. For all-pairs shortest path problems, the *Floyd-Warshall* algorithm computes distances by considering all possible intermediate nodes[36], while Johnson's algorithm handles graphs with negative weights by transforming them and applying *Dijkstra*'s algorithm[37]. Techniques such as bidirectional search enhance the efficiency of *Dijkstra*'s or A* algorithms by expanding the graph from both the source and target nodes simultaneously [38]. Multi-objective shortest-path problems involve optimizing various metrics (e.g., time and cost), employing techniques like Pareto optimization [40]. K-shortest path algorithms, such as Yen's algorithm, address alternative paths that are efficient but not necessarily the shortest [41]. For stochastic shortest path problems, where edge weights are random variables, Monte Carlo

simulations provide robust solutions [42]. Approximation algorithms yield near-optimal paths within a predefined margin[43], while distributed algorithms break down the problem for parallel processing across networked nodes [44]. Handling massive graphs (e.g., with MapReduce frameworks or platforms like Apache Spark GraphX) is essential for scaling up shortest-path algorithms [45]. Multi-agent pathfinding involves coordinating multiple agents to prevent congestion, utilizing decentralized methods [46].

In the context of multiplex networks, which represent layered interactions, the complexity increases as these networks require algorithms that account for both network heterogeneity and weight variations. This necessitates the integration of advanced methodologies, such as Pareto-based measures, to effectively manage the diverse interactions and optimize pathfinding across multiple layers [47] [48]. The study in [49], addresses the challenge of pathfinding in multi-layer networks, highlighting the complexities such as the need to traverse physical links multiple times, which can result in loops. It also notes that a segment of the shortest path does not necessarily qualify as the shortest path in isolation within this context. The problem is shown to be NP-complete, and two models for representing multi-layer networks are introduced: one focusing on devices and layers (Gl) and another based on devices and technology stacks (Gs). Each model has distinct algorithms for finding feasible paths, with their respective advantages and limitations discussed. The work in [48] presents a method for calculating the distribution and number of shortest paths in duplex Erdős–Rényi multiplex networks, which aids in understanding congestion in multimodal transportation networks. This analytical approach helps identify regions within a phase diagram where multiplex networks might become congested, with implications for dynamic processes dependent on shortest path routing in multi-layered systems. Khanda et al. [50] Introduce a parallel algorithmic framework for updating Single-Source Shortest Paths (SSSP) in large-scale dynamic networks, utilizing shared-memory and GPU platforms. This approach emphasizes the identification of affected network segments and efficient updates using a rooted tree data structure. In [51], a novel method for counting shortest paths in probabilistic networks is presented, addressing challenges posed by uncertain topologies. By employing x-polynomials within a bipartite graph framework, the authors accurately model path dependencies, applying this method to identify key genes and detect communities in gene networks, as demonstrated in analyses of the cell cycle pathway across cancer types. Research in [52], investigates information transport in multiplex networks comprising logical and physical layers. The study uses biased random walks to optimize the transmission and proposes an attack centrality for evaluating network robustness. Findings reveal that assortative coupling can enhance network efficiency compared to other configurations. This study presents a comprehensive framework for addressing the Markov Decision Process (MDP) problem using probabilistic formalism on a factor graph[53]. It investigates various algorithms, including standard sum-product (marginalization), dynamic programming, max-reward, and entropy methods. The methodologies are interpreted as distinct combination rules within two blocks of a factor graph in reduced normal form, highlighting the convergence of estimation and control. The review enhances classical methods through original parametric generalizations, yielding a suite of algorithms that can be implemented using a unified belief propagation framework. This approach provides decision-makers with improved options to adjust solution precision via entropic cost functions. The proposed solutions also advance online reinforcement learning algorithms that require V or Q-functions, balancing exploration and exploitation. Computational results from discrete grid simulations demonstrate that the max-product algorithm and its probabilistic variants converge more rapidly to steady-state configurations than conventional reward-based methods. A key advantage of the unified

formulation is its software transferability, allowing adaptation to decision-makers risk preferences. Additionally, simulation software for path planning on grids is available upon request, with future research aimed at extending this framework to continuous spaces, heuristic search approximations, and scenarios involving interacting agents. Finally [54], explores the single-agent framework to accommodate distributed goals, effectively incorporating elements commonly encountered in realistic environments, such as doors, benches, counters, and food sources. Building upon this foundation, we further developed the model to encompass multiple agents, wherein the Markov models of the agents operate concurrently. In our initial approach, each agent perceives the others as dynamic obstacles or potential goals. The resulting emergent behaviors exhibit a high degree of realism, suggesting that the implications of this robust framework warrant further investigation and exploration.

The studies reviewed in this section predominantly concentrate on the shortest path problem within the context of deterministic or deterministic graphs. This paper, however, pioneers the examination of the shortest path problem in stochastic multilayer graphs, representing a significant advancement in the field. Stochastic multilayer graphs offer a more nuanced framework that reflects real-world complexities, capturing the dynamic nature of individuals' behaviors, the specificity of interactions, and the time-varying characteristics inherent in social networking. Furthermore, this framework accommodates the simultaneous use of multiple social networks and their interconnections. To address the challenges posed by this novel context, the paper employs a Cellular Goore Game-based algorithm that leverages learning automata to effectively solve the shortest path problem in stochastic multilayer graphs. This innovative approach not only enhances our understanding of influence propagation in complex networks but also contributes to the development of more robust algorithms capable of navigating the intricacies of real-world social interactions.

3 PRELIMINARIES

In this section, our objective is to provide essential contextual information for the subsequent sections of this paper. We offer a concise exposition of the multi-layer graph, delving into the intricacies of the shortest path problem and exploring its associated variations.

3.1 Multi-layer Graph and Shortest Path

A multi-layer graph, denoted as $\mathcal{M} = (\mathcal{G}, \mathcal{C})$, is represented by $\mathcal{G} = \{G_q : q \in \{1, \dots, M\}\}$ where each $G_q = (V_q, E_q)$ signifies a layer of $\mathcal{M}$. In this representation, $E_q = V_q \times V_q$ represents the inter-layer edges within each G_q. The set $\mathcal{C}$ comprises edges connecting nodes from distinct layers G_q and G_k, where $q \neq k$. The elements of set $\mathcal{C}$ are referred to as crossed-layer elements, while each element within E_{qk}, where $q \neq k$, is designated as an interlayer edge. Cross-layer edges in multilayer graphs serve as connections between nodes belonging to different layers within the network. These edges play key roles in facilitating interactions and enhancing the understanding of complex systems. Moreover, cross-layer edges enable nodes in one layer to interact directly with nodes in another layer. This facilitates the exchange of information, resources, or influence between layers, promoting collaboration and cooperation within the network. Furthermore, each layer G_q is characterized by a set of nodes denoted as $V_q = \{v_1^q, \dots, v_{N_q}^q\}$. In the context of a single-layer graph, the shortest path is defined as the path between two vertices that minimizes the sum of edge weights. This optimal path represents the least traversal or distance required to navigate from one vertex to another along with the network.

Within a multi-layer graph, the concept of the shortest path pertains to the path characterized by the minimum total weight or distance between two nodes. This consideration incorporates the distinct layers or levels inherent in the structure of the graph. As an illustration, consider a transportation network where the layers signify various modes of transportation, including roads, railways, and air routes. In a multi-layer graph, determining the shortest path involves considering both the edges within each layer and the transitions between layers. The objective is to identify the path that minimizes the cumulative sum of edge weights or distances, facilitating movement across different layers of the network [55]. In the context of multi-layer social network analysis, the shortest path algorithms play a crucial role in examining social networks. These algorithms facilitate the identification of the shortest paths between individuals or groups, represented as a multi-layer graph. Such analysis contributes to comprehending influence patterns, and information flow dynamics, and aids in the detection of distinct communities within the network [47]. Furthermore, within the context of Internet search engines, algorithms play a crucial role in assessing the relevance and ranking of web pages for a particular search query. These algorithms take into account the shortest path between distinct web pages, utilizing links and connections as key factors in the evaluation process [56].

4 Learning Automata and Cellular Goore Game

In the domain of learning automata (LA), which involves a stochastic automaton interacting with a random environment as presented in Figure 1, the primary goal is to ascertain the optimal action from a predefined set of possible actions. During stage k, the automaton decides by choosing one action from a limited set of r actions denoted as $\alpha = (\alpha_1; \ldots; \alpha_r)$. This selection process is facilitated through the utilization of an action probability vector $p(k) = (p_1(k); \ldots; p_r(k))$. In this context, the notation $p_i(k)$ signifies the probability associated with the choice of action α_i during stage k. Consider $\alpha(k) = \alpha_i$ as the selected action, which is subsequently executed within the stochastic environment. The stochastic environment produces a reinforcement signal $\beta(k) \in \{0, 1\}$, indicating its stochastic nature. The environment applies a penalty to the selected action $\alpha(k) = \alpha_i$ with a probability of c_i, resulting in the production of the reinforcement signal $\beta(k) = 1$ to indicate the occurrence of a penalty. Conversely, it provides a reward for the action with a probability of $d_i = 1 - c_i$, generating the reinforcement signal $\beta(k) = 0$ to indicate the presence of a reward. Subsequently, the automaton proceeds to revise its action probability vector by utilizing a learning algorithm, taking into account the reinforcement signal it received. It is imperative to underscore that the learning algorithm (LA) begins without prior knowledge of the set of penalty probabilities denoted by $C = \{c_1, \ldots, c_r\}$. The main objective of the LA is to discover the action with the lowest penalty probability as it interacts with the environment.

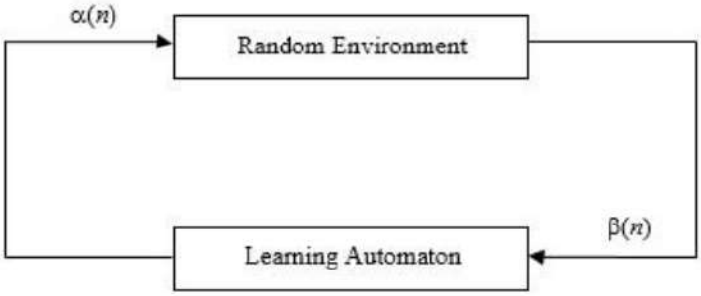


Figure 1. The relationship between the learning automaton and its random environment.

The learning algorithm utilized in this paper, known as L_{R-I}, demonstrates a dynamic capability to adjust the probability distribution associated with the selection of action α_j (where j varies from 1 to r) during stage k. This adjustment is governed by a specific rule embedded within the algorithm, as denoted by the following equation (1).

$$p_j(k+1) = \begin{cases} p_j(k) + a(k) \times [1 - p_j(k)] & if\ i = j \\ p_j(k) - a(k) \times p_j(k) & if\ i \neq j \end{cases} \quad (1)$$

where $a(k)$ is the learning rate that influences the magnitude of alterations in the probabilities associated with various actions. In scenarios where the value of β(k) is equal to 0, the value of p(k) remains unchanged. Even when β(k) equals 1, the learning rate at stage k assumes a crucial role in guiding and adapting the learning process.

Learning automata (LAs) have a wide range of applications across different domains. In computer networking, they can be utilized for load-balancing algorithms to optimize network traffic distribution [57]. In game theory, LAs provide a valuable tool for developing adaptive strategies in competitive games [58]. In machine learning, they can be employed for reinforcement learning tasks, allowing systems to learn and improve their decision-making abilities over time [59]. Furthermore, learning automata are applied in control systems[60], optimization problems[61], and even in behavioral modeling [62]. Their capacity to learn from feedback and adapt their behavior renders them a potent tool for addressing intricate problems and adjusting to dynamic environments[63]. Although an individual learning automaton (LA) inherently exhibits limitations in its capabilities, the potential for cooperation among multiple LAs becomes evident as a means to surpass these constraints. This collaboration yields a substantially expanded spectrum of capabilities that can be effectively harnessed.

4.1 Cellular Goore Game (CGG)

This section provides a brief overview of the Goore Game (GG) and its variant that incorporates Learning Automata (LA). Initially introduced by Tsetlin [24], the GG has been further analyzed by Narendra & Thathachar [25] and Thathachar & Arvind [64]. The GG is characterized by its ability to operate in a fully distributed manner, allowing participants, referred to as players, to express preferences through binary choices ("yes" or "no") that constitute their votes. A referee evaluates these votes each round, calculating a ratio 'k' that represents the proportion of "yes" votes relative to the total number of players. Based on this ratio, the referee allocates a dollar with a

probability determined by the function $G(k)$, while each player receives a dollar independently with a probability of $1 - G(k)$.

The modeling of local interactions among Learning Automata often employs familiar data structures, such as graphs, trees, or meshes. A notable advancement in this domain is the integration of Learning Automata with cellular automata, resulting in the Cellular Goore Game (CGG) [26]. This novel model enhances learning capabilities beyond those of both the traditional GG and cellular automata. The CGG operates as a network of interconnected Goore Games, where a subset of nodes serves as referees, facilitating localized interactions with neighboring players. Players in the CGG independently select actions based on the rewards and penalties communicated by their local referees, without awareness of other players' actions or the factors influencing outcomes.

In the CGG, each player utilizes a learning automaton with two actions to make decisions, while referees provide rewards based on performance criteria. The action probability vector of each player's learning automaton defines their internal state, which is subsequently updated based on reinforcement signals from referees. This iterative process continues until the desired performance is achieved. Formally, a Cellular Goore Game (CGG) with Learning Automata is defined as a 5-tuple $CGG = (N, P, R, A, G)$, where $N = (V, E)$ represents the undirected network structure, P is the subset of vertices acting as players, R is the subset acting as referees, A is the set of learning automata, and G denotes the unimodal performance criteria for the referees. The primary objective of the CGG is to maximize the overall performance criterion, represented by the sum of G_i values, where G_i corresponds to the performance criterion for referee i.

5 Stochastic Multi-layer Graph and Shortest Path in Stochastic Multilayer Graph

A stochastic multi-layer graph $\mathcal{M}$ can be described by a triple $\mathcal{M} = (\mathcal{G}, \mathcal{C}, \mathrm{W})$ where $\mathcal{G} = \{G_q : q \in \{1, \dots, M\}\}$ is a family of graphs $G_q = (V_q, E_q, W_q)$ called layers of $\mathcal{M}$ where $E_q \subseteq V_q \times V_q$ is an interlayer edge and $W_q \in W$ is a matrix in which w_{ij} is a random variable connected with edge $e_{ij} \in E_q$ if such an edge exists. Let $\mathcal{C} = \{E_{qk} \subseteq V_q \times V_k;\ q, k \in \{1, \dots, M\}, q \neq k\}$ is the set of interconnections among nodes of different layers G_q and G_k with $q \neq k$ and $W_{\mathcal{C}} \in W$ is a matrix in which $w_{ij} \in W_{\mathcal{C}}$ is a random variable connected with edge $e_{ij} \in \mathcal{C}$ if such an edge exists.

