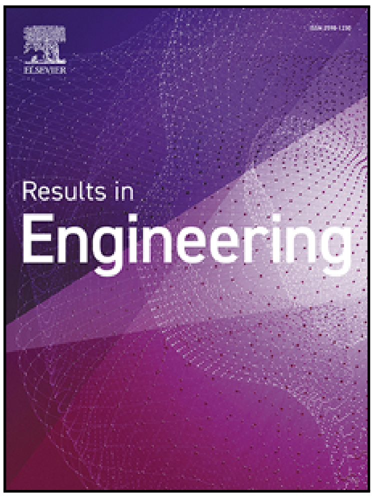


Efficient Identification of Maximum Independent Sets in Stochastic Multilayer Graphs with Learning Automata

Mohammad Mehdi Daliri Khomami , Mohammad Reza Meybodi , Alireza Rezvanian

PII: S2590-1230(24)01478-6
DOI: https://doi.org/10.1016/j.rineng.2024.103224
Reference: RINENG 103224

To appear in: *Results in Engineering*

Received date: 12 August 2024
Revised date: 9 October 2024
Accepted date: 22 October 2024

Highlights

- Defining the Maximum Independent Sets in Stochastic Multilayer Graphs
- Presenting efficient algorithms for solving Maximum Independent Sets in Stochastic Multilayer Graphs
- Leveraging learning automata in the proposed algorithm to serving efficient samples
- Presenting theoretical properties of the proposed algorithm

Efficient Identification of Maximum Independent Sets in Stochastic Multilayer Graphs with Learning Automata

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: Investigating the maximum independent set in stochastic multilayer graphs provides critical insights into the structural and dynamical properties of complex networks. Recently, stochastic multilayer graphs effectively model the intricate interactions and interdependencies inherent in real-world systems, including social, biological, and transportation networks. The identification of a maximum independent set -comprising nodes without direct connections- offers a significant understanding of phenomena such as information diffusion, resource allocation, and epidemic spread within complex social networks. For instance, independent sets play a crucial role in identifying influencers -individuals who profoundly impact their peers, propagating information or opinions widely. In this paper, we introduce the stochastic version of the maximum independent set and propose five algorithms based on learning automata to identify maximum independent sets in the stochastic multilayer graphs. Our approach leverages learning automata to provide a guided sampling from candidate independent sets of the stochastic multilayer graph, aiming to identify the independent set with the maximum expected value while utilizing fewer vertex samples than standard methods that do not incorporate the learning. In addition to proving several mathematical properties of the proposed approach, simulations conducted across diverse stochastic multilayer graphs demonstrate that our learning automata-based algorithms outperform traditional approaches, achieving higher convergence rates and requiring fewer samples.

Keywords: Independent set, Learning Automata, Multilayer Graphs, Stochastic Multilayer Graphs, Learning Automata.

[*] Corresponding author

1 Introduction

In recent years, multilayer social networks have attracted significant interest due to their ability to capture the complexity and diversity of real-world social interactions [1]. Traditional social network analysis focuses on single-layer networks, considering only one type of relationship among individuals. However, in many real-world scenarios, individuals interact across multiple domains or layers, such as friendship [2], communication [3], collaboration [4], and online social platforms [5]. Multilayer social networks provide a more comprehensive and realistic representation of social interactions, enabling researchers to analyze and understand complex systems with multiple interacting components [6]. In addition, multilayer social networks have been used to analyze user behavior, information diffusion, and community detection on platforms such as *Twitter* [7], *Facebook* [8], and online forums. Also, incorporating multilayer network information can improve personalized recommendations by considering multiple aspects of user preferences and interactions. Moreover, multilayer networks allow researchers to investigate the role of social influence in shaping individuals' opinions and decision-making processes [9]. However, modeling deterministic multilayer networks with multilayer graphs may not be suitable for solving most real-network problems due to the unpredictable and uncertain nature of their structural and behavioral parameters [10]. Consequently, stochastic multilayer graphs—where vertex weights are treated as random variables—may represent a more effective and promising approach for analyzing real-world social networks. By treating the graph model as stochastic, a set of particular concepts such as degree, path, cover, and independent set can also be regarded as stochastic [11]. One application of stochastic graphs is in modeling and analyzing complex systems with multiple interactions that have unpredictability or uncertainty. Stochastic multilayer graphs are graphs where the edges or nodes are associated with random variables. For example, consider a scenario in which each edge in the graph represents the probability of two individuals being friends on a social media platform. The strength of their friendship could be modeled as a random variable with a probability distribution [12]. By analyzing the stochastic multilayer graph, we can study various properties of the social network, such as the spread of information or the formation of communities in different social networks, while taking into account the inherent uncertainty in the relationships [11].

Stochastic multilayer graphs can also be used in other domains such as transportation networks, biological networks, financial networks, and other related domains. In these cases, the unpredictability or uncertainty may arise from factors like traffic flow, genetic mutations, or market fluctuations. A key challenge in the realm of social networks is identifying the largest independent group of people within these networks. This problem is known as the independent set in the context of graph theory and has various applications [13]. In the context of multilayer social networks, an independent set refers to a group of individuals who are not directly connected. One application of independent sets in multilayer social networks is identifying influential users or nodes. In this context, influential users are those who have a significant impact on the spread of information or influence others [14]. By finding independent sets within these networks, one can identify groups of users who are not directly connected but have the potential to influence a large number of people [15]. By analyzing the characteristics and behaviors of individuals through the independent set, one can gain insights into their influence strategies, content preferences, or patterns of interaction. This information can then be leveraged to design effective strategies for maximizing the reach and impact of messages or interventions within the stochastic multilayer social network [16]. In this paper, we introduce the notion of the stochastic maximum independent set specifically for stochastic multilayer graphs. We propose five distinct algorithms that leverage learning automata to identify the maximum independent set within these networks, where the probability distribution functions for the vertex weights remain unknown. The stochastic maximum independent set is defined as an independent set that maximizes the expected value. Our proposed algorithms sample from the stochastic multilayer graphs to ascertain the independent set with the highest expected value, employing learning automata in the process. The sampling methodology is optimized to reduce the number of samples required to identify the independent set with the maximum expected weight. In these algorithms, each vertex of the multilayer graph is associated with a learning automaton that can take one of two possible actions: to become a member of the independent set or to abstain from it. The learning automata execute a guided sampling process to acquire additional samples from the vertices of the multilayer networks that show potential. Specifically, they target the vertices situated along the paths leading to the independent set with the maximum expected weight.

We conducted extensive experimental investigations on both real and synthetic multilayer graphs to evaluate the performance of the proposed algorithms. The findings indicate that these algorithms significantly outperform the standard sampling method (SSM) in terms of the sample size required to identify the independent set with the highest expected weight in stochastic multilayer graphs [17]. Furthermore, our research demonstrates that with appropriate parameter selection, the probability of successfully identifying the independent set with the maximum expected value approaches one.

The structure of the paper is as follows: Section 2 covers the foundational concepts related to stochastic multilayer graphs and learning automata, along with an introduction to the Standard Sampling Method (SSM). Section 3 outlines the proposed algorithms for finding the maximum independent set and its variations in stochastic multilayer graphs, including a mathematical analysis of their convergence characteristics as well as the time and space complexity of the proposed algorithm. Section 4 presents the simulation results, and section 5 provides a brief discussion regarding the proposed algorithms. Finally, section 6 concludes the paper.

2 Preliminaries

In this section, we present key foundational concepts necessary for the subsequent discussions in the paper. This encompasses a concise overview of the maximum independent set problem, learning automata, stochastic multilayer graphs, the stochastic maximum independent set, and the conventional sampling method.

2.1 Independent Set

The independent set is a key concept in graph theory that has been extensively studied [18]. In a single-layer graph, an independent set consists of a subset of vertices in which no two vertices are connected by an edge. In other words, an independent set is a collection of vertices such that none of them share a direct edge. The challenge of finding the maximum independent set in a single-layer graph is classified as NP-hard [19]. Consequently, researchers have developed and proposed various algorithms to address this challenge. In [20] introduced the concept of NP-completeness and included the independent set problem as one of the first examples. *Aggrawal et al.* [21] discuss the introduction of optimized crossover for Genetic Algorithms for the optimization of independent sets. They tested their idea on the independent set problem and presented computational results that demonstrate its effectiveness, at least for a specific case. However, they acknowledge that not all problems can be solved using optimized crossover, so they propose some general principles to consider when determining if a problem can be successfully solved using this approach. In [22] introduces an algorithm for solving the Maximum Independent Set (MIS) problem and offers an exact estimate of the accuracy of the approximate solution. They introduced a new concept and presented theoretical research results.

The properties of the algorithm are examined, particularly its convergence within the class of trees. Additionally, the authors present the findings from computational experiments carried out on complementary graphs from DIMACS, which demonstrate both the high quality of the approximate solutions and the algorithm's efficiency in terms of time complexity. *Plotnikov et al,* [23] describe the development of a heuristic algorithm for finding the maximum independent set of vertices in a directed graph. The algorithm utilizes the concept of finite partially ordered sets, specifically partitioning the set into a minimum number of chains. A specially directed graph is created, and a solution algorithm is proposed based on a hypothesis about its properties. In [24] presents a heuristic algorithm known as ILS-VND for addressing the Maximum Weighted Independent Set (MWIS) problem. The approach integrates Iterated Local Search (ILS) with Variable Neighborhood Descent (VND). Additionally, the authors introduced two novel neighborhood structures and developed a reactive perturbation mechanism. Experimental results on benchmark instances demonstrate that LS-VND surpasses existing methods in terms of solution quality.