In the context of the multi-layer graph, the shortest path in a stochastic multilayer graph refers to a path that minimizes the expected total cost or distance between two nodes in the graph, taking into account both the edges connecting the nodes and the uncertainty associated with traversing those edges. The goal of finding the shortest path in a stochastic multilayer graph is to determine a path that, on average, has the lowest expected cost or distance between the desired nodes, considering the stochastic nature of the graph. This means that the path may not always have the minimum possible cost in any given instance, but it aims to minimize the average expected value over multiple instances.

Definition 2: The Stochastic Multi-Layer Shortest Path Problem (**SMSP**) is formally articulated as the endeavor to identify an optimal path between designated vertices v_s^i and v_d^j. This involves navigating from the source node vs in layer i to the destination node v_d in layer j, where i and j may exhibit either equality or disparity. Within this definition, the shortest path may manifest either within the same layer, traversing from the source node to destination nodes, or across different layers of the input networks. Furthermore, the primary objective of the SMSP is to minimize the total expected value associated with the identified path.

Definition 3: The Stochastic Multi-Layer Shortest Path and All-Destination problem (**SMSS&AD**) within a stochastic multi-layer graph aims to identify the shortest paths originating from a source node v_s^i to a set of destination nodes $\forall\, v_d^j \subseteq V_q$, where $\forall\, V_q \in \mathcal{g} - v_s^i$ with the minimum expected value. This definition specifically implies to determining the computation of the shortest path from a source node to a subset of destination nodes within a stochastic multi-layer graph.

Definition 4: The concept of Stochastic Multi-layer All-Pairs Shortest Paths (SMAPSP) in a stochastic multi-layer graph refers to the process of determining the shortest paths for all pairs of nodes $\forall v_s^i, v_d^j \subseteq V_q$, where $\forall\, V_q \in \mathcal{g}$ and $v_s^i \neq v_d^j$ with the minimum expected value. In SMAPSP, the objective is to minimize the total expected value of all pairwise paths.

Definition 5: The concept of the Stochastic Multilayer Constrained Shortest Path (**SMCSP**) in a stochastic multi-layer graph aims to identify the shortest paths originating from a source node v_s^i to destination node v_d^j, where, $v_s^i, v_d^j \in V_q$, $V_q \in \mathcal{g}$ and $v_s^i \neq v_d^j$ with the minimum expected value. Let C be the set of constraints, such that constraints may include mandates to avoid certain nodes or edges. In this definition, in addition to finding the shortest path from a source node v_s^i to destination node v_d^j, the objective is to minimize or maximize the expected value of the constraints encountered along the path, or the task of finding the shortest path within a given time limit.

Definition 6: Interlayer shortest path in stochastic multilayer graph **(ISPS)** refers to the shortest path from a source node v_s^i to destination node v_d^j where, $v_s^i, v_d^j \in V_q$, $\{i,j\} \in M$ and $i \neq j$. In this definition, the interlayer shortest path problem in a stochastic multilayer graph involves finding the most efficient path between nodes situated in distinct layers, considering the layer-wise connectivity, and minimizing the expected value of traversing edges between layers.

Remark 1: Our proposed algorithm is formulated based on the definitions outlined in Definition 2, which elucidates that the shortest path may be formed either within the same layer, traversing from the source node to destination nodes, or across different layers of the input networks.

6 Proposed Algorithm

In this section, we introduce our proposed algorithm based on the cellular goore game algorithm. The goal is to efficiently determine the shortest path, i.e., the path with the minimum expected weight, in stochastic multi-layer graphs. In the proposed algorithm, the stochastic multi-layer graph serves as the random environment for the cellular Goore game. The response generated by the environment is derived based on the sampled length of this specific path. By considering the sampled path length, the environment effectively determines the appropriate action or output in response to the selected action. The response generated by the environment, whether favorable or unfavorable, results in the rewarding or penalizing of the actions taken along the traversed path. The evaluation of these actions establishes a direct correlation between the response outcome and the consequences related to the selected action. The CGG-based algorithm consists of four phases, including: 1. Initialization, 2. Shortest Path Computation, 3. Updating the action probability vector, and 4. Stopping conditions, as described below.

6.1 Initialization

Let $\mathcal{M} = (\mathcal{G}, \mathcal{C}, W)$ represent a stochastic multi-layer graph, where $\mathcal{G} = \{G_q : q \in \{1, \dots, M\}\}$ is a graph $G_q = (V_q, E_q, W_q)$ called a layer of $\mathcal{M}$. Moreover, $E_q \subseteq V_q \times V_q$ represents interlayer edges and $W_q \in W$ is a weighted matrix. The weight w_{ij}^q is applied as a random variable to the edge $e_{ij} \in E_q$ in layer q if such an edge exists. Moreover, $\mathcal{C} = \{E_{qk} \subseteq V_q \times V_k;\ q, k \in \{1, \dots, M\}, q \neq k\}$ denotes the set of edges connecting nodes of different layers G_q and G_k $(q \neq k)$. $W_{\mathcal{C}} \in W$ is a matrix in which $w_{i,j}^{qk} \in W_{\mathcal{C}}$ is a random variable associated with edge $e_{i,j}^{qk} \in \mathcal{C}$ if such an edge exists. In the CGG-based algorithm, each node i, referred to as a cell, is equipped with a learning automaton with two available actions α_{i1}^q and α_{i2}^q, corresponding to "Yes" and "No" corresponding to α_{i1}^q ("Take a sample") and α_{i2}^q("Do not take a sample") in the context of the Goore Game (GG). Furthermore, each cell contains a referee function denoted as R_i^q. The learning automaton LA_i^q in cell i with actions α_{i1}^q ("Yes") and α_{i2}^q ("No") is described by the action probability vector, $P_i^q(k) = (p_{i1}^q(k), p_{i2}^q(k))$ with initial values $p_{i1}^q(k) = p_{i2}^q(k) = 0.5$. Let v_s^l and $v_a^{l'}$ be the source node and destination node in layer l and l', respectively. It is essential to note that the terms "player" and "automation" are used interchangeably.

6.2 Candidate Shortest Path Computation

In this stage, every cell resided by a learning automaton plays the Goore Game simultaneously and independently with its neighboring cells (as players) to identify a candidate shortest path in the stochastic multi-layer graph. During each iteration, each learning automaton in a cell selects one action based on its action probability vector. Let α_{i1}^q be the selected action perform by automaton LA_i^q in layer q, corresponding to a member of candidate multi-layer shortest path MSSP. Subsequently, the selected action by the learning automatons is checked whether it forms the shortest path v_s^l to $v_d^{l'}$ (from source node to destination node in layer l and l'). If the action selection results in the construction of a candidate shortest path, the weight and length of the path are computed simultaneously. If the action selected by learning automatons does not form a candidate shortest path, then the process of action selection persistently continues until a candidate shortest path is achieved. Let's suppose that path π_i (for $i = 1,2, \dots, r$) is selected k_i times, where $k = \sum_{i=1}^{r} k_i$. Let $L_{\pi_i}(j)$ and $\bar{L}_{\pi_i}(j)$ be the length of the path π_i (for $i = 1, 2, \dots, r$) at j^{th} sampling time (for $j = 1, 2, \dots, k_i$) and the mean of the path π_i $(i = 1, 2, \dots, r)$ respectively. The threshold computation at instance $k > 1$ is defined as:

$$\psi_k = \frac{1}{n} \sum_{i=1}^{r} \tau_i(k_i) \qquad (2)$$

Where k_i is the number of times that path π_i is traversed and

$$\tau_i(k_i) = \frac{1}{k_i} \sum_{j=1}^{k_i} L_{\pi_i}(j) \qquad (3)$$

Also, $\bar{\pi}_{L_{\pi_i}}$ refers to the average cost of the traversed path for each cell and determined by equation (4):

$$\bar{\pi}_{L_{\pi_i}} = \frac{\psi_k}{L_{\pi_i}(k)} \tag{4}$$

Subsequently, each referee associated with the candidate shortest path (SSP), utilizes $\bar{\pi}_{L_{\pi_i}}$ to evaluate the ratio of "Yes" responses from its neighboring players to the total number of LAs participants. This computation empowers the referee to ascertain the trajectory of the stochastic multi-layer network and choose a sample from the edge connected to the referee. Let $\lambda_i^q(t)$ represent the number of neighboring players to cell_i in layer q selecting their first actions denoted as α_{i1}^q. This function counts and computes the action α_{i1}^q by the neighboring players at round k and is defined by equation (5):

$$\lambda_i^q(k) = \sum_{i=1}^{d_i^q} I\{q_i(k) = \alpha_{i1}^q\} \tag{5}$$

Where d_i^q indicates as the degree of node i, and I represents the indicator function. It is noteworthy that, due to the connection between each cell in the input graph, denoted as cell_i, and the corresponding node v_i^q, in certain instances, the node v_i in layer q may be interchangeably referred to as $cell_i^q$ and vice versa. Let $\theta_i^q(k)$ is the fraction of edges that selects its first action over the degree of cell_i in layer q at round k as computed by the following equation (5):

$$\theta_i^q(k) = \frac{\lambda_i^q(k)}{|\mathrm{d}(v_i^q)|+1}\ , \tag{5}$$

where $\mathrm{d}(v_i^q)$ indicates the degree of node v_i in layer q.

$$\mathrm{G_i^q}\left(\theta_\mathrm{i}^\mathrm{q}(\mathrm{k})\right) = 0.9\mathrm{e}^{-((\theta_\mathrm{i}^\mathrm{q}(\mathrm{k})-\bar{\pi}_{L_{\pi_i}})(\theta_\mathrm{i}^\mathrm{q}(\mathrm{k})-\bar{\pi}_{L_{\pi_i}})/0.0625)} \tag{6}$$

After the computation of $G_i^q\left(\theta_i^q(k)\right)$ by the referee residing in $cell_i$ in layer q, each referee generates a reinforcement signal. The reinforcement signal, denoted as $\beta_i^q(t)$, is defined as the signal produced by the referee i in layer q and is computed using equation (7):

$$E\left[\beta_i^q(k)\middle|\theta_\mathrm{i}^\mathrm{q}(\mathrm{k})\right] = \mathrm{G_i^q}(0.9\mathrm{e}^{-\left(\frac{\left(\theta_\mathrm{i}^\mathrm{q}(\mathrm{k})-\bar{\pi}_{L_{\pi_i}}\right)\left(\theta_\mathrm{i}^\mathrm{q}(\mathrm{k})-\bar{\pi}_{L_{\pi_i}}\right)}{0.0625}\right)}) \tag{7}$$

Subsequently, each player adjacent to referee i updates their selected action.

6.3 Update action probabilities

In this stage, each referee, denoted as G_i^q, computes the difference value of the Kolmogorov–Smirnov test (KS-test) using the equation (27):

$$KS\left(G_i^q(\mathrm{k}), G_i^q(\mathrm{k}-1)\right) = \max_k |G_i^q(\mathrm{k}) - G_i^q(\mathrm{k}-1)| \quad (8)$$

where $G_i^q(\mathrm{k})$ is the value of referee i in layer q at iteration k. To update the selected action of the players, each player adjacent to referee i in layer q examines whether the value of equation (8) is less than ε_i^q. If this condition holds true, each adjacent LA generates a random number independently. If the selected action by LA_i is equal to α_{i1}^q and the random number is less than or equal to the value of the performance function KS, then the probability of the action, denoted as $p_{i1}(t)$ for each adjacent $LA_i$is updated according to equation (9):

$$\begin{aligned} p_{i1}(k+1) &= p_{i1}(k) + \lambda\left(1 - p_{i1}(k)\right) \\ p_{i2}(k+1) &= p_{i2}(k) - \lambda p_{i2}(k) \end{aligned} \quad (9)$$

where λ is determined as the learning rate. Conversely, if the selected action by each player is equal to α_{i2}^q and the random number is less than the value of the performance function KS, then the probability of action $p_{i2}(k)$ for each adjacent $LA_i$is updated according to equation (10):

$$\begin{aligned} p_{i2}(k+1) &= p_{i2}(k) + \lambda\left(1 - p_{i2}(k)\right) \\ p_{i1}(k+1) &= p_{i1}(k) - \lambda\, p_{i1}(k) \end{aligned} \quad (10)$$

Otherwise, if the random number is greater than the value of either $p_{i1}(t)$ or $p_{i2}(t)$ the action probability vector remains unchanged.

6.4 Stopping Conditions

For all the learning automatons, two criteria may be employed to stop the CGG-based algorithm for the shortest path computation. The algorithm will cease under either of two conditions: either the number of iterations, denoted as k, exceeds a pre-defined threshold K_{max}, or the value of $PC(k)$ reaches a predetermined limit designated as P_{max}.

$$PC(k) = \prod_{\substack{v_i^l \in L_{\pi_k} \\ l \in \{1,\ldots,M\}}} (\max(p(v_i^l)) \quad (11)$$

where $PC(k)$ represents the outcome obtained by multiplying the highest probabilities found within the probability vectors of learning automata associated with the vertices of the candidate shortest path in a stochastic multilayer graph during iteration k. Moreover $p(v_i^l)$ is the action probability vector associated with learning automaton A_i in layer l, which is resided in vertex v_i^l. The pseudo-code of the proposed algorithm is given in Figure 2.

Algorithm 1: The proposed Goore-based algorithm for finding the shortest path in stochastic multilayer graph
Inputs: The stochastic multilayer graph $\mathcal{M} = (\mathcal{G}, \mathcal{C}, \mathrm{W})$, Threshold $\mathrm{K_{max}}$, $\mathrm{P_{max}}$, v_s^l source node, destination nodes $v_d^{l'}$
Output: Computed Stochastic Multilayer shortest path
Initialization
Create CGG isomorphic to the graph $\mathcal{M}$ by associating with a cell in each node and then assigning a Player/LA in each cell.
Let P is the set of player and P_i^q is the player resided in cell i in layer q.
Let R is the set of referees and R_i^q is the referee resided in cell i in layer q.
Let G is the set of uni-modal performance function and G_i^q is the performance that is assigned to cell i in layer q.
Let $\alpha_i(k) = \{\alpha_1^i(k). \ldots . \alpha_n^i(k)\}$ denotes the action set in which $\alpha_j^i(k)$ consist of two actions $\alpha_{j.1}^i$and $\alpha_{j.2}^i$ for LA_j^iin cell v_j^i

Let $p_j^i(k) = (p_{j.1}^i(k), p_{j.2}^i(k))$ be the action probability vector of LA_j^i in cell v_j^i and initialized to $\frac{1}{2}$.
Let $rand(k)$ be the random number at iteration k.
Let $\pi_i(k)$ indicates as the i[th] path at iteration k.
Let $L_{\pi_i}(k)$ indicates as the length of the i[th] path at iteration k
Beginning algorithm
Let k be the iteration number of the algorithm and initially set to 1.
while k< K_{max} **or** PC < P_{max} **do**
  While (a candidate multilayer shortest path from v_s^l to $v_d^{l'}$ is not obtained **do**
    For each $cell_i^q$ in $\mathcal{G}$ **do** in parallel
      LA_i^q chooses one of its action $\alpha_i^q(k)$ from two actions $\alpha_{i.1}^q$("YES") and $\alpha_{i.2}^q$ ("NO") according to its action probability vector $p_i^q(k) = (p_{i.1}^q(k), p_{i.2}^q(k))$;
    End For
  End While
  Compute $\psi_k$and L_{π_k} at iteration k for the candidate shortest path.
  For each $Cell_i^q$ in R, **do** in parallel
    $Cell_j^i$ count the fraction of "Yes" corresponding to taking a sample from neighboring player nodes based on equation (5)
    Compute $\hat{\theta}_i^q(k)$ based on equation (6)
    Generate reinforcement signal based on equation (7)
  End For
  For each $cell_i$ in P **do** in parallel
    Let $G_i^q\left(\theta_i^q(k)\right)$ be the reward probability at round k computed by the randomly selected referee.
    If $(rnd_i(k)) \leq G_i^q\left(\theta_i^q(k)\right)$ **and** the selected action is equal to α_{i1} **then**
      Each cell which is residing in LA_i updates its action probability vector based on equation (9);
    End IF
    If $(rnd_i(k)) \leq G_i^q\left(\theta_i^q(k)\right)$ **and** the selected action is equal to α_{i2} **then**
      Each cell which is residing in LA_i updates its action probability vector based on equation (10);
    End IF
  End For
  // Calculate the product of the probabilities of LAs.
  Set $PC(k)$ = Calculate the product of all LAs in each cell at iteration k.
  Set k = k + 1;
end while
End

Figure 2: The pseudo-code for the proposed algorithm

Remark 2: In the CGG-based algorithm for finding the shortest path in a stochastic multilayer graph, if inter-layer edges are utilized to identify a potential shortest path, the cost associated with these edges is factored into the calculation of the candidate path with a minimal value (epsilon). This epsilon value is incorporated to simplify the implementation process, given its negligible magnitude. For instance, in a real-world scenario, consider a user in a Twitter social network endeavoring to propagate news to another user within the Facebook social network. In such cases, inter-layer connections are employed during the shortest path determination process from the source node to the destination node. However, due to the connectivity between these networks, the epsilon value is deemed inconsequential for the sake of implementation simplicity and ease.

7 IMPROVEMENTS

There are various ways to enhance this algorithm, and a few of these methods will be described below.

7.1 Improvement 1

According to Algorithm 1, the probability of selecting edges that demonstrate greater promise, particularly those belonging to the shortest path, gradually increases. In contrast, the probability of choosing other edges decreases. A potential approach to expedite the learning process, thereby reducing the number of required graph samples, involves applying a higher learning rate for the learning automata situated within vertices along more promising paths or those forming part of the shortest path. To implement this strategy, we can utilize equation (12) as the learning parameter, denoted as $\lambda_i(k)$, for the automaton A_i during iteration k. This adjustment in learning rate is expected to enhance the algorithm's efficiency and promote more effective convergence towards the stochastic multi-layer shortest path.

$$\lambda_i(k) = cp_{i1}^j(k) \tag{12}$$

Where c, which falls within the range of 0 to 1, represents an important factor in the equation. $p_{i1}^j(k)$ denotes the probability of selecting the first action of automaton i in layer j. As a result, algorithm 1 incorporating this adjustment in learning rate, based on improvement 1 is designated as Algorithm 2.

7.2 Improvement 2

In this strategy, the learning parameter λ_i of automaton A_i^j during iteration k is determined using the following equation:

$$\lambda_i(k) = cd_i^j(k) \tag{13}$$

Where c is a value ranging from 0 and 1, $d_i^j(k)$ represent an estimate of the reward probability for edge (v_i, v_j) computed by the algorithm at iteration k. The calculation of $d_i^j(k)$ involves dividing the number of times that action α_{i1}^j is selected up to iteration k and receives a reward by the total number of times automaton A_i^j is selected. The algorithm employing this learning rate adjustment based on equation (13) is denoted as Algorithm 3 in the context of Algorithm 1.

7.3 Improvement 3

In this strategy, the learning parameter $\lambda_i(k)$ for automaton A_i^j at iteration k is determined using the following equation (14).

$$\lambda_i(k) = c(d_i^j(k))^{(k_{max}-k)/k_{max}} \tag{14}$$

where the value of c falls within the range of 0 to 1. The term α_i^j represents the probability of choosing an action, which corresponds to selecting the edge (v_i, v_j) at iteration k. Moreover, K_{max} represents the maximum number of iterations allowed for the algorithm to continue. This strategy, shares similarities with Improvement 1 but involves a smoother increase in the learning rate. The algorithm, where the learning rate is adjusted according to improvement 3, is denoted as Algorithm 4 in the context of Algorithm 1.

7.4 Improvement 4

In all stochastic search techniques, achieving a balance between the exploration and exploitation processes is crucial. One common approach to strike this balance is by gradually decreasing the step length of the algorithm. Consequently, the algorithm emphasizes higher exploration with lower exploitation during the initial periods and shifts towards higher exploitation with lower exploration during the final periods. In strategy 4, we calculate the learning rate $\lambda_i(k)$ used by automaton A_i at iteration k using the following equation (15):

$$\lambda_i(k) = \frac{(k_{max} - k)((\lambda_{min} - \lambda_{max}) + \lambda_{max})}{k_{max}} \tag{15}$$

The search process is constrained by the maximum number of iterations, denoted as K_{max}. Additionally, predefined within the algorithm are the minimum and maximum values that the learning rate, denoted as λ_i, can take, designated as λ_{min} and λ_{max}, respectively. Employing this approach, we anticipate the proposed algorithm to explore a greater number of regions in the graph during the initial stages of the search process (exploration phase). As the algorithm progresses, it gradually shifts its focus towards regions near the solution during the later stages (exploitation phase). The dynamic adjustment of the learning rate according to Strategy 4 distinguishes Algorithm 5 as the proposed algorithm.

8 Complexity Analysis

To conduct a complexity analysis of the algorithm designed for computing the stochastic multilayer shortest path, we will systematically deconstruct the algorithm into its core components and evaluate the time complexity associated with the different operations executed within the established loops. The first step in the algorithm involves initialization, during which the algorithm sets up various sets and assigns actions and probabilities to the players (LAs) within each cell. This initialization step generally has a complexity of $O(N)$, where N represents the total number of nodes or cells in the graph. Following this, the outer while loop is initiated, which continues to execute as long as either $k < K_{max}$ or $PC < P_{max}$. Consequently, this outer loop may iterate up to K_{max} times, a constant determined by the constraints of the problem. Nested within this outer loop is an inner while loop that persists until a candidate multilayer shortest path is successfully identified. In the worst-case scenario, this inner loop could run multiple times; however, we will assume it converges within a reasonable timeframe, resulting in a total complexity generally bounded by $O(K_{max})$. For each cell i^q in the graph g, a parallel for loop allows the LAs to select their actions based on their action probability vectors. If we denote C as the number of cells, the complexity of this step is $O(C)$. Since this process is executed in parallel, it can be considered $O(1)$ in the context of the overall algorithm's time, although it is crucial to recognize that it contributes to the cumulative operations. The algorithm then computes ψ_k and $L_{(\pi_k)}$, which depends on the specifics of the algorithm's implementation; we assume this computation takes $O(1)$ for each iteration, as it merely updates values based on previously computed paths. Next, the algorithm counts the fraction of "Yes" actions for each cell i^q within the referees R, which has a complexity of $O(R)$ for each cell, leading to a cumulative complexity of $O(R \times C)$ across all iterations, assuming each cell independently counts its actions. Generating reinforcement signals

for each cell in P is similar; this operation involves parallel computations and can be approximated as $O(P \times C)$. Furthermore, the updates to the action probability vectors, based on random numbers and selected actions, are conducted in each cell, resulting in a complexity of $O(C)$ for each iteration. The calculation of $PC(k)$, which computes the product of all LAs in each cell, also holds a complexity of $O(C)$ for each iteration. When we consolidate these analyses, we find that the outer loop may run a maximum of $O(K_{max})$ times, and within each iteration of the outer loop, the inner while loop executes until a path is identified. The operations within the parallel loops contribute $O(C)$ for the actions of LAs, reinforcement signal generation, and updates. Therefore, the overall time complexity of the algorithm can be succinctly represented as $O(Kmax \times (C + R + P))$, where K_{max} denotes the maximum number of iterations permitted, C is the number of cells (or nodes), R is the number of referees, and P is the number of players (LAs). In conclusion, this complexity analysis approximates the algorithm's complexity as $O(Kmax \times (C + R + P))$, contingent upon the reasonable convergence of the inner while loop and the effective parallelization of operations within each iteration.

Remark 3: The proof of the algorithm converges to the optimal solution through the selection of an appropriate learning rate is detailed in Appendix I. Furthermore, it is demonstrated that by appropriately choosing this learning rate, the probability of reaching the stochastic multi-layer shortest path (SMSP) with the minimum expected weight exceeding 1-ε is established. The complete proof is provided in Appendix I for further reference.

9 EXPERIMENTS

To evaluate the performance and effectiveness of the proposed algorithms, a series of experiments were conducted on various stochastic multilayer graphs. The primary aim of these experiments was to evaluate the algorithms' performance across a range of criteria, thereby determining the efficacy of the Cellular Goore Game (CGG)-based algorithm in real-world applications. Two distinct types of stochastic multilayer structured graphs were utilized for the experiments: graphs with a predetermined structure and random graphs. Table 1 outlines the characteristics of the predefined multilayer structure.

Table 1: The stochastic multi-layer graphs with predefined structure used for experiments

Test Graph	#Nodes	# Edges	# Layers
Graph 1	5	5	2
Graph 2	10	18	2
Graph 3	15	42	2

The second category comprised random graphs, with node counts varying from 100 to 500 with step size 100; all generated random graphs consisted of two layers. In the stochastic multilayer graphs characterized by a predefined structure, it is assumed that nodes are distributed across different layers, each assigned unique index numbers. Additionally, it is assumed that a snapshot of the stochastic multilayer graph exists at a specific moment, serving as the basis for algorithm operation. Within this framework, the structure of the generated stochastic graph is regarded as static, indicating that the relationships and interactions among nodes do not change over time during the execution of the algorithms. For the random graphs, edges are established between pairs of nodes based on a defined low probability, with the length of each edge determined by normally distributed random variables. The mean value of each edge length is uniformly distributed within

the interval [6, 200], while the variance of the normal distribution is set to 3. Finally, a source node and a destination node are randomly selected from the set of nodes within the stochastic multilayer graph, ensuring that these nodes are drawn from different layers.

Table 2: The real multi-layer graphs used for experiments

Test Graph	#Nodes	# Edges	# Layers
CS-AARHUS	61	620	5
C.ELEGANS	279	5863	3
CKM PHYSICIANS INNOVATION	246	1551	3

A third type of network used in our experiments involves real multilayer social networks, offering rich and diverse structures across multiple dimensions. These datasets include CS-AARHUS, a multilayer social network of 61 nodes, 620 edges, and 5 distinct layers, representing interactions among computer science researchers at Aarhus University. Another dataset, C. ELEGANS, consists of 279 nodes, 5863 edges, and 3 layers, capturing the neural, muscle, and chemical interaction networks of the C. ELEGANS organism. Finally, CKM Physicians Innovation features 246 nodes, 1551 edges, and 3 layers, illustrating the collaborative and professional relationships among physicians in an innovation network. These multilayer networks serve as an excellent foundation for analyzing complex relationships and interactions across multiple layers. Table 2 gives the characteristics of the real networks used for experiments.

All presented simulations were carried out using the L_{R-I} updating the learning scheme for the probability vector of Learning Automata (LAs). Each algorithm was terminated either when the product of probabilities exceeded 0.9 or the number of iterations reached $N \times 100$. The algorithm is implemented in the C# platform, and it runs on a system with an Intel Core i5 processor and 4 gigabytes of memory. Furthermore, to assess the performance of each algorithm, two distinct types of multi-layer structured graphs were employed, namely, graphs with predetermined structures and random graphs. It should be noted that every value in each table was obtained by averaging the results of 100 independent runs of the algorithm.

To evaluate the efficacy of the proposed CGG-based algorithm for shortest path determination, we conducted a comparative analysis against two established learning automata algorithms: the extended distributed learning automata (e-DLA) [61] and the adaptive Petri net[68]. The Extended Distributed Learning Automata (eDLA) algorithm operates by facilitating cooperation among learning automata (LAs) within a network, where each LA possesses an activity level that dynamically adjusts over time in response to the underlying problem and established communication rules. At each iteration of the algorithm, one LA is designated as "Fire," indicating it has the highest activity level among its peers. This selective activation serves to streamline the decision-making process as the eDLA algorithm introduces a new adaptive procedure specifically designed to tackle optimization problems in stochastic graphs. By implementing this method, the algorithm effectively identifies a set of edges to be sampled, thereby allowing for the specification of a subgraph that yields the optimal expected weight. This operational framework enhances the algorithm's efficiency in solving complex optimization tasks, making it a valuable tool in various applications involving stochastic environments. In the context of the Adaptive Petri Net with Learning Automata (APN-LA), they proposed an adaptive Petri net system based on Irregular Cellular Learning Automata (ICLA) that facilitates cooperation among multiple learning automata. This system, referred to as APN-ICLA, is structured into two layers: the PN-layer and the ICLA-layer. The PN-layer is designed as a Petri net in which conflicting transitions are divided

into several clusters, each managed by a designated controller to address potential conflicts among the transitions within that cluster. The ICLA-layer serves to provide the necessary controllers for the PN-layer and consists of an ICLA structure in which each cell corresponds to a cluster in the PN-layer, with the learning automaton within each cell acting as the controller for its respective cluster.