In [25], the authors addressed the issue of identifying maximum independent sets by employing reduction rules to convert the problem into a more manageable instance known as a kernel. This kernel can subsequently be solved rapidly using either exact or heuristic algorithms, or by recursively applying kernelization within the branch-and-reduce framework. The performance of the proposed algorithm relies heavily on efficient and rapid kernelization that generates compact kernels. Additionally, the authors introduced an effective parallel kernelization algorithm founded on graph partitioning and parallel bipartite maximum matching techniques. Piao et al [26] concentrate on examining and enhancing the algorithms employed to compute the Maximum Independent Set (MIS). Their research explores a scheduling framework that dynamically implements various reduction and greedy algorithms based on benefit-cost analysis. They propose a novel data structure known as RGraph, which mitigates the worst-case time complexity associated with degree-two reduction. Furthermore, they develop a more effective vertex addition strategy, termed the vertex index and lazy update mechanism, to improve overall time efficiency.

Khomami et al. [13] present an algorithm based on learning automata to tackle the Maximum Independent Set (MIS) problem. In this approach, each vertex in the graph is paired with a learning automaton that collaborates with others to identify a potential independent set. The algorithm iteratively adjusts the action probability vector of the learning automata, steering them toward the discovery of the maximum independent set.

In [27], the authors explore the challenge of computing high-quality independent sets within the realm of combinatorial optimization. They note that earlier algorithms typically employed kernelization techniques to derive exact maximum independent sets in medium-sized sparse graphs and to locate high-quality independent sets in larger, sparse graphs where precise solutions are more challenging to achieve. They propose a method demonstrating that applying straightforward kernelization techniques in an online manner can enhance the performance of local search, offering a faster alternative to pre-computing a kernel using more complex techniques. Furthermore, the removal of high-degree vertices significantly boosts local search efficiency, particularly in large, intricate networks characterized by sparse connections. In [28], the authors investigate the maximum weight-independent set problem within a vertex-weighted graph and introduce the branch-and-reduce paradigm as a solution strategy. The objective is to identify a set of non-adjacent vertices that yield the highest total weight. The branch-and-reduce methodology applies data reduction rules to diminish the problem's size, ensuring that an optimal solution for the reduced problem can be swiftly adapted to yield an optimal solution for the original problem. Additionally, they present new generalized data reduction and transformation rules to address the issue effectively.

In [29], the significance of determining the number of clusters in clustering algorithms is examined, particularly within the framework of gene co-expression networks. The authors note that the number of clusters in such networks is not predefined. They propose leveraging the concept of the maximum independent set from graph theory as a means to estimate the number of clusters in a gene expression dataset. *Maher* et al. [30] conducted a comparison of self-stabilizing vertex cover (VC) and independent set (IS) algorithms for addressing the link monitoring problem in wireless sensor networks (WSNs). Their implementation and simulation of existing algorithms revealed that IS algorithms yield smaller and more effective covers than their VC counterparts. In [31], the authors explored the application of algorithmic tile self-assembly for tackling the maximum independent set problem, a well-recognized combinatorial optimization challenge. They introduced a method that employs tile self-assembly to solve this problem using three small systems: the nondeterministic guess system, the AND operation system, and the comparing system. This approach allows for the parallel resolution of the maximum independent set problem in approximately polynomial time, providing a cost-effective solution within the realm of nanotechnology. Gencer et al. [32] also examined the importance of the maximum independent set problem as a combinatorial optimization issue with diverse applications in science and engineering, and they proposed a genetic algorithm-based strategy for its resolution. Their approach incorporates condition-based genetic operators to locate IS effectively.

In reference [33], the authors proposed a novel method named MISNN, which reduces a graph to a neural network structure based on its connectivity. The input for the MISNN is generated by solving a box-constrained nonlinear optimization problem, from which the Maximum Independent Set (MIS) is derived using a specific mapping. Reference [34] examines algorithms aimed at identifying maximal independent sets in both randomly generated and manually constructed graphs. Their research has practical implications, particularly in areas such as register allocation in compilers, channel assignment for radio stations, examination scheduling, and graph coloring. Additionally, the study explored the concept of maximal independent sets as a particular type of dominating set within a graph and demonstrated that any dominating set that is also independent must necessarily be maximal independent. As a result, maximal independent sets are frequently referred to as independent dominating sets. Wang et al. [35] introduced a new binary neuron model called the stochastic neuron model for Hopfield neural networks. This model aims to resolve the issue of local minima in these networks. To achieve this, the stochastic neuron model integrates a stochastic mechanism into the neuron's input/output function, enabling the network to occasionally transition to higher-energy configurations. By employing the stochastic neuron model, the neural network can potentially escape local minima and progress toward more optimal states, including the global minimum. Furthermore, to assess the effectiveness of the proposed neuron model, the researchers applied it to the maximum independent set problem and conducted extensive simulations.

In [36], the authors addressed the challenge of finding the maximum independent set (MIS) and independent set (IS) in dynamic graphs, where the network structure evolves over time. They proposed a baseline algorithm that updates the MIS/IS at a given point based on the MIS/IS from the previous iteration, rather than recalculating it from scratch. To mitigate the computational complexity associated with determining an exact MIS, they developed an efficient constant-time algorithm known as LSTwo, which provides a high-quality (large-size) independent set. Additionally, they introduced a lazy search algorithm designed to generate even higher-quality independent sets. In [37], the authors developed a parallel algorithm to identify a near-maximum independent set in a circular graph, which has also been adapted to predict the secondary structure of RNA molecules. This approach employs a system of multiple neural networks organized in an array, with the number of networks corresponding to the number of edges in the circular graph or the potential number of base pairs. This system not only identifies a near-maximum independent set but also predicts the RNA's secondary structure through multiple iterations.

In [38], the focus was on calculating a maximum quasi-independent set while adhering to constraints regarding the number of conflicting edges. The authors introduced an innovative strategy called pay-and-recycle, which intertwines the initialization phase for near-MIS and edge expansion. Instead of using greedy peeling when no reduction rules can be applied, this method removes an edge to create conditions conducive to applying reduction rules. Furthermore, unnecessary edges are identified and recycled for additional expansion, enhancing overall performance. They also proposed an effective technique that leverages offline samples to assist in guiding edge expansion. The authors in [39], presented an overview of genetic algorithms as a computational strategy for solving optimization problems in the field of evolutionary computation. They introduced a new genetic algorithm specifically designed to tackle the maximum independent set problem using an elitist strategy. Deng et al [22] explore the use of Differential Evolution (DE) for solving subset problems, with a particular focus on the Maximum Independent Set Problem (MISP). They note that traditional DE techniques have primarily been applied to continuous optimization problems. In response, the authors introduce a novel approach known as Structure-encoding Differential Evolution (SEDE) tailored for subset problems. The efficacy of SEDE is assessed through a computational study involving instances generated by specific methods. The results are benchmarked against other heuristic algorithms, showcasing SEDE's promising capabilities in tackling the MISP.

Zheng et al [40] examined the Maximum Independent Set (MIS) problem within the context of optimization in graph theory, highlighting its practical applications. They argue that previous research has often neglected the unequal weights attributed to vertices in real-world situations, leading to the conclusion that the weight of an MIS may not always be maximal. To address this oversight, they investigated the Maximum Weighted Independent Set (MWIS) problem, which seeks to identify the set of independent vertices with the highest total weight. Given the computational intensity of finding an independent set, they proposed an exact algorithm based on a reduction-and-branching strategy. Furthermore, to accommodate large graphs where exact solutions are infeasible, they developed an efficient greedy algorithm aimed at calculating a near-optimal weighted independent set. In [41], the authors tackled the resource allocation problem in Device-to-Device (D2D) communications within OFDMA-based cellular systems. Their algorithm integrates concepts from graph theory and the Knapsack problem to facilitate efficient resource allocation. The conflict graph for D2D pairs is constructed utilizing maximal independent sets, which are subsequently employed iteratively as inputs to the Knapsack problem to create D2D groups and allocate sub-channels. Moreover, the proposed approach demonstrates superior performance compared to graph coloring, particularly in scenarios with a large number of D2D pairs within cellular systems, as evidenced by increased average data rates. Li et al [42] expand upon the ant colony optimization heuristic to address maximum independent set (MIS) problems.

Unlike other problems such as the traveling salesman problem (TSP), which involve the concepts of "path" or "order" in their solutions, MIS problems have distinct characteristics that necessitate modifications to the ant colony optimization approach. These adaptations include the introduction of a new computational method for heuristic information, enhancements to the pheromone update rule, and the development of a complementary solution construction process. Simulation results indicate that the proposed ant colony optimization heuristic effectively and efficiently tackles MIS problems.

Bai et al [43] examined the application of Connected Dominating Sets (CDS) as a Virtual Backbone (VB) for routing in ad hoc wireless networks. They clarified that many approximation algorithms designed to construct a compact CDS in these networks utilize a two-phased approach. The first phase typically involves creating a Dominating Set (DS), often derived from a Maximum Independent Set (MIS). The size of the MIS and its relationship to the Minimum Connected Dominating Set (MCDS) is critical for evaluating the performance of these algorithms. They further noted that while the relationship between the sizes of MISs and DSs has been thoroughly investigated in homogeneous wireless networks, modeled as Unit Disk Graphs (UDG) and Unit Ball Graphs (UBG), it remains underexplored in heterogeneous wireless networks represented by Disk Graphs with bidirectional links (DGB) and Ball Graphs with bidirectional links (BGB). Zhu et al [44] addressed the partition coloring problem (PCP), which focuses on selecting one vertex from each partite set in a graph to minimize the chromatic number of the induced subgraph. They established that PCP has various real-world applications, including constraint encoding and scheduling tasks. However, solving PCP for large graphs presents significant challenges due to its NP-completeness. To tackle this, they introduced the concept of a partition independent set (PIS) and presented an efficient algorithm called FastPIS to identify a maximum PIS. By integrating FastPIS with a straightforward coloring procedure, a high-quality initial solution for PCP can be achieved. Additionally, they proposed a reduction rule based on the concept of an ℓ-clustering-degree bound ordered set (ℓ-CDBOS) to iteratively minimize the size of the working graph. These strategies contribute to the development of an effective method called HotPGC for solving PCP.