9.1 Performance Measures

To assess the performance of the proposed algorithms and compare various enhancement strategies, we utilized four distinct performance measures. These measures are defined as follows:
Sampling Ratio: The Sampling Ratio (SR) serves as a metric quantifying the proportion of samples acquired in a data sampling process. It is calculated by dividing the "samples per iteration" by the average number of samples obtained from the network during the sampling process. Here, "samples per iteration" denotes the number of samples taken in each iteration of the sampling process. Conversely, "average samples from the network" represents the mean or typical number of samples collected from the network over a given period, as denoted by equation (16):

$$\textbf{\textit{Sampling Ratio}} = \frac{\#\text{ samples per iteration}}{\#\text{ Average samples from the network}} \tag{16}$$

The sampling ratio offers insights into the efficiency and comprehensiveness of the sampling procedure. A higher sampling ratio indicates that a larger number of samples are collected per iteration relative to the average number of samples from the network, signifying a more thorough exploration of the data. Conversely, a lower sampling ratio suggests that fewer samples are collected per iteration compared to the average, potentially resulting in a less representative sample or reduced coverage of the network data.

Shortest Path Ratio: The Shortest Path Ratio **(SPR)** is a mathematical metric calculated by dividing the number of samples collected from the shortest path by the total number of samples taken, as denoted by the equation (17):

$$\textbf{\textit{Shortest Path Ratio}} = \frac{\#\text{ samples from the shortest path}}{\#\text{ total number of samples}} \tag{17}$$

The Shortest Path Ratio (SPR) serves as a quantitative measure of the emphasis on the shortest path samples during the sampling process. By dividing the number of samples taken from the shortest path by the total number of samples, this ratio offers insight into the concentration of sampling efforts on the most efficient or critical path within the network.

Average number of iterations: The average number of iterations (N_1) performance measure indicates the mean number of iterations required for the algorithm to successfully identify the multi-layer shortest path. In cases where the algorithm fails to identify the shortest path, the number of iterations it takes to terminate is utilized to determine the average number of iterations. A lower value for (N_1) is indicative of superior performance.

Proportion of Convergence: This performance metric, known as the proportion of converged runs (M_1), evaluates the ratio of successful runs that have converged. Given the stochastic nature of both the input multi-layer graphs and the proposed algorithm, it becomes essential to execute

the algorithm multiple times and analyze its behavior across these runs. This performance measure reflects the percentage of runs that accurately identifies the shortest path. A higher (M_1) value indicates superior performance.

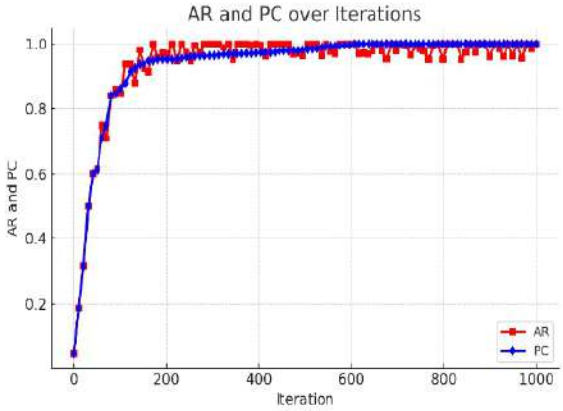
9.2 Experimental results

9.2.1 Experiment 1

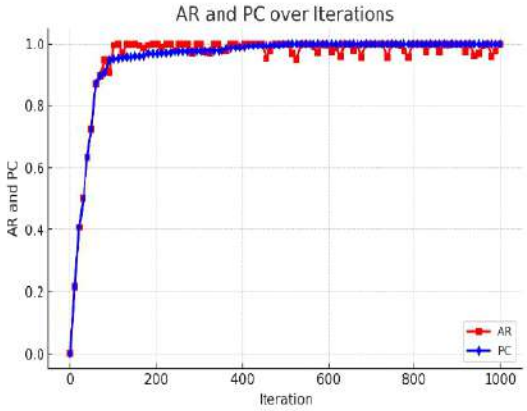
This experiment is designed to scrutinize the performance of Algorithm 1 in the context of searching for the shortest path in a stochastic multi-layer graph. Our analysis focuses on the behavior of the proposed algorithm concerning the metrics $PC(k)$ and average referee value $AR(k)$ relative to the iteration number k. It is noteworthy that $AR(k)$ serves as the optimal threshold, representing the average weight of all stochastic multi-layer shortest paths discovered up to iteration k (as per the definition in Equation (4)). The experimental results, depicted in Figure 3 for various test graphs, underscore the selection of Algorithm 1 due to its pertinence in seeking optimal solutions.

The convergence metric, PC, gradually approaching unity as the algorithm progresses towards a solution, is pivotal. The results illustrated in Figure 3 reveal that $AR(k)$ consistently converges to the weight of the shortest path with the minimum expected weight. Simultaneously, PC converges to unity, signifying the successful identification of the multi-layer shortest path with the minimum expected weight by Algorithm 1. This consistent performance is evident across diverse tested stochastic multi-layer graphs, affirming the algorithm's robustness in finding optimal solutions.

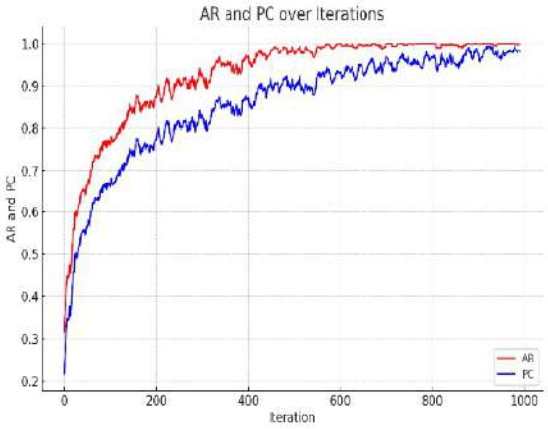
It is crucial to emphasize that the comparative outcomes observed in this experiment extend to other algorithms. Thus, the insights gained from this experiment shed light on the progressive convergence of Algorithm 1 to the weight of the shortest path with the minimum expected weight, as indicated by the converging AR and PC metrics. These findings provide empirical support for the algorithm's effectiveness in discerning multi-layer shortest paths with optimal weights. The generalizability of these results implies that analogous achievements can be realized with alternative algorithms.



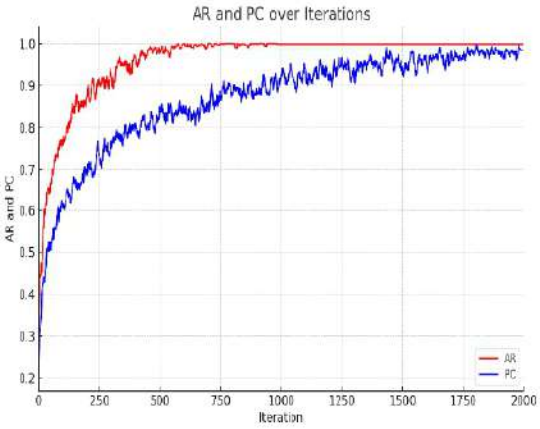
a) G1



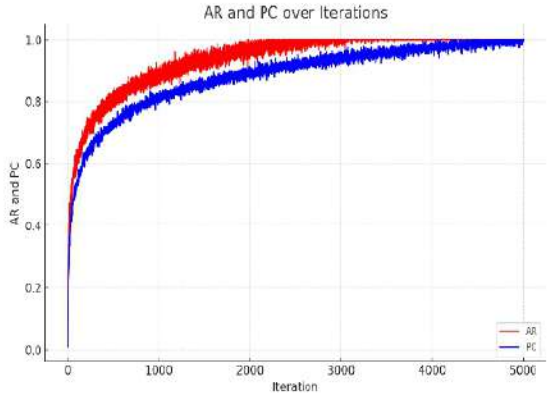
b) G2



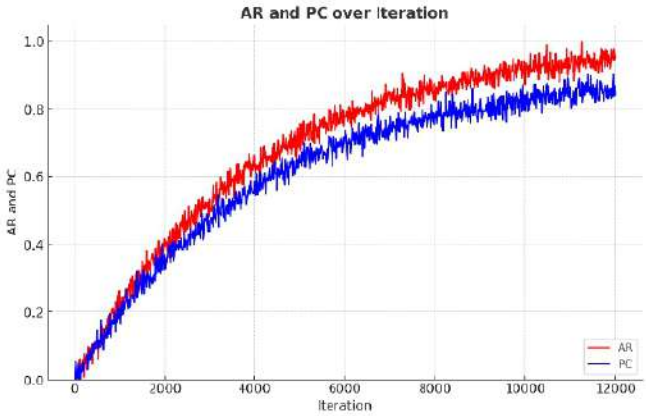
c) G3



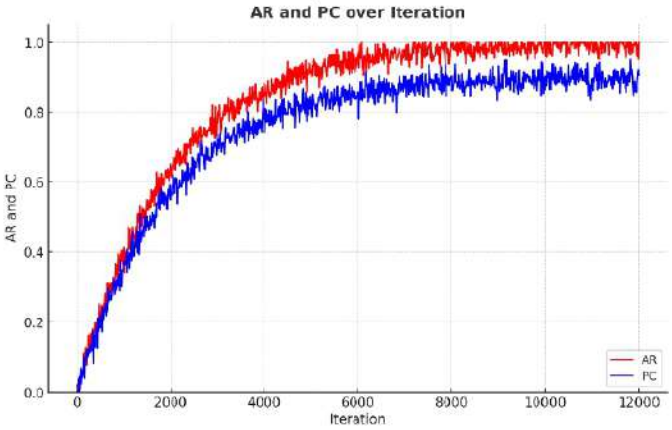
d) Rand 50



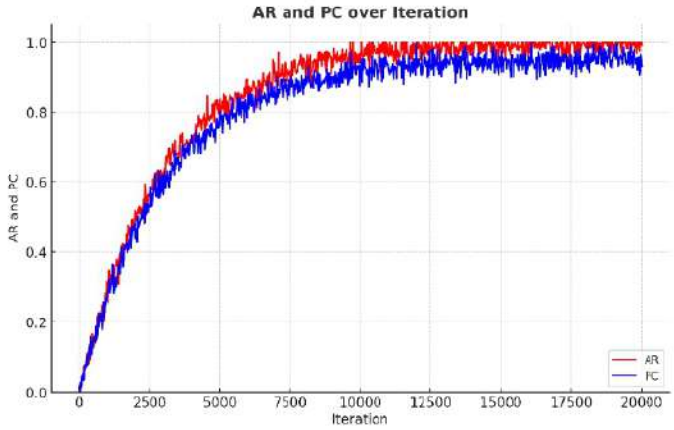
e) Rand 100



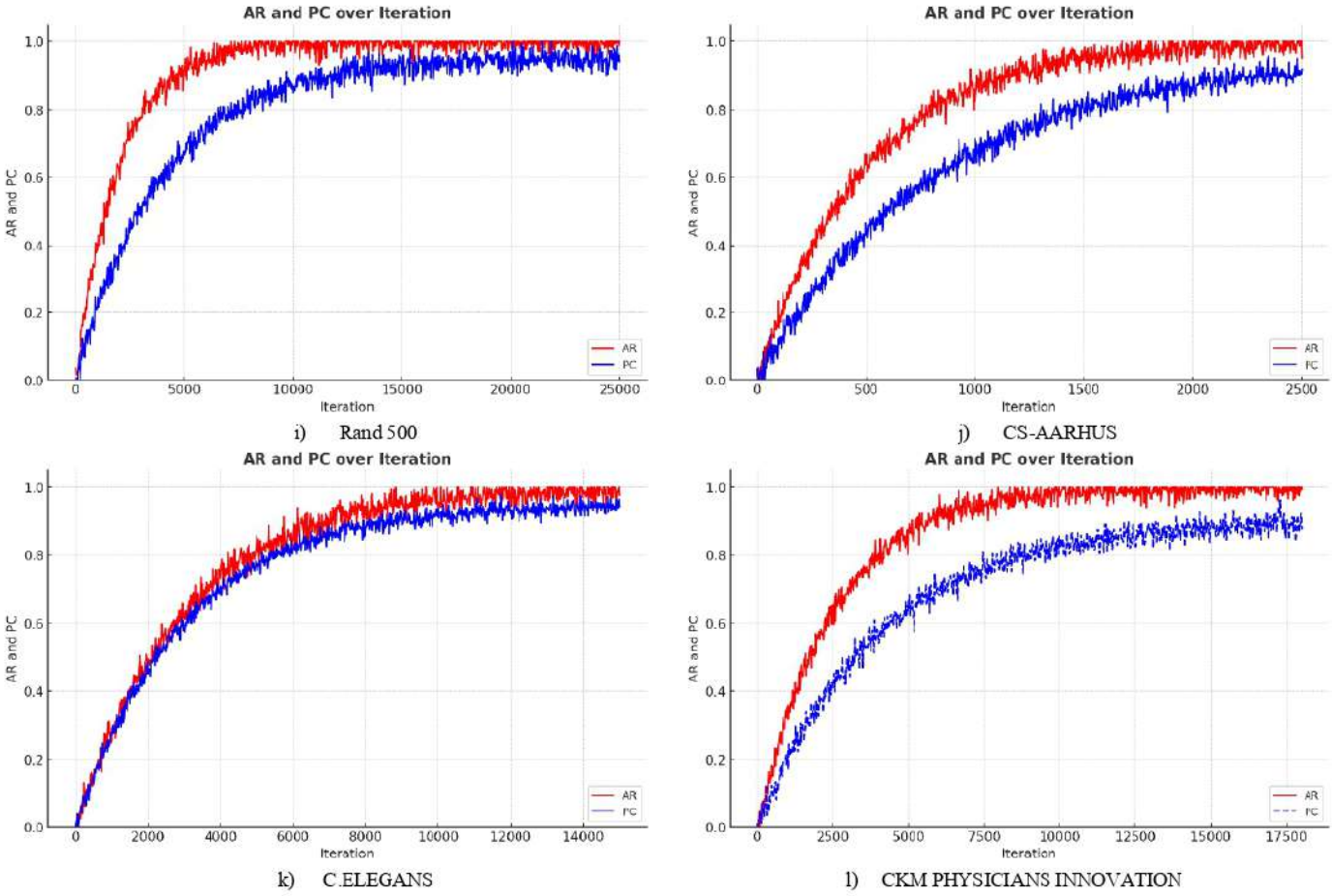
f) Rand 200



g) Rand 300



h) Rand 400



i) Rand 500 j) CS-AARHUS

k) C.ELEGANS l) CKM PHYSICIANS INNOVATION

Figure 3: The graph illustrating the relationship between PC and AR over the course of multiple iterations for various graphs.

9.2.2 Experiment 2

In this experiment, our objective was to investigate the relationship between the learning rate and the number of samples required by different algorithms to achieve efficient learning. We compared the performance of diverse strategies applied to Algorithm 1, examining how the learning rate influenced the average number of samples extracted from the stochastic multilayer graph and the determination of the optimal path. We selected various algorithms recognized for their learning rate capabilities, spanning from Algorithm 1 to Algorithm 5. The study involved the manipulation of the learning rate from 0.001 to 0.050, with step increments of 0.005.