Joo et al [45] examined the difficulties encountered by scheduling algorithms in achieving optimal throughput and delay performance within network environments. They noted that addressing the Maximum Weighted Independent Set (MWIS) problem is frequently essential for these algorithms; however, the MWIS problem is recognized as NP-hard in most realistic network scenarios. In non-fading environments, there exist low-complexity scheduling algorithms that either converge to the MWIS solution or attain a fraction of the achievable throughput. The authors proposed a low-complexity scheduling scheme that performs effectively in fading channels and can be implemented in a distributed manner. This scheme relies solely on local information, exhibits logarithmic complexity relative to network size, and offers performance guarantees that are comparable to those of the centralized Greedy Maximal Scheduler. In reference [46], the authors explored the challenges faced by Radio Frequency Identification (RFID) systems, particularly in mobile RFID contexts where multiple readers simultaneously interrogate tags, resulting in reader collisions. They concentrated on evaluating the throughput of five NFRA protocols, which include the basic NFRA protocol and four variations, through the lens of the maximum independent set. The interference among readers is represented as an undirected graph, with each maximum independent vertex set corresponding to an optimal solution for reader anti-collision. Furthermore, they established an evaluation framework grounded in the maximum independent set to assess the effectiveness of reader anti-collision protocols.

It is essential to note that significant research has been conducted on addressing the maximum independent set problem within single-layer networks. However, due to the inherently unpredictable and uncertain nature of the structural and behavioral parameters in multilayer networks, it has become necessary to model deterministic multilayer networks as stochastic multilayer graphs. This approach seeks to redefine network characteristics, such as independent sets, to more accurately reflect the complex dynamics of these networks. In this paper, we introduce the concept of the maximum independent set in stochastic multilayer graphs for the first time and propose five algorithms based on learning automata to identify the maximum independent set in stochastic multilayer networks.

2.2 Learning Automata

This section offers a general overview of Learning Algorithms (LA) and their associated learning methods. The learning process unfolds as follows: LA randomly selects an action from a predefined action set. It then executes this action within the environment and assesses the outcome. The environment presents reinforcement signals, which may consist of rewards or penalties, to the LA. In light of these reinforcement signals, LA adjusts its action probability vector and reiterates the learning process. To facilitate this adjustment, LA employs a specified algorithm to update the action probability vector, commonly referred to as a learning algorithm. The primary objective of this learning algorithm is to identify optimal actions that will maximize the average total reward obtained from the environment [47]. The general scheme for the main components of a simple learning automaton is illustrated in figure 1.

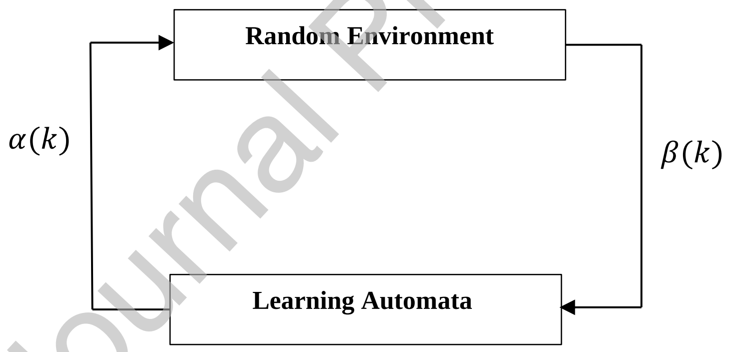


Figure 1: Communication between the learning automata and random environment.

The learning automata (LAs) can be characterized as a 4-tuple $\langle A.B.\mathcal{J}.P(k)\rangle$, where $A = \{\alpha_1.\ldots.\alpha_r\}$ represents a finite set of actions, B denotes the set of all potential inputs or reinforcements to the automata, $\mathcal{J}$ is a learning algorithm that updates the action probabilities, and $P(k)$ is the action probability vector at instance k. The learning algorithm employs a recurrence relation to adjust the action probability vector.

$$\begin{aligned} p_i(k+1) &= p_i(k) + a\big(1 - p_i(k)\big) \\ p_j(k+1) &= p_j(k) - ap_j(k) \;\; \forall j \neq i \end{aligned} \tag{1}$$

$$\begin{aligned} p_i(k+1) &= (1-b)p_i(k) \\ p_j(k+1) &= \frac{b}{r-1} + (1-b)p_j(k) \ \ \forall j \neq i \end{aligned} \tag{2}$$

In the framework of a linear learning algorithm, let $q_i(k)$ denote the action chosen by Learning Automata (LAs) at instance k from the distribution $q(k)$. The update rule governing this algorithm is outlined in Equation (1), where ($\beta = 1$) indicates a favorable response, and ($\beta = 0$) signifies an unfavorable response. The likelihood of receiving an unfavorable response for action q_i is represented as $c_i = \Pr\,[\beta = 0|\alpha_i]$. It is impossible to predict whether the LA will encounter a favorable or an unfavorable response [48].

Let a denote the reward value that governs the increase in the action probability vector, and let b signify the penalty value that dictates the decrease in action probabilities. When $a = b$, the recursive equations (1) and (2) specify the linear reward-penalty (L_{R-P}) algorithm. If a is strictly greater than $b = 0$, the recursive equations characterize the linear reward-ε penalty ($L_{R-\varepsilon P}$) algorithm. Finally, when $b = 0$, the algorithm is referred to as the linear reward inaction (L_{R-I}) algorithm, which indicates that penalizing an action does not affect the action probability vectors [48].

In complex and dynamic settings characterized by considerable uncertainty or insufficient information, learning algorithms are vital for enhancing performance. Learning automata, with their straightforwardness and efficacy in unfamiliar environments, have made noteworthy contributions to this area, as highlighted by numerous studies [14],[11].

2.3 Stochastic multilayer Graph and Stochastic Independent Set

A stochastic multilayer graph $\mathcal{M}$ can be characterized by a 3-tuple $\mathcal{M} = (\mathcal{g}, \mathcal{C}, W)$, where $\mathcal{g} = \{G_q\colon q \in \{1, \dots, M\}\}$ represents a collection of graphs $G_q = (V_q, E_q, W_q)$ known as the layers of $\mathcal{M}$. Here, $E_q \subseteq V_q \times V_q$ denotes the interlayer edges, while $W_q \in W$ is a matrix in which w_q^i is a random variable associated with vertex i in layer q, applicable for all $q \in \{1, \dots, M\}$ if such an edge exists. The set $\mathcal{C}$ denotes the interconnections among nodes belonging to different layers G_q and G_k, where $q \neq k$. A subset of vertices H is deemed independent within H if it contains no edges. We refer to an independent set as maximal if it is not a subset of any larger independent set. A maximum independent set in a stochastic multilayer graph is defined as an independent set that possesses the maximum expected value. Specifically, an independent set H is identified as a subset of V_q such that there is no edge between any pair of vertices u_q and v_q in H for all $q \in \{1, \dots, M\}$, and this set has the highest expected value compared to all possible independent sets.

2.4 Standard Sampling

The theorem presented by Papoulis and Pillai [49] offers a methodology for determining the required sample size for a random variable to achieve a desired confidence level of $1 - \varepsilon_i$.

Utilizing this theorem allows one to calculate the appropriate sample size necessary to attain the specified confidence level [11].

Theorem 1: Let $(x_1, x_2, \dots, x_N)$ represent N random samples from the random variable X_i, which has an unknown mean μ and variance σ^2. If the interval $\overline{x} \pm \sigma/\sqrt{N\varepsilon_i}$, where $\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$, serves as a $(1-\varepsilon_i)\%$ confidence interval for the mean μ, then for any sufficiently small error δ, there exists a positive integer N_0 such that for all N_0, the probability $P[|\overline{x}_n - \mu| < \delta]$ is greater than $1-\varepsilon$.

3 Proposed Algorithm

Let $\mathcal{M} = (\mathcal{g}, \mathcal{C}, W)$ be the input stochastic multilayer graph 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 $\mathcal{M}$, $E_q \subseteq V_q \times V_q$ is an inter-layer edge and $W_q \in W$ is a matrix wherein w_q^i is a random variable associated with vertex i in layer q for all $q \in \{1, \dots, M\}$. The set $\mathcal{C}$ indicates the interconnections between nodes across different layers G_q and G_k with $q \neq k$. The proposed algorithm utilizes a network of learning automata that in its configuration is isomorphic to the input stochastic multilayer graph.
The primary goal of the proposed algorithm is to identify an independent set with maximum expected weight from the stochastic multilayer graph, utilizing sampling through learning automata. This sampling strategy aims to minimize the number of samples taken from the vertices of the stochastic multilayer graph. Each learning automaton A_q^i, assigned to vertex i in layer q, has two possible actions: "Yes" or "No." Specifically, if the learning automaton selects the action "Yes," it indicates that the node is included in the candidate of the independent set, whereas "No" signifies that the node is not part of the candidate of the independent set. The pseudo-code of the proposed algorithm is given in figure 2 and the proposed algorithm is structured into five steps, which are detailed in the following subsections.

3.1 Initialization

A network of learning automata represented as a 2-tuple $\langle A, \alpha \rangle$ that is isomorphic to the input stochastic multilayer graph, is constructed. The set of learning automata is defined as $A = \{A_1, \dots, A_M\}$, with each automaton assigned to a vertex of the input multilayer graph. The associated action set is denoted $\alpha = \{\alpha_1, \dots, \alpha_M\}$. Let $A_q = \{A_q^1, \dots, A_q^n\}$ is the set of automata assigned to layer q where A_q^i is the automaton corresponding to vertex i in layer q. Let $\alpha_q = \{\alpha_q^1, \dots, \alpha_q^n\}$ is the set of actions assigned to layer q where α_q^i indicates as the corresponding action assigned to automaton A_q^i. We note that each learning automaton has two actions. Let $\alpha_q^{i,1}$ and $\alpha_q^{i,2}$ are correspond to two actions with equal initial probabilities. Let Φ_k represent the candidate stochastic independent set at instant k. Initially, all learning automata are inactive.