Upon employing a small learning rate, we observed that all algorithms necessitated a larger number of samples to attain satisfactory outcomes for identifying the shortest path. This implies that algorithms with a small learning rate require more samples to converge accurately. Conversely, as we increased the learning rate to a moderate level, we noted a reduction in the number of samples needed by the algorithms. This suggests that a higher learning rate facilitated faster convergence, enabling the algorithms to yield appropriate results with fewer samples.

Remarkably, across all learning rate scenarios, Algorithm 5, incorporating a guided learning approach, consistently demonstrated fewer sample requirements compared to the other algorithms. This suggests that the guided learning technique employed by Algorithm 5 contributed to its ability

to achieve accurate solutions using limited amounts of data. The experiment results are summarized in Table 4, Table 5, Table 6, Table 7, and Table 8.

Table 4: The result for average samples from the edges along the multi-layer shortest path of Graph 1

a	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS
0.001	3450	5893	2530	5074	815	1831	750	1876	721	1750
0.005	2150	4550	2247	4652	785	1756	680	1654	654	1504
0.010	1975	3950	1842	3842	652	1732	545	1457	514	1350
0.015	1720	3500	1680	3451	623	1650	520	1241	498	1115
0.020	1450	3200	1358	3104	542	1420	485	1127	474	860
0.025	1430	2789	1389	2651	511	1411	432	1021	421	650
0.030	1382	2500	1254	2335	475	1268	395	950	320	540
0.035	1250	2451	1185	2202	465	1231	384	842	298	495
0.040	1182	2341	1095	2195	452	1210	342	767	276	442
0.045	1170	2250	1067	2174	431	1189	320	721	254	431
0.050	1101	2210	1061	2142	412	1154	315	698	247	418

Table 5: The result for average samples from the edges along the multi-layer shortest path of Graph 2

a	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS
0.001	6524	13254	5665	11542	2412	4213	2264	4310	2145	4125
0.005	5351	13100	4420	11320	2314	3762	2143	3420	1950	3214
0.010	5148	12570	4124	10542	2478	3241	1995	3215	1741	3102
0.015	5002	12102	4002	10984	2154	3120	1860	2954	1321	2781
0.020	4953	11745	3985	10521	2100	3004	1741	2451	1248	2654
0.025	3841	10541	3741	9852	2004	2854	1652	2310	1102	2457
0.030	3641	10325	3800	9540	1985	2654	1345	2220	954	2320
0.035	3123	10241	3651	9321	1740	2351	1263	2142	920	2110
0.040	3012	10142	3254	8957	1564	2210	1141	1980	840	1975
0.045	2897	9995	3126	8341	1421	2104	950	1974	645	1920
0.050	2780	9914	3002	8450	1325	1940	624	1893	595	1842

Table 6: The result for average samples from the edges along the multi-layer shortest path of Graph 3

a	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS
0.001	15241	26081	14521	21021	13540	19748	9451	18751	8432	17543
0.005	15110	25740	13541	19540	12547	19540	7564	19540	7521	19540
0.010	14982	24657	12985	18954	11540	18954	6528	18954	5412	18954
0.015	14657	24142	11542	18520	10421	18520	6142	18520	4781	18520
0.020	13265	24002	10654	18103	9542	18103	5419	18103	4213	18103
0.025	13110	23987	10231	17864	9195	17864	5100	17864	3541	17864
0.030	12547	23871	10004	17547	8751	17547	4124	17547	3120	17547
0.035	12414	23421	9987	17310	7541	17310	3140	17310	2784	17310
0.040	12198	23125	8541	17002	7124	17002	1750	17002	1450	17002
0.045	11324	23025	7512	16952	6241	16952	1150	16952	956	16952
0.050	11124	23001	7214	16541	6124	16541	950	16541	741	16541

Table 7: The result for average samples from the edges along the multi-layer shortest path in a random graph with 50 nodes

a	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS
0.001	30500	58500	29870	49857	27586	47542	28542	48578	31540	48540
0.005	27500	55240	28654	48975	25402	45870	26587	46587	28654	47856
0.010	26400	53200	25430	46751	23546	43254	24251	44325	22450	35042
0.015	24200	49254	22410	45702	22543	41540	23214	42351	20145	31450
0.020	23890	47521	20541	44571	21547	39854	22354	39957	18574	28750
0.025	22450	43870	19547	42145	19854	36542	20154	37541	16468	26540
0.030	21040	42010	18540	40214	17842	33546	16521	33214	15421	25464
0.035	20451	40254	17542	38654	16524	32540	15421	31021	14254	24105
0.040	19547	38954	16947	37540	16034	31542	14870	28654	13410	23658
0.045	18750	37546	15401	36542	15980	30215	13950	25648	12658	22897
0.050	17560	36542	14021	33254	15564	29850	13240	25410	12998	21548

Table 8: The result for average samples from the edges along the multi-layer shortest path in a random graph with 100 nodes

a	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS	MSPS	TS
0.001	69800	117000	55412	105241	53241	95241	47245	89572	35214	79854
0.005	65478	112450	52314	102413	51240	92415	41255	88742	31542	75842
0.010	63241	110240	48762	100411	50235	85472	39234	84512	30254	73542
0.015	64215	104720	47541	98475	48795	82145	46744	79854	29854	72112
0.020	62314	101245	43654	95412	47568	80214	45541	75412	27541	69542
0.025	61241	998750	41221	93241	46875	78950	42831	73254	26547	65874
0.030	60500	960450	38745	91024	45002	76541	40021	71254	25412	61245
0.035	58940	935641	34572	85405	34572	75654	33242	70214	24514	59874
0.040	57846	925414	32154	82451	32154	73541	30124	58741	22145	57421
0.045	56412	935460	30254	81214	30254	72366	28641	56871	21450	45142
0.050	54214	912577	29854	79854	29854	70051	26541	52142	18795	43541

9.2.3 Experiment 3

This experiment aims to assess the performance of a predefined structure of a stochastic multi-layer graph, utilizing two key metrics: The Average Number of Iterations (AVI) and the Percentage of Converged Runs (PC) taken from the edges of the graph. The AVI metric quantifies the average number of iterations needed for the algorithm to converge to a solution. Meanwhile, the PC metric denotes the percentage of runs that successfully converged to a solution. These metrics were employed to gauge the effectiveness of the algorithm in solving problems on the predefined structure multi-layer graph, comparing its performance against randomly generated graphs and other algorithms. The experiment results are summarized in Table 9 and Table 10, respectively.

Table 9: The outcomes of various algorithms tested on graphs that have predetermined pattern or structures.

	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
	N_1	M_1	N_1	M_1	N_1	M_1	N_1	M_1	N_1	M_1

Graph 1	1601	97	1393	99	761	95	820	99	730	100
Graph 2	8235	99	3152	98	1741	98	1847	99.23	1754	100
Graph 3	8520	99.74	6199	100	2054	97	2350	99.21	2235	100

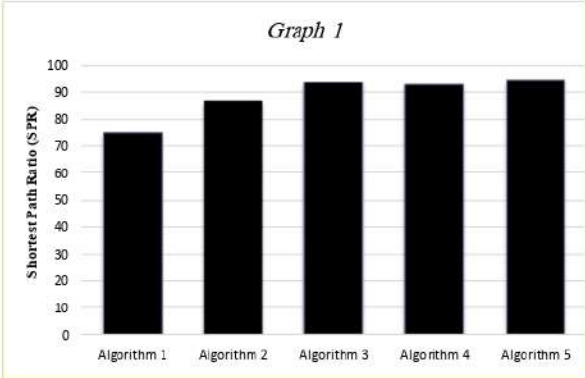
Table 10: The outcomes of various algorithms tested on random graphs that have predetermined pattern or structures.

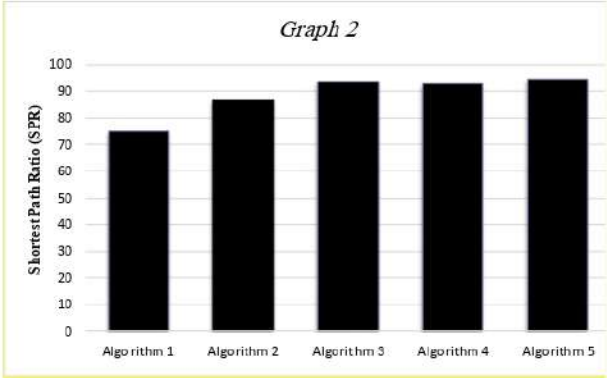
Size	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
	N_1	M_1	N_1	M_1	N_1	M_1	N_1	M_1	N_1	M_1
50	16010	97.5	14502	98.4	12410	99	9584	99.3	92412	99.2
100	32020	98.2	28541	99.1	24510	98	21452	99.2	15480	99.3
150	48030	98.4	35261	98.7	30214	99	27542	99.3	25410	99.1
200	64040	97.3	54004	98.8	47541	99	45214	99.2	33021	99.4
250	80050	98.7	75041	98.5	65840	99	61245	98.9	45124	100
300	96060	97.7	91024	97.9	75681	98	71214	99.4	54120	99.5
350	112070	98.2	100542	98.7	85472	99	81024	99.3	65241	99.4
400	128080	98.3	111502	98.8	102430	99	95472	99.2	74102	99.7
450	144090	98.4	112450	98.8	101243	98	95472	99.1	84120	100
500	160100	97.9	114210	99.1	98423	99	91423	99.7	74512	100

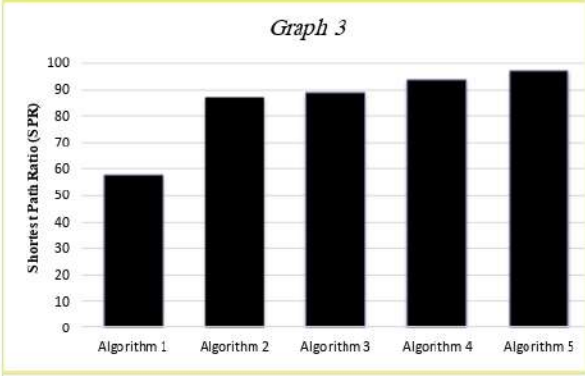
Based on the results and comparisons, it is evident that two algorithms, namely Algorithm 4 and Algorithm 5, surpass other algorithms in performance on both predefined graph structures and random graphs due to the utilization of adaptive strategies. Notably, Algorithm 4 and Algorithm 5 exhibit similarities in terms of the Average Number of Iterations (AVI) and the Percentage of Converged Runs (PC). However, Algorithm 5 outperforms Algorithm 4 when considering all metrics collectively. Specifically, Algorithm 5 demonstrates a higher AVI and PC compared to Algorithm 4. This suggests that Algorithm 5 is more efficient and effective in solving problems on the predefined graph structure.

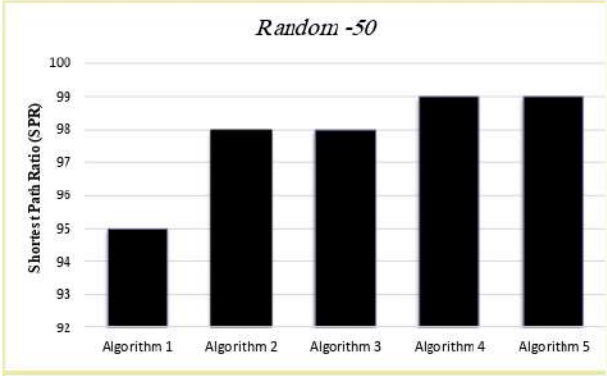
9.2.4 Experiment 4

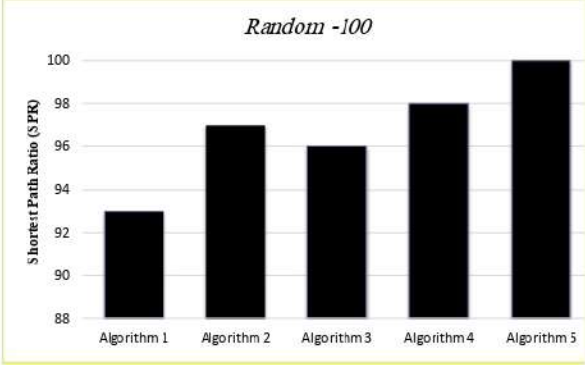
In this experiment, our primary objective is to quantify the Shortest Path Ratio (SPR), a fundamental mathematical metric used to assess the efficiency of the proposed algorithms. The Shortest Path Ratio (SPR) is defined as a mathematical ratio calculated by dividing the number of samples collected from the shortest path by the total number of samples obtained. This metric is denoted by equation (78) and serves as a reliable indicator of the algorithms' effectiveness in optimizing path selection. Through meticulous measurement and analysis of the SPR, we aim to gain valuable insights into the algorithmic performance and ultimately enhance the overall efficiency of our proposed solutions. The results are depicted in Figure 4.