3.2 Construction Stochastic Independent Set

During this stage, every learning automaton simultaneously selects one of its potential actions according to the action probability vector. In the proposed algorithm, the selection of $\alpha_q^{i,1}$ indicates that the action corresponds to a member of the candidate independent set, while $\alpha_q^{i,2}$ indicates that it does not belong to the candidate independent set. Let $\varphi(k)$ be the candidate stochastic independent set at iteration k. Let v_q^i and v_l^j are two nodes that select their first action at iteration k, then these nodes are added to $\varphi(k)$ and the algorithm checks weather these nodes are connected or not. If these nodes are connected the algorithm tries to find another candidate stochastic independent set in the next round. If the algorithm finds a candidate stochastic independent set the next stage begins starts.

3.3 Computation Candidate Threshold

The algorithm employs a candidate threshold to assess the quality of the discovered solution. The computed candidate threshold at iteration k, denoted as $CT(k)$, is determined by averaging the weights of all the stochastic independent set generated by the algorithm, as computed by equation (3):

$$CT(k) = \frac{1}{k}\sum_{q=1}^{M}\sum_{i=1}^{n}\overline{W}(v_q^i(k)), \qquad (3)$$

where k denotes the number of iterations, while and $\overline{W}(v_q^i(k))$ represents the sum weights of the vertices in $\varphi(k)$. The utilization of a candidate threshold allows the algorithm to determine if a maximum independent set discovered during an iteration possesses the potential to qualify as a candidate for a stochastic maximum independent set. If the independent set discovered during iteration k possesses a weight that is equal to and larger than the dynamic threshold $CT(k)$, it is considered a potential candidate for the stochastic maximum independent set ϕ^*.

3.4 Updating Action Probability Vector

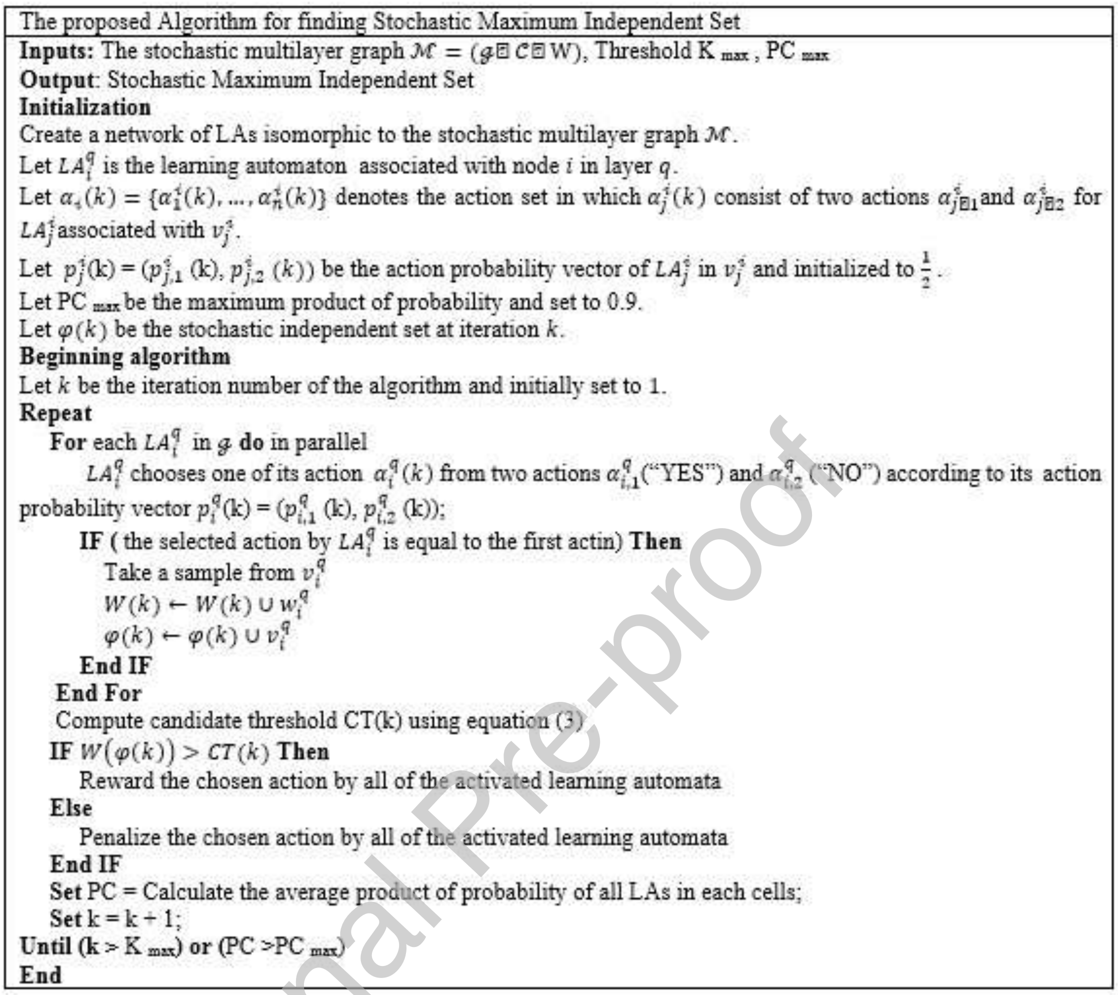
In this stage, the probability vectors associated with the nodes that are members of the candidate stochastic independent set in the input stochastic multilayer graph are adjusted using the learning algorithm, drawing on the results from previous iterations. Moreover, each learning automaton employs an L_{R-I} reinforcement scheme to modify its action probability vector. If the value of $\overline{W}(\varphi(k))$ is equal or greater than the dynamic threshold $CT(k)$, all actions selected by the learning automata belonging to the candidate independent set receive rewards; otherwise, they face penalties.

3.5 Stopping the Algorithm

The iteration process of the algorithm terminates when either the count of iterations k surpasses the predefined threshold K_{max} or when the calculated value $PC(k)$ as defined by Equation (4) reaches the specified threshold P_{max}.

$$PC(k) = \prod_{v_l^i\in\varphi} Max(p(v_l^i)), \qquad (4)$$

where $PC(k)$ represents the product of the highest probabilities found in the probability vectors of the learning automata associated with the vertices of the candidate independent set φ_k during iteration k. Moreover, $p(v_i^l)$ refers to the action probability vector of the learning automaton residing in vertex i in layer q.



```
The proposed Algorithm for finding Stochastic Maximum Independent Set
Inputs: The stochastic multilayer graph $\mathcal{M} = (\mathcal{G}□\mathcal{C}□W)$, Threshold $K_{max}$, $PC_{max}$
Output: Stochastic Maximum Independent Set
Initialization
Create a network of LAs isomorphic to the stochastic multilayer graph $\mathcal{M}$.
Let $LA_i^q$ is the learning automaton  associated with node $i$ in layer $q$.
Let $\alpha_q(k) = \{\alpha_1^q(k), \dots, \alpha_n^q(k)\}$ denotes the action set in which $\alpha_j^q(k)$ consist of two actions $\alpha_{j□1}^q$and $\alpha_{j□2}^q$ for
$LA_j^q$associated with $v_j^q$.
Let $p_j^q(k) = (p_{j,1}^q(k), p_{j,2}^q(k))$ be the action probability vector of $LA_j^q$ in $v_j^q$ and initialized to $\frac{1}{2}$.
Let $PC_{max}$ be the maximum product of probability and set to 0.9.
Let $\varphi(k)$ be the stochastic independent set at iteration $k$.
Beginning algorithm
Let $k$ be the iteration number of the algorithm and initially set to 1.
Repeat
    For each $LA_i^q$ 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 $p_i^q(k) = (p_{i,1}^q(k), p_{i,2}^q(k))$;
        IF ( the selected action by $LA_i^q$ is equal to the first actin) Then
            Take a sample from $v_i^q$
            $W(k) \leftarrow W(k) \cup w_i^q$
            $\varphi(k) \leftarrow \varphi(k) \cup v_i^q$
        End IF
    End For
    Compute candidate threshold CT(k) using equation (3)
    IF $W(\varphi(k)) > CT(k)$ Then
        Reward the chosen action by all of the activated learning automata
    Else
        Penalize the chosen action by all of the activated learning automata
    End IF
    Set PC = Calculate the average product of probability of all LAs in each cells;
    Set k = k + 1;
Until (k > $K_{max}$) or (PC >$PC_{max}$)
End
```

Figure 2: The pseudocode for finding an independent set in a stochastic multilayer graph.

3.6 Improvements

In the subsequent portion of this section, four strategies for the proposed algorithm will be expounded upon. These modifications aim to augment either the algorithm's speed or the overall quality of its solutions.

3.6.1 Strategy 1

The initial algorithm has two limitations that could potentially extend its execution time. Firstly, the subset of learning automata chosen at each stage may not form a valid candidate independent set. Second, the algorithm might gather non-informative samples while constructing the candidate independent set, a result of the independent action choices made by the learning automata. In contrast, the revised algorithm, designated as Algorithm 2, associates the action set of each learning automaton with its degree, distinguishing it from Algorithm 1.

3.6.2 Strategy 2

A proposed solution to mitigate the initial limitation of Algorithm 1 involves employing distributed learning automata. This approach, designated as Algorithm 3, allows each learning automaton to exist in one of two states: active or passive. Initially, all learning automata are configured to the passive state. At each stage, Algorithm 3 activates specific learning automata to construct a candidate of the independent set. Guided by a policy, each activated automaton is included in the candidate independent set. In this algorithm, a learning automaton is chosen at random and then activated, after which the activated automaton is added to the candidate of the independent set. Moreover, this activated automaton selects one of its actions to activate a new learning automaton. We note that, after the action selection, the activated learning automaton changes its internal state to passive. In addition, the new LA selects one of its actions activates another learning automaton, and then changes its internal states to passive. Moreover, the new activated learning automaton is added to the candidate independent set. The process of activating an automaton and action selection is continued until are LAs in stochastic multilayer graphs are met.