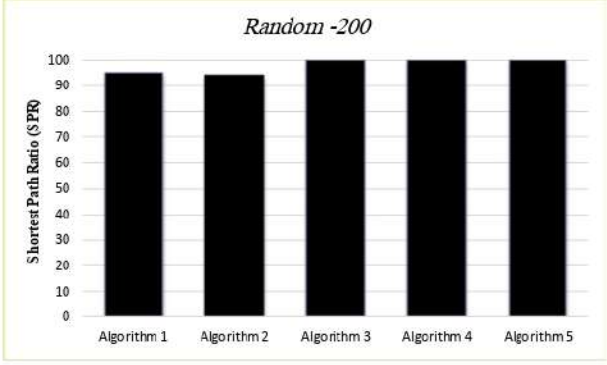

Graph 1
Shortest Path Ratio (SPR)
100
90
80
70
60
50
40
30
20
10
0
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5

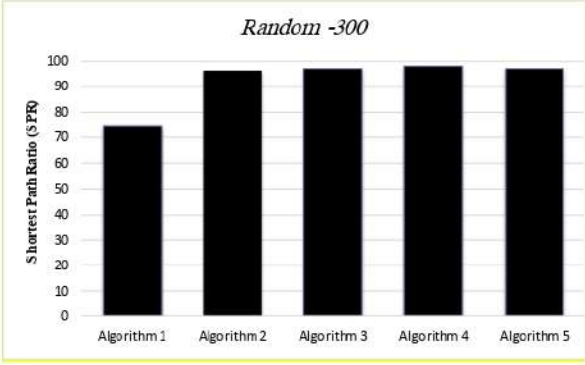

Graph 2
Shortest Path Ratio (SPR)
100
90
80
70
60
50
40
30
20
10
0
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5

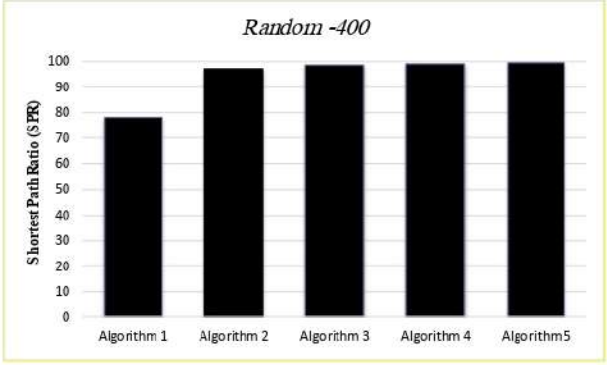

Graph 3
Shortest Path Ratio (SPR)
100
90
80
70
60
50
40
30
20
10
0
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5


Random -50
Shortest Path Ratio (SPR)
100
99
98
97
96
95
94
93
92
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5


Random -100
Shortest Path Ratio (SPR)
100
98
96
94
92
90
88
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5


Random -200
Shortest Path Ratio (SPR)
100
90
80
70
60
50
40
30
20
10
0
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5


Random -300
Shortest Path Ratio (SPR)
100
90
80
70
60
50
40
30
20
10
0
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5


Random -400
Shortest Path Ratio (SPR)
100
90
80
70
60
50
40
30
20
10
0
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm5

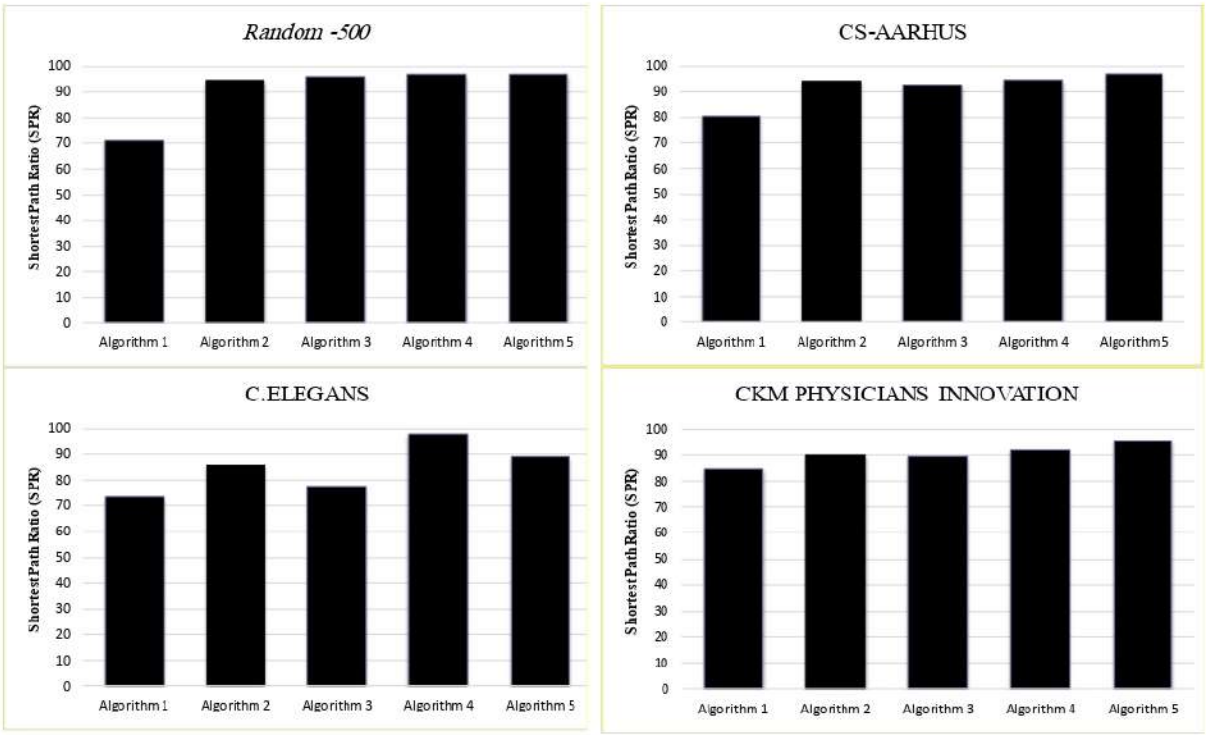


Figure 4: The comparison of different strategies in terms of the Shortest Path Ratio.

Based on the results obtained, it is evident that Algorithm 5 showcases superior performance when calculating the shortest path ratio compared to other adaptive strategies. This can be attributed to Algorithm 5's utilization of an exploration phase during the initial iteration, followed by a sufficient exploitation phase, leading to successful findings. However, it is essential to mention that the other strategies employed also yield commendable outcomes.

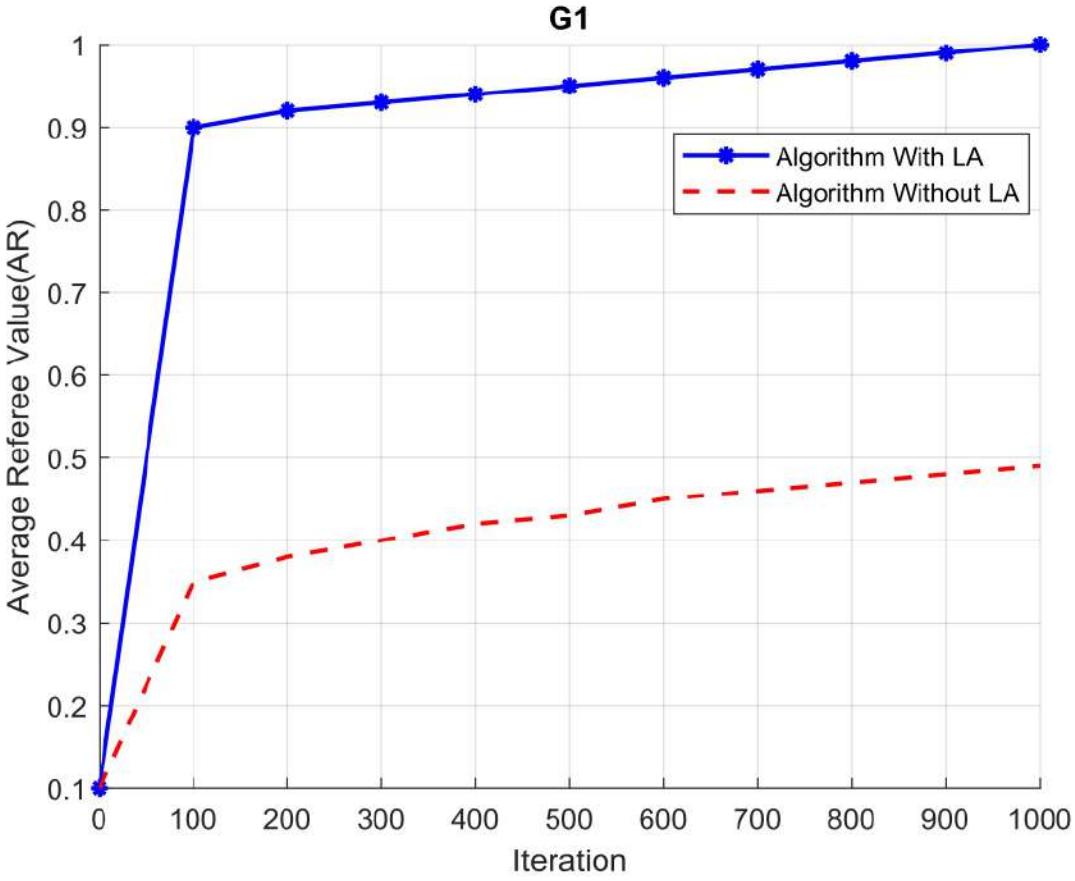
9.2.5 Experiment 5

In this experiment, we compared the performance of two algorithms: Algorithm 1 and an alternative version of Algorithm 1 in which the learning automaton residing in each vertex was replaced by a pure chance automaton in Figure 5. The purpose of the experiment was to evaluate the impact of the learning automaton on the performance of Algorithm 1.

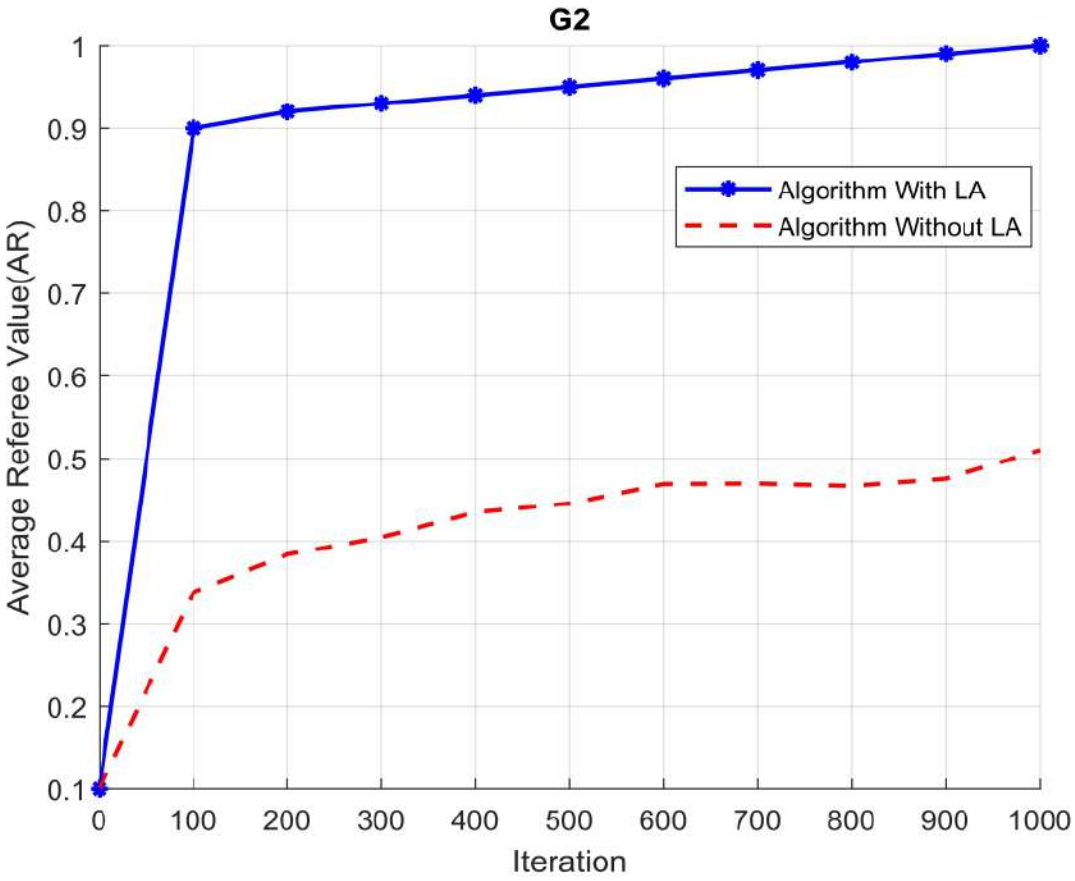
To experiment, we used a simulated network environment with a fixed number of vertices and edges and a predetermined set of rewards associated with each vertex. The algorithms were tasked with finding the optimal path through the network that maximized the total reward. The results of the experiment showed that Algorithm 1 outperformed the alternative version with pure chance automaton in terms of both convergence speed and solution quality. Algorithm 1 was able to converge to the optimal solution much faster than the pure chance automaton version, and it also found a higher-quality solution in terms of total reward.

These results suggest that the learning automaton in Algorithm 1 plays a crucial role in its ability to efficiently and effectively navigate the network and find the optimal solution. The use of a pure chance automaton, on the other hand, results in a less efficient and less effective algorithm.

Overall, this experiment highlights the importance of the learning automaton in Algorithm 1 and the impact it has on the performance of the algorithm. It also demonstrates the value of conducting experiments to evaluate the effectiveness of different algorithms and their components.



G1


G2
Average Referee Value(AR)
Iteration
Algorithm With LA
Algorithm Without LA
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500
600
700
800
900
1000
G2

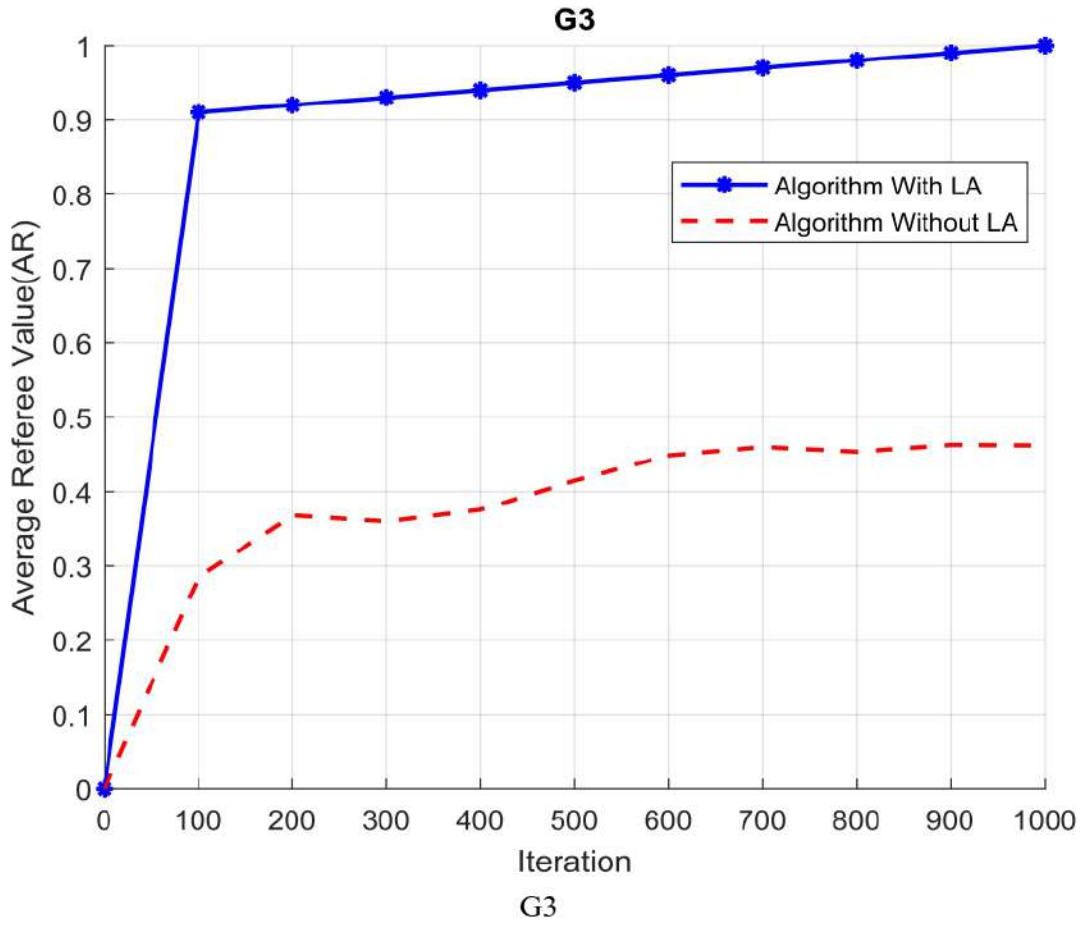
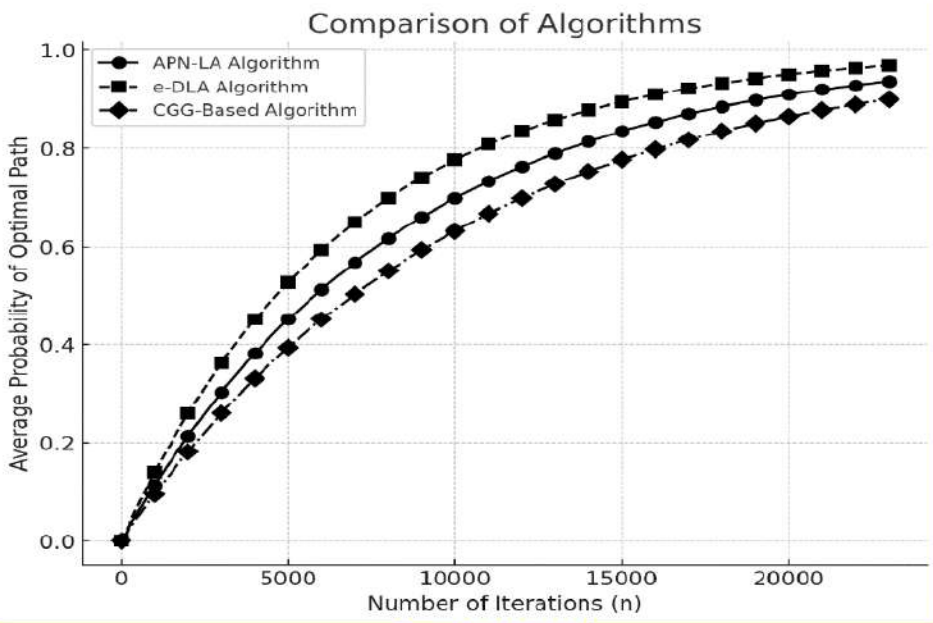


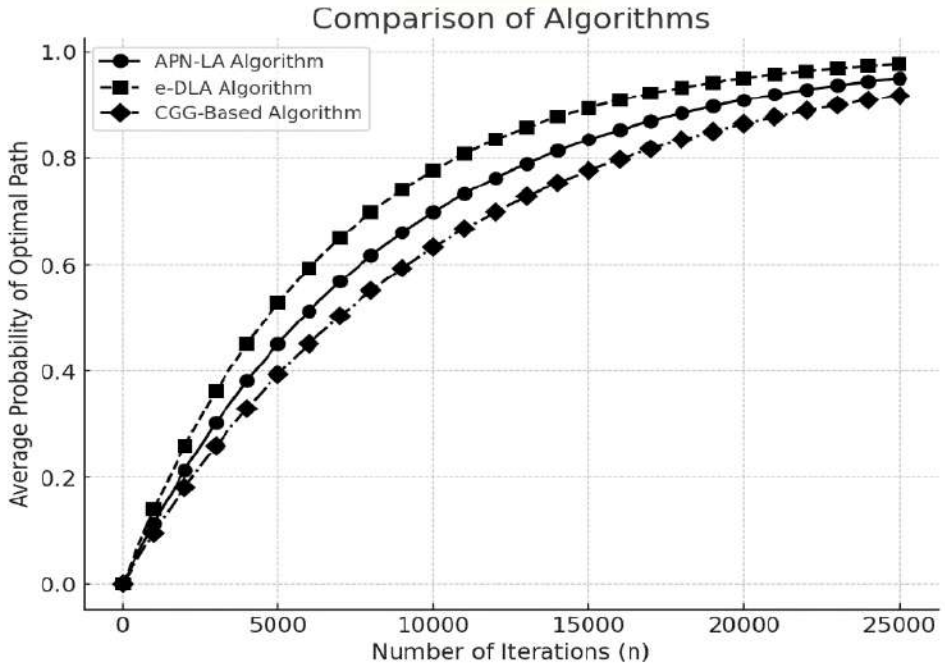
Figure 5: Algorithm 1 was compared with a modified version in which learning automata were replaced with pure chance automata to assess the effect of learning on the algorithm's performance.

9.2.6 Experiment 6

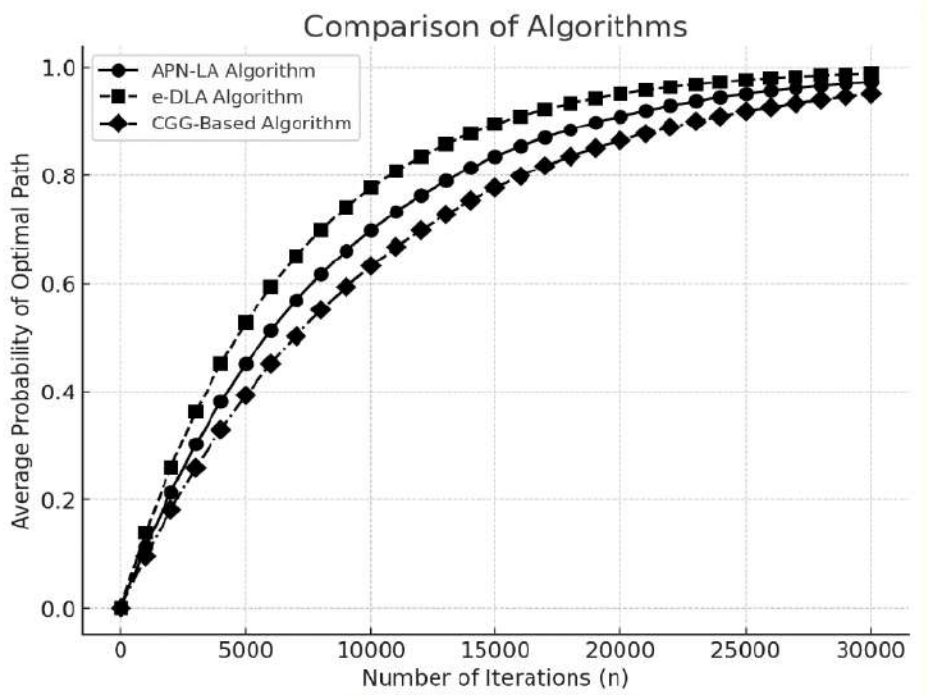
To evaluate the performance of the proposed CGG-based algorithm, a comparative analysis was conducted against two established learning automata algorithms: The Extended Distributed Learning Automata (e-DLA) and the Adaptive Petri Net with Learning Automata (APN-LA). The experiments were performed on real stochastic multi-layer graphs to assess the algorithms' efficacy in determining the shortest path. The comparison was based on the average probability of the optimal path, which measures the ratio of the sampled subgraph's weight to the optimal path weight, providing a quantitative metric for algorithm performance. By evaluating the CGG-based algorithm alongside e-DLA and APN-LA, we aim to highlight its advantages in terms of accuracy and efficiency under stochastic and multi-layered graph conditions.



a) CS-AARHUS



b) C.ELEGANS



C) CKM PHYSICIANS INNOVATION

Figure 6: Algorithm 1 was compared with other learning automata-based algorithms in terms of the average probability of selecting the optimal path.

From the results, that are shown in Figure 6, the CGG-Based Algorithm converges the fastest compared to both the e-DLA and APN-LA algorithms, as observed in the steep rise in the probability curve during the early iterations. This indicates that the CGG-based algorithm is more efficient in reaching a high probability of finding the optimal path. In contrast, the e-DLA Algorithm shows a slower but steady increase, requiring more iterations to approach the optimal probability, while the APN-LA Algorithm exhibits the slowest convergence rate, with the most gradual increase in the probability curve, particularly at the beginning. As the number of iterations increases, all three algorithms approach the same upper limit for the probability of finding the optimal path (close to 1), but the CGG-based algorithm reaches this limit significantly faster. This makes the CGG-based algorithm highly effective at quickly converging to an optimal or near-optimal solution, making it the preferred choice when rapid convergence is essential. In terms of computational efficiency, the CGG-Based Algorithm requires fewer iterations to achieve high probability, making it more efficient than e-DLA and APN-LA, which need more iterations for similar accuracy. For all real multilayer networks, this suggests that the CGG-based algorithm is more scalable and faster, particularly for applications requiring rapid path optimization, such as real-time decision-making systems. In practical applications where time and computational resources are limited, using the CGG-Based Algorithm offers significant advantages in both efficiency and effectiveness. Overall, the CGG-Based Algorithm demonstrates superior performance in terms of convergence speed and computational efficiency, making it the optimal choice for applications where time and resources are critical, especially with the real dataset.

10 Conclusion

Stochastic multi-layer graphs serve as a formidable framework for effectively capturing the intricacies and interrelationships inherent in many complex systems, which are modeled as graphs. Nonetheless, the analysis of these graphs involves formidable challenges due to their inherent complexity and the multifaceted nature of their layers. To tackle this challenge, the development of a stochastic multi-layer shortest path algorithm constitutes a significant contribution to the field of graph theory and network analysis. The algorithm is designed to find the shortest path between two nodes in a stochastic multi-layer graph, a complex and challenging task owing to the presence of multiple layers and stochastic elements. The proposed algorithm, based on Cellular Goore Game (CGG), employs a reinforcement learning approach with the assistance of learning automata to effectively determine the shortest path in a stochastic multi-layer graph. This algorithm represents a novel and unprecedented advancement, as it has not been previously employed for the specific purpose of finding the shortest path in stochastic multi-layer graphs. In the proposed algorithm, each cell contains a learning automaton and a referee, both of which collaborate harmoniously to guide the learning automata toward identifying the optimal shortest path. Within the CGG-based algorithm for shortest path determination, each learning automaton consists of two distinct actions, namely a member of the shortest path and not a member of the shortest path, the identification of a candidate shortest path from the source node to the destination node. Subsequently, the algorithm computes the length and expected value of the identified shortest path, a crucial value utilized by the referee to generate the reinforcement signal. The algorithm is effective in finding the shortest path in various types of stochastic multi-layer graphs. The applicability of the algorithm is closely tied to specific research fields, given the stochastic nature of many complex systems. This algorithm is well-suited for identifying the shortest path in diverse domains, including but not limited to biology, economics, engineering, and social networks. Hence, the development of a stochastic multi-layer shortest path algorithm is a significant contribution to the field of network analysis and has the potential to advance our understanding of complex systems and their interactions. The algorithm's ability to adapt to changing environments and learn from experience makes it a promising approach for analyzing stochastic multi-layer graphs and finding the shortest path between nodes in these graphs.

The proposed CGG-based algorithm for the stochastic multi-layer shortest path (SMSP) problem can be applied in various real-world applications across different domains. In transportation systems, SMSP can be used to find optimal routes considering multiple layers of uncertainty, such as traffic congestion, weather conditions, and road closures. This can aid in efficient route planning for vehicles, leading to reduced travel time and improved transportation management. In telecommunications, SMSP can be applied to optimize routing decisions in multi-layered networks, considering factors such as network congestion, link failures, and varying communication costs. This helps in ensuring reliable and efficient data transmission. SMSP can be utilized in supply chain networks to determine the optimal path for goods or resources in the presence of uncertainties, such as transportation delays, inventory availability, and demand fluctuations. By considering multiple layers of stochasticity, companies can improve their logistics operations, reduce costs, and enhance overall supply chain performance. In energy distribution networks, SMSP can aid in optimizing power flow across interconnected layers, considering factors such as renewable energy generation, demand variability, and transmission line failures. This can help in maximizing the utilization of renewable resources and ensuring an efficient and reliable energy supply.

11 APPENDIX I.

This section brings to light two crucial findings that are paramount to this paper for convergence behavior. At first, it demonstrates that Algorithm 1 can converge to the optimal solution by selecting an appropriate learning rate a when using a learning algorithm L_{R-I} at each node. Secondly, it indicates that by selecting an appropriate learning rate, the probability of obtaining the stochastic multi-layer shortest path (SMSP) with the minimum expected weight is higher than $1-\varepsilon$. We note that the second result supports the connection between the convergence error parameter ε and the learning rate applied in the initial proposed algorithm.

Theorem 1: For a given stochastic multi-layer graph, the probability $q_i(k)$ of constructing the SMSP ω_i at stage k can be updated using Algorithm 1. It further states that for any small positive value ε, there exists a learning rate $a^*(\varepsilon)$ between 0 and 1 such that, for any value of $a \in (0, a^*)$ the following holds:

$$prob\left[\lim_{k\to\infty} q_i(k) = 1\right] \geq 1-\varepsilon \tag{18}$$

The method utilized to illustrate Theorem 1 shares a close resemblance to the approach outlined in [38], which is relied on to analyze the LAs' performance in a dynamic environment. The proof comprises several steps, which are outlined concisely. Initially, we prove that the penalty probabilities for all SMSP approaches the constant values of the final penalty probabilities as k becomes sufficiently large (Lemma 1). Moreover, it's been shown that the probability of selecting the SMSP with the lowest weight is a sub-Martingale process for large k. Therefore, it can be deduced that any alterations in the probability of creating the SMSP are consistently non-negative and proved by Lemmas 2 and 3. Finally, with the aid of Martingale convergence theorems, the convergence of the proposed algorithm is proven.

Lemma 1. Let ω_i is the SMSP penalized with a probability $c_i(k)$ at iteration k and if the $\lim_{k\to\infty} c_i = c_i^*$, then for any value $\varepsilon \in (0,1)$and any stage $k>K(\varepsilon)$, the following statement holds:

$$-prob[|c_i^* - c_i(k)| > 0] < \varepsilon, \tag{19}$$

where $K(\varepsilon)$ indicates the minimum number of stages of algorithm 1 to achieve error rate ε.