3.6.3 Strategy 3

In Algorithm 3, it is assumed that each learning automaton maintains a constant number of actions throughout the process. This reliance on a fixed action-set learning automaton considerably improves the sampling rate. However, to further decrease the sampling rate in Algorithm 3, we propose Algorithm 4, which features a dynamic action set for each learning automaton. In this approach, each passive learning automaton modifies the number of its actions or adjusts its action probability vector by disabling actions linked to the candidate of the independent set. Essentially, the learning automata adopt a variable action set structure.

3.6.4 Strategy 4

To enhance the convergence speed of learning automata, the research outlined in [47] introduced a novel estimator based on the Pursuit algorithm, referred to as CP_{RP}. In the CP_{RP} algorithm, it is assumed that the current action taken is optimal. This optimal action is identified using the estimator vector $\widehat{D}$, which is computed according to equation (5):

$$\widehat{D}_i = \frac{W_i(k)}{Z_i(k)}, \tag{5}$$

where, $Z_i(t)$ represents the cumulative number of times action q_i has been chosen up to iteration t, while $W_i(t)$ indicates the total number of times action q_i has received a reward by iteration t. In equation (5), a higher value of $\widehat{D}_i$ signifies that action q_i is the currently optimal choice for iteration t. According to the CP_{RP} framework, an action is first selected from the set of available options. Following this selection, the action is either rewarded or penalized, with CP_{RP} enhancing the likelihood of the optimal action being chosen in the future. This approach is implemented in Algorithm 1, referenced as Algorithm 5, wherein the learning automaton (LA) operates in pursuit mode.

3.7 Convergence

Theorem 1: Let $q_i(k)$ be the probability of constructing MIS ω_i in a given stochastic multilayer graph at stage k. If $\underline{q}(k)$ evolves according to algorithm 1, then for every $\varepsilon > 0$, there exists a learning rate $a^*(\varepsilon) \in (0, 1)$ such that for all $a \in (0, a^*)$, we have

$$prob\left[\lim_{k\to\infty} q_i\,(k) = 1\right] \geq 1-\varepsilon \qquad (6)$$

Proof: The convergence proof involves several key steps. First, it is established that the penalty probability associated with Maximum Independent Sets (MISs) converges to fixed values of the final penalty probabilities for sufficiently large values of k (Lemma 1). Subsequently, it is demonstrated that the probability of selecting the MIS with the maximum expected weight forms a sub-martingale process for large k. This facilitates the proof of variations in the probability of constructing the maximum MIS using martingale convergence theorems. Therefore, it is essential to prove the following lemmas before presenting the proof of the theorem. Before presenting the proof of the theorem outlined in [50], it is crucial to first establish the validity of the following lemmas. These foundational results will support the subsequent argument and ensure a clear and rigorous progression of the proof.

Lemma 1: if the MIS ω_i is penalized with probability $c_i(k)$ at stage k (i.e., $c_i(k) = prob\left[\overline{W}_{\omega_i} > T_k\right]$), and $\lim_{k\to\infty} c_i\,(k) = c_i^*$, then for each $\varepsilon \in (0,1)$ and $k > K(\varepsilon)$ we have, $prob[|c_i^* - c_i(k)| > 0] < \varepsilon$.

Proof: Let $c_j(k)$ (for all $j = 1, 2, \ldots, r$) represent the probability with which the maximum independent set (MIS) ω_i is penalized at stage k, where r denotes the total number of MISs in G. Let c_j^* represent the probability $c_j(k)$ for large values of k. By invoking the weak law of large numbers, it follows that:

$$\lim_{k\to\infty} prob[|c_i^* - c_i(k)| > \varepsilon] \to 0, \qquad (7)$$

for every $\varepsilon \in (0{,}1)$, there exists a $a^*(\varepsilon) \in (0,1)$ and $K_0(\varepsilon) < \infty$ such that for all $a < a^*$ and $k > K_0$ we have, $prob[|c_i^* - c_i(k)| > 0] < \varepsilon$. This completes the proof of Lemma 1.

Lemma 2: Let $c_j(k) = prob[\overline{W}_{\omega_j}(k+1) > T_k]$ and $d_j(k) = 1 - c_j(k)$ represent the probabilities of penalizing and rewarding the MIS ω_j for all $j = 1, 2, \ldots, r$, repectivelly.
Assuming that $\underline{q}(k)$ evolves under Algorithm 1, the conditional expectation of $q_i(k)$ can be defined as follows:

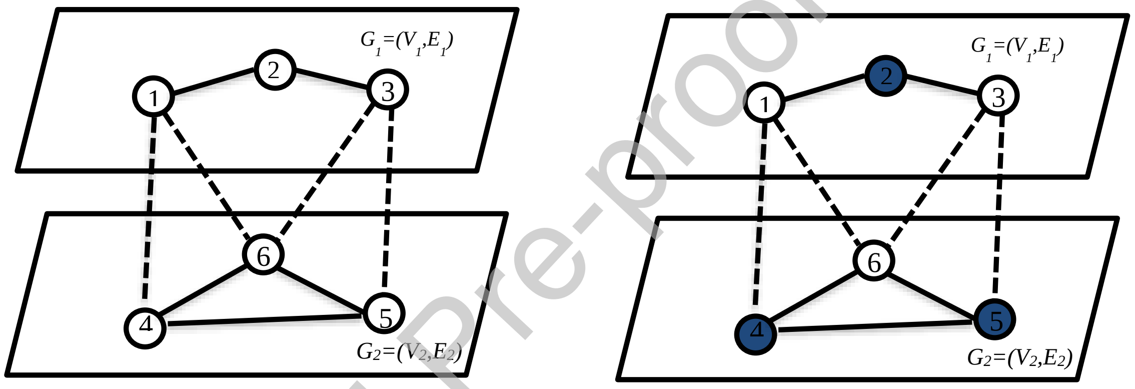
$$E[q_i\,(k+)|q(k)] = \textstyle\sum_{j=}^{r} q_j\,(k)[c_j(k)q_i(k) + d_j(k)\left(\prod_{v_m\notin\omega_i}(1 - p_1^m(k)\right)\prod_{v_m\notin\omega_i}(p_1^m(k))], \qquad (8)$$

where

$$p_1^m(k+1) = \begin{cases} p_1^m(k) + a(1 - p_1^m(k)) & v_m \in \omega_j \\ p_1^m(k)(1-a) & v_m \notin \omega_j \end{cases}. \qquad (9)$$

Let $p_1^m(k)$ represent the probability that vertex v_m declares itself as a member of the independent set at stage k. If vertex v_m declares itself as a member of the independent set (i.e., selects action α_m^1, then v_m belongs to ω_j; otherwise, v_m does not belong to ω_j.

Proof: In Algorithm 1, the reinforcement scheme used to update the probability vectors, L_{R-I}, ensures that at each stage k, the probability of constructing the MIS ω_j (denoted as $q_i(k)$) remains unchanged with probability $c_j(k)$ for all $j = 1, 2, \ldots, r$, when the constructed MIS ω_j is penalized by the random environment. Conversely, when the constructed MIS ω_j is rewarded, the probability of selecting the vertices of ω_j that are included in the constructed MIS increase by a specified learning rate, while the probability for other vertices decreases. To illustrate the proof of the lemma in greater detail, we will examine the maximum independent set shown in Figure 3. In this figure, the multilayer graph G consists of 6 nodes and 10 edges distributed across 2 layers. This graph includes 7 MISs: : $\omega_1 = \{2, 4, 5\}$, $\omega_2 = \{1, 5\}$, $\omega_3 = \{2, 6\}$, $\omega_4 = \{2, 4\}$, $\omega_5 = \{2, 5\}$, $\omega_6 = \{3, 4\}$ and $\omega_7 = \{1, 3\}$. It is assumed that ω_7 is the MIS with the maximum expected weight in the graph depicted in Figure 3.



(a) **(b)**

Figure 3: Sample Multilayer Graph G and its maximum independent set.

Let $q_i(k) = \prod_{v_m \notin \omega_i} p_0^m(k) \prod_{v_m \in \omega_i} p_1^m(k)$ indicates as the probability of constructing MIS $\omega_i$at stage k, where p_0^m and p_1^m denote the probability, which node v_m declares itself is not a member of a multilayer independent set (or chooses action α_m^0) and as a member of an independent set (or choosing action α_m^1), respectively. Hence, for the graph shown in figure 3, we have:

$$\begin{aligned}
q_1(k) &= p_0^1(k)p_1^2(k)p_0^3(k)p_1^4(k)p_1^5(k)p_0^6(k),\\
q_2(k) &= p_1^1(k)p_0^2(k)p_0^3(k)p_0^4(k)p_1^5(k)p_0^6(k),\\
q_3(k) &= p_0^1(k)p_1^2(k)p_0^3(k)p_0^4(k)p_0^5(k)p_1^6(k),\\
q_4(k) &= p_0^1(k)p_1^2(k)p_0^3(k)p_1^4(k)p_0^5(k)p_0^6(k),\\
q_5(k) &= p_0^1(k)p_1^2(k)p_0^3(k)p_0^4(k)p_1^5(k)p_0^6(k),\\
q_6(k) &= p_0^1(k)p_0^2(k)p_1^3(k)p_1^4(k)p_0^5(k)p_0^6(k),\\
q_7(k) &= p_1^1(k)p_0^2(k)p_1^3(k)p_0^4(k)p_0^5(k)p_0^6(k).
\end{aligned} \qquad (10)$$