Proof: Before presenting the proof of Lemma 1, we address the convergence of $c_i(k)$ to c_i^* (as k tends to infinity). This discussion follows from the insights derived from Algorithm 1. According to this algorithm, the probability of penalizing SMSP ω_i during stage k is determined by the $c_i(k) = prob[\bar{W}_{\omega_i} > T_k]$, meaning that each activated LA is penalized based on selected action, such that the average weight of the built SMSP is greater than the optimal threshold T_k. In the initial stages of the algorithm (for small values of k), the optimal threshold T_k and average weight $\bar{W}_{\omega_i}$deviate significantly from their true values. This creates a situation where certain SMSPs that are close to optimal are penalized, while SMSPs with higher weights are rewarded. By employing learning automata and sampling from the graph, the algorithm progressively converges T_k and $\bar{W}_{\omega_i}$ to their precise values. Consequently, the algorithm gains a deeper understanding of the random environment's characteristics with the growth of k. More precisely, as k increases, the probability of penalizing the optimal or nearly optimal SMSP decreases, while the probability of penalizing other solutions increases. Therefore, when k reaches sufficiently high values, the probability of

penalizing the SMSP ω_i approaches its true value c_i^*. Assuming r represents the total count of constructed SMSPs and c_i^* represents the eventual probability value of $c_i(k)$ as k approaches a large value, we can apply the weak law of large numbers may conclude that:

$$\lim_{k\to\infty} prob[|c_i^* - c_i(k)| > \varepsilon] \to 0. \qquad (20)$$

For every $\varepsilon \in (0,1)$, there exists a $a^*(\varepsilon) \in (0,1)$ and $K(\varepsilon) < \infty$ such that for all $a < a^*$ and $k > K(\varepsilon)$ we have, $prob[|c_i^* - c_i(k)| > 0] < \varepsilon$ and the proof is completed.■

Lemma 2. Let $c_j(k)$ represent the probability of $[\bar{W}_{\omega_j}(k+1) > T_k]$ where $\bar{W}_{\omega_j}(k+1)$ is the average reward obtained by SMSP ω_j(for all $j = 1,2,\dots,r$). Additionally, let $d_j = 1 - c_j$ represent the probability of penalizing or rewarding SMSP ω_j. If $\underline{q}(k)$ follows from algorithm1, then the conditional expectation of $q_i(k)$ is defined as

$$E[q_i(k+1)|q(k)] = \sum_{j=1}^{r} q_j(k)\left[c_j(k)q_i(k) + d_j(k)\left(\prod_{v_m \notin \omega_i}(1 - p_m^1(k)\prod_{v_m \in \omega_i}(p_m^1(k))\right)\right] \qquad (21)$$

Where

$$p_m^1(k+1) = \begin{cases} p_m^1(k) + a(1 - p_m^1(k)) & v_m^l \in \omega_j \\ p_m^1(k)(1-a) & v_m^l \notin \omega_j \end{cases} \qquad (22)$$

Where p_m^1 corresponds to the probability that vertex v_m^l as a member of SMSP, at iteration k and r is the remaining SMSP achieved until iteration k. $v_m^l \in \omega_j$, if it declares itself as a member of SMSP and $v_m^l \notin \omega_j$.

Proof: The proposed algorithm leverages a reinforcement learning scheme, known as L_{R-I}, to continuously update the probability vector at each stage k. Therefore, when $\omega_j$is penalized by the random environment, the construction probability of the SMSP ω_i (i.e, $q_i(k)$) remains unchanged. Also, when the candidate SMSP ω_j is rewarded, the probability of choosing the vertices that belong to SMSP ω_i, increasing by a given learning rate with respect to other vertices.

Assuming $q_i(k) = \prod_{v_m^l \notin \omega_i} p_m^0(k) \prod_{v_m^l \in \omega_i} p_m^1(k)$ be the probability of constructing SMSP, $\omega_i$at iteration k, where p_m^0 and p_m^1 indicate as the probability with which vertex v_m^l is the member of the shortest path and does not belong to the shortest path respectively. The conditional expectation of $q_1(k+1)$, assuming $\underline{q}(k)$ evolves according to the proposed algorithm, is defined as

$$E[q_1(k+1)|q(k)] = \sum_{j=1}^{r} q_1(k)\left[c_j(k)q_1(k) + d_j(k)\left(\prod_{v_m \notin \omega_i}(p_m^0(k)\prod_{v_m \in \omega_i}(p_m^1(k))\right)\right] \qquad (23)$$

Where

$$p_m^0(k+1) = p_m^0(k+1) = \begin{cases} p_m^0(k) + a(1 - p_m^0(k)) & v_m^l \notin \omega_j \\ p_m^0(k)(1-a) & v_m^l \in \omega_j \end{cases} \qquad (24)$$

$$p_m^1(k+1) = \begin{cases} p_m^1(k) + a(1 - p_m^1(k)) & v_m^l \notin \omega_j \\ p_m^1(k)(1-a) & v_m^l \in \omega_j \end{cases} \qquad (25)$$

Since based on the proposed algorithm each LA has two choices, the probability with which vertex v_m^l be a member of SMSP (i.e p_m^0) is defined as $1 - p_m^1$, we obtain

$$E\left[q_j(k+1)\middle|q(k)\right] = \sum_{j=1}^{r} q_j(k)\left[c_j(k)q_i(k) + d_j(k)\left(\prod_{v_m \notin \omega_i}(1 - p_m^1(k))\prod_{v_m \in \omega_i}(p_m^1(k))\right)\right] \quad (26)$$

Where

$$p_m^1(k+1) = \begin{cases} p_m^1(k) + a(1 - p_m^1(k)) & v_m^l \in \omega_j \\ p_m^1(k)(1-a) & v_m^l \notin \omega_j \end{cases} \quad (27)$$

And hence the proof of the lemma.■

Lemma 3. Let $q_i(k)$ denotes the probability of the formation of the SMSP ω_i, at stage k. The increment in the conditional expectation of $q_i(k)$ is always non-negative $\underline{q}(k)$ is updated according to algorithm 1. That is, $\Delta q_i > 0$.

Let define $\Delta q_i(k) = E[q_i(k+1)|q(k)] - q_i(k)$ from lemma 2. By applying lemma 2, we observe the following result:

$$\Delta q_i(k) = \sum_{j=1}^{r} q_j(k)\left[c_j(k)q_i(k) + d_j(k)\left(\prod_{v_m \notin \omega_i}(1 - p_m^1(k)\prod_{v_m \in \omega_i}(p_m^1(k))\right)\right] - q_i(k) \quad (28)$$

Where

$$p_m^1(k+1) = \begin{cases} p_m^1(k) + a(1 - p_m^1(k)) & v_m^l \notin \omega_j \\ p_m^1(k)(1-a) & v_m^l \in \omega_j \end{cases} \quad (29)$$

Let $\prod_{n\in\{0,1\}} \omega_i \pi_m^n(k) = (\prod_{v_m \notin \omega_i}(1 - p_m^1(k)\prod_{v_m \in \omega_i}(p_m^1(k))$, for notational convenience. Therefore, we have

$$\Delta q_i(k) = \sum_{j=1}^{r} q_j(k)\left[c_j(k)q_i(k) + d_j(k)\right] - q_i(k) \quad (30)$$

Where

$$\pi_m^n(k+1) = \begin{cases} p_m^n(k) + a(1 - p_m^n(k)) & v_m^l \notin \omega_j, n = 0 \text{ or } v_m^l \in \omega_j, n = 1 \\ p_m^n(k)(1-a) & v_m^l \notin \omega_j, n = 1 \text{ or } v_m^l \in \omega_j, n = 0 \end{cases} \quad (31)$$

Where $\pi_m^n(k)$ is the probability with which the vertex v_m^l declare itself as belonging to SMSP and not to belong to the SMSP otherwise. Hence, the by-product of the probability for the vertices that belong to SMSP, we obtain, the probability of constructing, rewarding, and penalizing a SMSP and we have:

$$\Delta q_i(k) = \sum_{j=1}^{r} \prod_{\substack{\omega_j \\ n\in\{0,1\}}} p_m^n(k)\left[\prod_{\substack{\omega_j \\ n\in\{0,1\}}} c_m^n(k) \prod_{\substack{\omega_i \\ n\in\{0,1\}}} p_m^n(k) + \left(\prod_{\substack{\omega_j \\ n\in\{0,1\}}} d_m^n(k) \prod_{\substack{\omega_j \\ n\in\{0,1\}}} \pi_m^n(k)\right] - \prod_{\substack{\omega_i \\ n\in\{0,1\}}} p_m^n(k) \quad (32)$$

Where $\pi_m^n(k)$ is determined by equation (11), $c_m^n(k)$ indicates as the probability of penalizing the action that is selected by vertex v_m^l at iteration k and $d_m^n(k) = 1 - c_m^n(k)$.

$$\Delta q_i(k) = \prod_{\substack{\omega_j \\ n\in\{0,1\}}} E[p_m^n(k+1)|p_m(k)] - \prod_{\substack{\omega_j \\ n\in\{0,1\}}} p_m^n(k) \tag{33}$$

The $\Delta q_i(k)$ be either greater or equal to:

$$\Delta q_i(k) \geq \prod_{\substack{\omega_j \\ n\in\{0,1\}}} (E[p_m^n(k+1)|p_m(k)] - p_m^n(k)) = \prod_{\substack{\omega_j \\ n\in\{0,1\}}} \Delta p_m^n(k) \tag{34}$$

And

$$\Delta p_m^n(k) \underset{n\in\{0,1\}}{} = a\, p_m^n(k) \sum_{s\neq n} p_m^s(k).(C_m^s(k) - C_m^n(k)) \tag{35}$$

Where $q_i(k)$ is in range of the (0,1) for all $\underline{q} \in S_r^0$, and $S_r = \{\underline{q}(k): 0 \leq q_i(k) \leq 1; \sum_{i=1}^r q_i(k) = 1\}$. Also, S_r^0 indicates the interior of S_r. We may conclude that, $p_m^n(k) \in (0,1)$ for all, n. Since action α_m^n is the action with minimum probability which can be selected by automation A_m, it is shown that $C_m^{S^*} - C_m^{n^*} > 0$, for all $s \neq n$, where $C_m^{S^*}$ is the final value to which the penalty probability $C_m^{S^*}$ is converged. Lemma 1 shows that when k is a large value, the difference between $C_m^{S^*}$ and C_m^n is positive. Hence, for a large value of k, the right side of the equation contains only non-negative quantities. So, we can confidently say that.

$$\prod_{\substack{\omega_i \\ n\in\{0,1\}}} a.\, p_m^n(k) \sum_{s\neq n} p_m^s(k).(\, c_m^s(k) - c_m^n(k)) \geq 0 \tag{36}$$

And from equation (12), we have

$$\Delta q_i(k) \geq \prod_{\substack{\omega_i \\ n\in\{0,1\}}} a.p_m^n(k) \sum_{s\neq n} p_m^s(k).(\, c_m^s(k) - c_m^n(k)) \geq 0 \tag{37}$$

That concludes the proof of this lemma.■

Corollary 1. The set of unit vectors in $S_r - S_r^0$, where $S_r^0 = \{\underline{q}(k): q_i(k) \in (0,1); \sum_{i=1}^r q_i(k) = 1\}$, represents the set of entire absorbing barriers of the Markov process $\{\underline{q}(k)\}_{k\geq 1}$.

Proof: This corollary has already been demonstrated in[25].■

Let $\Gamma_i(q)$ represent the probability of the algorithm converging to the unit vector e_i using the initial probability vector $\underline{q}$. It can be defined as $\Gamma_i(q) = prob\left[q_i(\infty) = 1 \middle| \underline{q}(0) = \underline{q}\right] = prob[q^* = e_i|\underline{q}(0) = \underline{q}]$. Moreover, let $C(S_r): S_r \to \mathcal{R}$ be the state space of all real-valued continuously differentiable functions with bounded derivative defined on S_r, where $\mathcal{R}$ is the real line. If $\psi(.) \in C(S_r)$, the operator U is defined by

$$U\psi(q) = E[\psi q(k+1)|q(1) = q], \tag{38}$$

where $E[.]$ represents the mathematical expectation. It has been shown in [25] that operator U is linear, and as the expectation of a non-negative function remains non negative, operator U preserves the non-negative functions. In other words, $U\psi(q) \geq 0$ for all $q \in S_r$, if $\psi(q) \geq 0$. If operator U is repeatedly applied n times (for all n>1), we have

$$U^{n-1}\psi(q) = E[\psi q(k+1)|q(1) = q] \quad (39)$$

A function $\psi(q)$ is referred to as super-regular (sub-regular) if and only if $\psi(q) \geq U\psi(q) (\psi(q) \leq U\psi(q))$ for all $q \in S_r$. It has also been demonstrated in [69] that $\Gamma_i(q)$ is the sole continuous solution to the equation $U\Gamma_i(q) = \Gamma_i(q)$, along with specific boundary conditions.

$$\Gamma_i(e_i) = 1 \quad (40)$$
$$\Gamma_i(e_j) = 0, j \neq i$$

Let's define $\phi_i[x,q]$ as $\frac{e^{-xq_i/a}-1}{e^{-x/a}-1}$, where $x > 0$ is chosen. The function $\phi_i[x,q]$ belongs to $C(S_r)$ and fulfills the mentioned boundary conditions.

Theorem 2: Let $\psi_i(.) \in C(S_r)$ is sub-regular with the same boundary conditions, then

$$\psi_i(q) \geq \Gamma_i(q) \quad (41)$$

For all $q \in S_r$.

Proof. Let $\psi_i(.)$ be sub-regular, that is,

$$\psi_i(q) - U\psi_i(q) \geq 0 \quad (42)$$

For all $q \in S_r$. Applying U again to the l.h.s. of equation (42) and recognizing the fact that U is linear and preserves nonnegative functions,

$$U\psi_i(q) - U^2\psi_i(q) \geq 0 \quad (43)$$

A repeated application of the operation yields

$$U\psi_i(q) \geq U^2\psi_i(q). \geq \cdots \geq U^\infty\psi_i(q). \quad (44)$$

We know that $q(n)$ converges w.p.1. It follows that $U^\infty\psi_i(q)$ exists and is unique. Moreover,

$$U^\infty\psi_i(q) = E[\psi_i(q(\infty))|q(0) = q] = \sum_{j=1}^{r} \psi_i(e_j) \Pr[q(\infty) = e_j|q(0) = q] = \Pr[q(\infty) = e_i|q(0) = q] \quad (45)$$

Since $\psi_i(e_j) = 0$ for $j \neq i$ and $\psi_i(e_j) = 1$.

Thus, by inspection of the r.h.s of equation (44), we have

$$U^\infty\psi_i(q) = \Gamma_i(q) \quad (46)$$

From (42), (44) and (46) it follows that $\psi_i(q) \geq \Gamma_i(q)$.■