The conditional expectation of $q_1(k+1)$, assuming $\underline{q}(k)$ evolves according to algorithm 1, is defined as follows [50]:

$$
\begin{aligned}
&E[q_1(k+1)|q_1(k)]\\
&\qquad = q_1(k)[c_1 q_1(k)\\
&\qquad + \Big\{d_1(k)\big[p_0^1(k) + a\big(1 - p_0^1(k)\big)\big\}\{p_1^2(k) + a\big(1 - p_1^2(k)\big)\}\{p_0^3(k)\\
&\qquad + a\big(1 - p_0^3(k)\big)\}\{p_1^4(k) + a\big(1 - p_1^4(k)\big)\}\Big\{p_1^5(k) + a\Big(1 - p_1^5(k)\Big)\Big\}\{p_0^6(k)\\
&\qquad + a\big(1 - p_0^6(k)\big)\}\Big] +\\
&q_2(k)[c_2 q_1(k) + \Big\{d_2(k)\big[p_0^1(k) + a\big(1 - p_0^1(k)\big)\big\}\{p_0^2(k)(1-a)\}\{p_0^3(k)\\
&\qquad + a\big(1 - p_0^3(k)\big)\}\{p_0^4(k)(1-a)\}\Big\{\Big\{p_1^5(k) + a\Big(1 - p_1^5(k)\Big)\Big\}\Big\}\{p_0^6(k)\\
&\qquad + a\big(1 - p_0^6(k)\big)\}\Big] +\\
&q_3(k)[c_3 q_1(k) + \{d_3(k)\{p_0^1(k)\\
&\qquad + a\big(1 - p_0^1(k)\big)\}\{p_1^2(k) + a(1 - p_1^2(k))\}\{p_0^3(k) + a\big(1 - p_0^3(k)\big)\}\{p_0^4(k)(1\\
&\qquad - a)\}\{p_0^5(k)(1-a)\}\{p_0^6(k) + a\big(1 - p_0^6(k)\big)\}] +\\
&q_4(k)[c_4 q_1(k) + \Big\{d_4(k)\big[p_0^1(k) + a\big(1 - p_0^1(k)\big)\big\}\{p_1^2(k) + a\big(1 - p_1^2(k)\big)\}\{p_0^3(k)\\
&\qquad + a\big(1 - p_0^3(k)\big)\}\{p_1^4(k) + a\big(1 - p_1^4(k)\big)\}\Big\{\{p_0^5(k)(1-a)\}\Big\}\{p_0^6(k)\\
&\qquad + a\big(1 - p_0^6(k)\big)\}\Big] +\\
&q_5(k)[c_5 q_1(k) + \Big\{d_5(k)\big[p_0^1(k) + a\big(1 - p_0^1(k)\big)\big\}\{p_1^2(k) + a\big(1 - p_1^2(k)\big)\}\{p_0^3(k)\\
&\qquad + a\big(1 - p_0^3(k)\big)\}\{p_0^4(k)(1-a)\}\Big\{\Big\{p_1^5(k) + a\Big(1 - p_1^5(k)\Big)\Big\}\Big\}\{p_0^6(k)\\
&\qquad + a\big(1 - p_0^6(k)\big)\}\Big] +\\
&q_6(k)[c_6 q_1(k) + \Big\{d_6(k)\big[p_0^1(k) + a\big(1 - p_0^1(k)\big)\big\}\{p_1^2(k)(1-a)\}\{p_0^3(k)(1-a)\}\{p_1^4(k)\\
&\qquad + a\big(1 - p_1^4(k)\big)\}\Big\{\{p_0^5(k)(1-a)\}\Big\}\{p_0^6(k) + a\big(1 - p_0^6(k)\big)\}\Big] +\\
&q_7(k)[c_7 q_1(k) + \Big\{d_7(k)\big[p_0^1(k) + a\big(1 - p_0^1(k)\big)\big\}\{p_0^2(k)(1-a)\}\{p_0^3(k)(1-a)\}\{p_0^4(k)(1\\
&\qquad - a)\}\Big\{\{p_0^5(k)(1-a)\}\Big\}\{p_0^6(k) + a\big(1 - p_0^6(k)\big)\}\Big]
\end{aligned}
$$

By simplifying all the terms on the right side of the equation above and conducting some algebraic manipulations, we arrive at:

$$E[q_1(k+1)|q_1(k)] = \sum_{j=1}^{7} q_j(k)\left[c_j(k)q_1(k) + d_j(k)\left(\prod_{v_m \notin \omega_1} p_0^m(k) \prod_{v_m \in \omega_1} p_1^m(k)\right)\right],$$

where

$$p_0^m(k+1) = \begin{cases} p_0^m(k) + a(1 - p_0^m(k)) & v_m \notin \omega_j \\ p_0^m(k)(1-a) & v_m \in \omega_j \end{cases}$$

$$p_1^m(k+1) = \begin{cases} p_1^m(k) + a(1 - p_1^m(k)) & v_m \in \omega_j \\ p_1^m(k)(1-a) & v_m \notin \omega_j \end{cases}.$$

In this equation, $p_1^m(k)$ indicates the probability that vertex v_m selects itself as a member of the independent set, at iteration k. As mentioned, each learning automata consists of two possible actions, the probability with which vertex v_m selects itself as does not a member of the independent set is equal to $1 - p_1^m(k)$. Hence, q_i at stage k can be defined as [50]:

$$\prod_{v_m \notin \omega_i} (1 - p_1^m(k)) \prod_{v_m \in \omega_i} p_1^m(k)$$

and so we have

$$E(q_1(k+1)|q(k)] = \sum_{j=1}^{7} q_j(k) \left[c_j(k) q_1(k) + d_j(k) \left(\prod_{v_m \notin \omega_1} (1 - p_1^m(k)) \prod_{v_m \in \omega_1} p_1^m(k)\right)\right],$$

where

$$p_1^m(k+1) = \begin{cases} p_1^m(k) + a(1 - p_1^m(k)) & v_m \in \omega_j \\ p_1^m(k)(1-a) & v_m \notin \omega_j \end{cases},$$

and hence the proof of the lemma.

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

Proof: Define $\Delta q_i(k) = E[q_i(k+1)|q(k)] - q_i(k)$. From lemma 2, we have

$$\Delta q_i(k) = \sum_{j=1}^{r} q_j(k) \left[c_j(k) q_i(k) + d_j(k) \left(\prod_{v_m \notin \omega_i} (1 - p_1^m(k)) \prod_{v_m \in \omega_i} p_1^m(k)\right)\right] - q_i(k), \qquad (11)$$

where

$$p_1^m(k+1) = \begin{cases} p_1^m(k) + a(1 - p_1^m(k)) & v_m \in \omega_j \\ p_1^m(k)(1-a) & v_m \notin \omega_j \end{cases}. \qquad (12)$$

For notational convenience, let

$$\prod_{n \in \{0,1\}}^{\omega_j} \pi_n^m(k) = \prod_{v_m \notin \{0,1\}} (1 - p_1^m(k)) \prod_{v_m \in \{0,1\}} p_1^m(k), \qquad (13)$$

therefore, we have

$$\Delta q_i(k) = \sum_{j=1}^{7} q_j\,(k)\left[c_j(k)q_i(k) + d_j(k)\prod_{n\in\{0,1\}}^{\omega_i} \pi_n^m(k)\right] - q_i(k), \tag{14}$$

where

$$\pi_n^m(k+1) = \begin{cases} p_n^m(k) + a(1-p_n^m(k)) & (v_m \notin \omega_j, n=0) or\ (v_m \in \omega_j, n=1) \\ p_n^m(k)(1-a) & (v_m \notin \omega_j, n=1) or\ (v_m \in \omega_j, n=0) \end{cases}, \tag{15}$$

where π_n^m indicates as the probability with which vertex v_m, at stage k, declares itself as a member of the independent set vertex when $n = 1$ and as doesn't member of the independent set vertex otherwise.
The probability that the maximum independent set (MIS) is formed, rewarded, or penalized is determined by the product of the probabilities of identifying the vertices in the constructed MIS as members of the independent set, as well as those not included in the independent set, as outlined in equation (10), we have

$$\Delta q_i(k) = \sum_{j=1}^{r} \prod_{n\in\{0,1\}}^{\omega_j} p_n^m(k)\left[\prod_{n\in\{0,1\}}^{\omega_j} c_n^m(k)\prod_{n\in\{0,1\}}^{\omega_i} p_n^m(k) + \prod_{n\in\{0,1\}}^{\omega_j} d_n^m(k)\prod_{n\in\{0,1\}}^{\omega_i} \pi_n^m(k)\right] - \prod_{n\in\{0,1\}}^{\omega_i} p_n^m(k)\ , \tag{16}$$

where $\pi_n^m(k)$ is defined as in equation (11), $c_n^m(k)$ denotes the probability of penalizing the action selected by vertex v_m at instant k, and $d_n^m(k) = 1 - c_n^m(k)$.

$$\Delta q_i(k) = \prod_{n\in\{0,1\}}^{\omega_j} E[p_n^m(k+1)|p_n^m(k)] - \prod_{n\in\{0,1\}}^{\omega_i} p_n^m(k)\ , \tag{17}$$

The above quality can be rewritten as

$$\Delta q_i(k) \geq \prod_{n\in\{0,1\}}^{\omega_j} E[p_n^m(k+1)|p_n^m(k)] - p_n^m(k) = \prod_{n\in\{0,1\}}^{\omega_i} \Delta p_n^m(k)\ , \tag{18}$$

where

$$\Delta p_n^m(k) = a p_n^m(k)\sum_{s\neq n} p_s^m(k)[c_s^m(k) - c_n^m(k)] \quad \substack{\omega_i \\ n\,\in\,\{0,1\}}\ , \tag{19}$$

and $q_i(k) \in (0,1)$ {for all $\underline{q} \in S_r^0$}. S_r^0 is defined as the interior of S_r, where $S_r = \{\underline{q}(k)|1 \leq q_i(k) \leq 1, \sum_{i=1}^{r} q_i(k) = 1\}$. Hence, $p_n^m(n) \in (0,1)$ for all m,n. Since action α_n^m is the action with the minimum penalty probability which can be selected by automaton A_m, it is shown that $c_s^{m^*} - c_n^{m^*} > 0$, for all $s \neq n$, where $c_s^{m^*}$ is the final value to which the penalty probability $c_s^m(k)$ is converged. It follows from lemma 1 that for a large value of k, $c_s^m(k) - c_n^m(k) > 0$. Therefore, we conclude that for large values of k, the right-hand side of the aforementioned equation comprises nonnegative quantities [47]. Consequently, we have:

$$\frac{\prod_{n\in\{0,1\}}^{\omega_i} ap_n^m(k)\sum_{s\neq n} p_s^m(k)[c_s^m(k)-c_n^m(k)]}{\omega_i \atop n\in\{0,1\}} \geq 0, \tag{20}$$

and from equation (12), we have

$$\Delta q_i(k) \geq \frac{\prod_{n\in\{0,1\}}^{\omega_i} ap_n^m(k)\sum_{s\neq n} p_s^m(k)[c_s^m(k)-c_n^m(k)]}{\omega_i \atop n\in\{0,\,1\}}, \tag{21}$$

which completes the proof of this lemma 3.

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

Proof: Lemma 3 implicitly demonstrates that $\{\underline{q}(k)\}$ is a sub-martingale. By applying the martingale convergence theorem and considering that $\{\underline{q}(k)\}$ is non-negative and uniformly bounded, it is concluded that $\lim_{k\to\infty} q_i(k) = q^*$ exists with probability one. Therefore, from equation (11), it is evident that $q_i(k+1) \neq q_i(k)$ with nonzero probability if and only if $q_i(k) \notin \{0,1\}$, and $\underline{q}(k+1) = \underline{q}(k)$ with probability one if and only if $q^* \in \{0,1\}$, where $\lim_{k\to\infty} q_i(k) = q^*$. Hence, the proof is completed.■

Let $\Gamma_i(q)$ be defined as the probability that the proposed Algorithm 1 converges to the unit vector e_i, given the initial probability vector$\underline{q}$. Specifically, $\Gamma_i(q)$ is defined as $\Gamma_i(q) = prob\left[q_i(\infty) = 1 \middle| \underline{q}(0) = \underline{q}\right] = prob[q^* = e_i | \underline{q}(0) = \underline{q}]$.

Let $C(S_r): S_r \to \mathcal{R}$ denotes the state space of all real-valued continuously differentiable functions with bounded derivatives, defined on S_r, where $\mathcal{R}$ represents the real line. If $\psi(.) \in C(S_r)$, the operator U is defined by:

$$U\psi(q) = E[\psi q(k+1) | q(1) = q], \tag{22}$$

where $E[.]$ represents the mathematical expectation. It has been demonstrated in [47] that the operator U is linear and preserves non-negative functions, as the expectation of a non-negative function remains non-negative. In other words, $U\psi(q) \geq 0$ for all \ $q \in S_r$ if $\psi(q) \geq 0$. If the operator U is applied repeatedly n times (for all $n > 1$), we have:

$$U^{n-1}\psi(q) = E[\psi q(k+1) | q(1) = q] \tag{23}$$

A function $\psi(q)$ is termed 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 [47]. that $\Gamma_i(q)$ is the unique continuous solution to the equation $U\Gamma_i(q) = \Gamma_i(q)$, subject to specific boundary conditions.

$$\begin{aligned} &\Gamma_i(e_i) = 1 \\ &\Gamma_i\left(e_j\right) = 0, j \neq i \end{aligned} \tag{24}$$

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 super-regular with $\psi_i(e_i) = 1$ and $\psi_i\left(e_j\right) = 1$ for $j \neq i$ then

$$\psi_i(q) \geq \Gamma_i(q), \tag{25}$$

For all $q \in S_r$. If $\psi_i(.) \in C(S_r)$ is sub-regular with the same boundary conditions, then
$\psi_i(q) \leq \Gamma_i(q) \qquad \forall q \in S_r$.

Proof. The proof of Theorem 2 can be found in [32].■

In what follows, we show that $\phi_i[x,q]$ is a sub-regular function. Therefore, $\phi_i[x, q]$ qualifies as a lower bound on $\Gamma_i(q)$. Since super and sub-regular functions are closed under addition and multiplications by a positive constant, and if $\phi(.)$ is super regular then $-\phi(.)$ is sub-regular, it follows that $\phi_i[x, q]$ is sub-regular if and only if

$$\theta_i[x, q] = e^{\frac{-xq_i}{a}} \tag{26}$$

is super-regular. We now determine the conditions under which $\theta_i[x, q]$ is super-regular. From the definition of operator U given in equations 13 and 11 we have

$$\begin{aligned} U\theta_i[x, q] &= E\left[e^{\frac{-xq_i(k+1)}{a}}\middle| q(k) = q\right] \\ &= \sum_{j=1}^{r} q_j\, d_j^* e^{-\frac{x}{a}\left[\prod_{\substack{v_m \notin \omega_j, n=0 \\ or \\ v_m \in \omega_j, n=1}} \left(p_m^n + a(1-p_m^n)\right)\right]} \\ &+ \sum_{j=1}^{r} q_j\, d_j^* e^{-\frac{x}{a}\left[\prod_{\substack{v_m \notin \omega_j, n=1 \\ or \\ v_m \in \omega_j, n=0}} \left(p_m^n (1-a)\right)\right]} \end{aligned} \tag{27}$$

$$= [q_i d_i^* e^{-\frac{x}{a}\left(q_i + a(1-q_i)\right)} + \sum_{j \neq i} q_i d_i^* e^{-\frac{x}{a}\left[\prod_{\substack{v_m \notin \omega_j, n=0 \\ or \\ v_m \in \omega_j, n=1}} \left(p_m^n + a(1-p_m^n)\right)\right]} + \sum_{j=1}^{r} q_j \, d_j^* e^{-\frac{x}{a}\left[\prod_{\substack{v_m \notin \omega_j, n=1 \\ or \\ v_m \in \omega_j, n=0}} \left(p_m^n (1-a)\right)\right]}$$

In the context of large values of k, we denote d_j^* as the final value to which the reward probability $d_j(k)$ converges. Furthermore, $e^{-\frac{x}{a}(q_i + a(1-q_i))}$ represents the expected value of $\theta_i(x,q)$when rewards are applied to the maximum stochastic multilayer independent set ω_i.

$$U\theta_i(x,q) = [q_i d_i^* e^{-\frac{x}{a}\left(q_i + a(1-q_i)\right)} + \sum_{j \neq i} q_i d_i^* e^{-\frac{x}{a}\left(q_i(1-q_i) \prod_{\substack{v_m \notin \omega_j, n=0 \\ or \\ v_m \in \omega_j, n=1}} \frac{(p_m^n + a(1-p_m^n))}{(p_m^n(1-a))} \right)}] \tag{28}$$

$$= [\sum_{j=i} q_i d_i^* e^{-\frac{x}{a}\rho_i^j\left(q_i + a(1-q_i)\right)} + \sum_{j \neq i} q_i d_i^* e^{-\frac{x}{a}\left(\rho_i^j q_i (1-q_i)\right)} \tag{29}$$

where $\rho_i^j > 0$ and is defined as

$$\rho_i^j = \begin{cases} \prod_{\substack{v_m \notin \omega_j, n=0 \\ or \\ v_m \in \omega_j, n=1}} \dfrac{(p_m^n + a(1-p_m^n))}{(p_m^n(1-a))} & i \neq j \\ 1 & i = j \; or \; (\omega_i \cap \omega_j) = \phi \end{cases} \tag{30}$$

$$U\theta_i(x,q) - \theta_i(x,q) = \left[e^{-\frac{x q_i \rho_i^j}{a}} \sum_{j=i} q_j d_j^* \, e^{-x(1-q_i)\rho_i^j} + e^{-\frac{x q_i \rho_i^j}{a}} \sum_{j \neq i} q_j d_j^* \, e^{x q_i \rho_i^j} \right] - e^{-\frac{x q_i}{a}} \tag{31}$$

$\theta_i(x,q)$ is super-regular if

$$e^{-\frac{x q_i \rho_i^j}{a}} \sum_{j=i} q_j d_j^* \, e^{-x(1-q_i)\rho_i^j} + e^{-\frac{x q_i \rho_i^j}{a}} \sum_{j \neq i} q_j d_j^* \, e^{x q_i \rho_i^j} \leq e^{-\frac{x q_i}{a}} \tag{32}$$

And

$$U\theta_i(x,q) \leq e^{-\frac{x q_i}{a}} q_i d_i^* e^{-x(1-q_i)} + e^{-\frac{x q_i}{a}} \sum_{j \neq i} q_j d_j^* \, e^{x q_i} \tag{33}$$

If $\theta_i(x, q)$ is super-regular, so we have

$$U\theta_i(x,q) - \theta_i(x,q) \le \left[e^{-\frac{xq_i}{a}} q_i d_i^* e^{-x(1-q_i)} + e^{-\frac{xq_i}{a}} \textstyle\sum_{j\neq i} q_j d_j^* \, e^{xq_i}\right] - e^{-\frac{xq_j}{a}}, \tag{34}$$

by multiplying and dividing the right side of the above inequality by $-xq_i$ and simplifying the resulting algebraic expressions, we obtain:

$$\begin{aligned} U\theta_i(x,q) - \theta_i(x,q) &\le -xq_i e^{\frac{-xq_i}{a}} \left[q_i d_i^* \frac{e^{-x(1-q_i)} - 1}{-xq_i} - \sum_{j\neq i} q_i d_i^* \frac{e^{xq_i} - 1}{xq_i} \right] \\ &= -xq_i e^{\frac{-xq_i}{a}} \left[d_i^* \frac{e^{-x(1-q_i)} - 1}{-x} - \sum_{j\neq i} q_i d_i^* \frac{e^{xq_i} - 1}{xq_i} \right] \\ &= -xq_i e^{\frac{-xq_i}{a}} \left[(1-q_i) d_i^* \frac{e^{-x(1-q_i)} - 1}{-x(1-q_i)} - \sum_{j\neq i} q_i d_i^* \frac{e^{xq_i} - 1}{xq_i} \right] \end{aligned} \tag{35}$$

and

$$V[u] = \begin{cases} \frac{e^u - 1}{u} & u \neq 0 \\ 1 & u = 0 \end{cases} \tag{36}$$

$$\begin{aligned} \theta_i(x,q) - \theta_i(x,q) &\le -xq_i e^{\frac{-xq_i}{a}} \left[(1-q_i) d_i^* V[-x(1-q_i)] - (\sum_{j\neq i} q_i d_j^*) V[xq_i] \right] \\ &= xq_i \theta_i(x,q) G_i(x,q) \end{aligned} \tag{37}$$

where $G_i(x,q)$ is defined as

$$G_i(x,q) = (1-q_i) d_i^* V[-x(1-q_i)] - \left(\sum_{j\neq i} q_j d_j^* \right) V[xq_i] \tag{38}$$

therefore, $\theta_i(x,q)$ is super-regular if

$$G_i(x,q) \ge 0 \tag{39}$$

for all $q \in S_r$, from equation (16), it follows that $\theta_i(x, q)$ is super regular if we have

$$f_i(x,q) = \frac{V[-x(1-q_i)]}{V[xq_i]} \le \frac{\sum_{j\neq i} q_j d_j^*}{(1-q_i) d_i^*}, \tag{40}$$

the right side of the inequality (18) contains only non-negative terms, therefore

$$\left(\sum_{j\neq i} q_i\right)\underset{j\neq i}{\min}\left(\frac{d_j^*}{d_i^*}\right) \leq \frac{1}{(1-q_i)}\sum_{j\neq i} q_j\frac{d_j^*}{d_i^*} \leq \left(\sum_{j\neq i} q_i\right)\underset{j\neq i}{\max}\left(\frac{d_j^*}{d_i^*}\right) \tag{41}$$

by replacing $\sum_{j\neq i} q_j$ with (1-q_i) in the previous inequality, we can rephrase it as

$$\underset{j\neq i}{\min}\left(\frac{d_j^*}{d_i^*}\right) \leq \sum_{j\neq i} q_j\frac{d_j^*}{d_i^*} \leq \underset{j\neq i}{\max}\left(\frac{d_j^*}{d_i^*}\right) \tag{42}$$

according to equation (18), we can deduce that $\theta_i(x,q)$ is super-regular if

$$f_i(x,q) \geq \underset{j\neq i}{\max}\left(\frac{d_j^*}{d_i^*}\right)(x,q) \tag{43}$$

for further simplification, let's employ logarithms. Let

$$\Delta(q,x) = Ln\, f_i(x,q) \tag{44}$$

It has also been shown in [51], that

$$-\int_0^x H'(u)du \leq \Delta(q,x) \leq -\int_{-x}^0 H'(u)du \tag{45}$$

where $(u) = \frac{dH(u)}{du}$, $H(u) = \ln V(u)$. We have

$$\frac{1}{V[x]} \leq \frac{V[-x(1-q_i)]}{V[xq_i]} \leq V[-x] \tag{46}$$

and

$$\frac{1}{V[x]} = \underset{j\neq i}{\max}\left(\frac{d_j^*}{d_i^*}\right) \tag{47}$$

Let's denote the value of x that satisfies equation (19) as x^*. It has been demonstrated that there exists a positive x value that satisfies equation (19) when $(\frac{d_j}{d_i})$ is less than 1 for all $j\neq i$. By selecting $x = x^*$, equation (19) remains true. As a result, equation (17) is also true, and $\theta_i(x,q)$ is a super regular function. Thus,

$$\varphi_i[x,q] = \frac{e^{\frac{-xq_i}{a}}-1}{e^{\frac{-x}{a}}-1}, \tag{48}$$

considering a function that meets the boundary conditions (14) and is sub-regular, along with the information from theorem 2 and inequality, we can reach the following conclusion:

$$\varphi_i[x,q] \leq \Gamma_i(q) \leq 1 \tag{49}$$

By observing the definition of $\varphi_i[x, q]$, we can notice that for any given $\varepsilon > 0$, there exists a positive constant $a^* < 1$. This means that for all $0 < a \leq a^*$ the following conditions are satisfied

$$1 - \varepsilon \leq \phi_i[x, q] \leq \Gamma_i(q) \leq 1 \tag{50}$$

So, the conclusion is that as k (the number of iterations) approaches infinity, the probability of Algorithm 1 converging to MIS with the maximum expected weight is equal to 1.

3.8 Time and space complexity

The time complexity of the proposed algorithm can be analyzed based on the number of layers M, the number of nodes in each layer N, and the maximum number of iterations K_{max} , which is treated as a constant. In each round of execution, all $(M \times N)$ nodes operate in parallel, where each node selects one of two possible actions; consequently, the action selection phase incurs a time complexity of $O(2 \times M \times N)$. However, since this selection occurs concurrently across all nodes, the effective time for the action selection phase can be considered $O(1)$. The algorithm continues this process for a maximum of K_{max} iterations, resulting in an overall time complexity of $O(2 \times K_{max})$. Therefore, the complete time complexity of the algorithm can be succinctly expressed as $O(2 \times K_{max})$, indicating that the computational demand is primarily driven by the number of iterations, while the parallel execution across nodes ensures efficiency. The space complexity of the proposed algorithm can be evaluated based on the number of layers M, the number of nodes in each layer N, and the storage requirements for each node. Each node maintains two vectors to store the action probability values associated with its possible actions. Consequently, for M layers and N nodes per layer, the total space required for storing these vectors is $O(2 \times M \times N)$. This linear space complexity indicates that the memory usage of the algorithm scales proportionally with the product of the number of layers and the number of nodes in each layer. As a result, the algorithm is efficient in terms of space utilization, making it suitable for applications involving large multilayer graphs where memory resources may be a concern.

4 Experiments

In this section, the evaluation of the proposed algorithms' performance was accompanied by a series of experiments on synthetic stochastic multilayer graphs. The objective of this section is to assess the performance of the proposed algorithm through analysis using diverse stochastic multilayer graphs. Moreover, the evaluation of the proposed algorithm is performed by considering its performance across two distinct types of multilayer graphs: graphs with pre-specified topology, and real multilayer graphs.

- The initial three graphs are established stochastic multilayer graphs, derived from the work of Hutson and Shier [52]. These graphs are characterized in table 1 and are shown in figure 4, figure 5, and Figure 6, the vertex weights are represented as random variables. Besides, the weights and their probabilities on each edge are provided in Tables A1, Table A2, and Table A3 in the appendix.

Table 1: Stochastic Multilayer Graph Applied in Simulations

Test Graph	#Nodes	# Edge	# Layer	Parameter of distribution	Size Stochastic Multilayer maximum Independent Set	Expected Weight of Maximum Independent Set
Graph 1	8	14	2	-	3	32.24
Graph 2	9	15	2	-	3	41.91
Graph 3	9	21	2	-	4	49.37

Table 1 illustrates the characteristics and outcomes of simulations conducted on three stochastic multilayer graphs, each varying in complexity and connectivity. Graph 1 with 8 nodes and 14 edges, and Graph 2 with 9 nodes and 15 edges, have relatively simple structures compared to Graph 3, which contains 9 nodes but a higher count of 21 edges, indicating a denser and possibly more complex network topology. All three graphs consist of two layers, suggesting dual modes of interactions among the nodes, potentially representing different social networks or communication layers. The analysis reveals that while the size of the maximum independent set remains stable at three for the first two graphs, it increases to four in Graph 3, aligning with the graph's increased edge density and complexity. This is reflected in the growing expected weights of the maximum independent sets, measuring 32.24, 41.91, and 49.37 respectively, which highlights a trend where more interconnected graphs tend to yield more valuable independent sets. These findings underscore the critical role of network architecture in determining the effectiveness of algorithms aimed at optimizing network functionalities such as resource distribution, traffic management, or information dissemination within complex, multi-layered network environments. Visualizations of Graphs 1 through 3 are depicted in Figures 6 to 6.

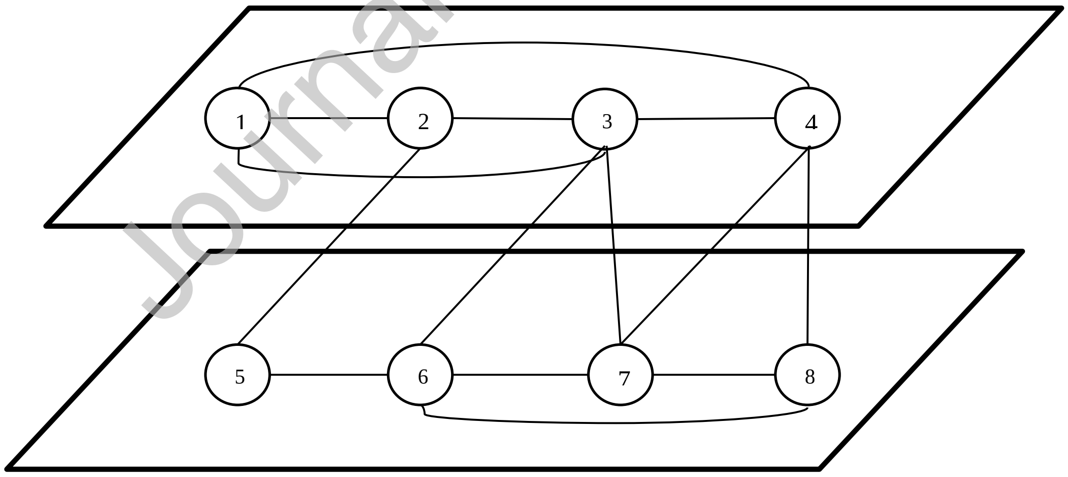


Figure 4: Stochastic Multilayer Graph "Graph 1" with predefined structure used for experiments.

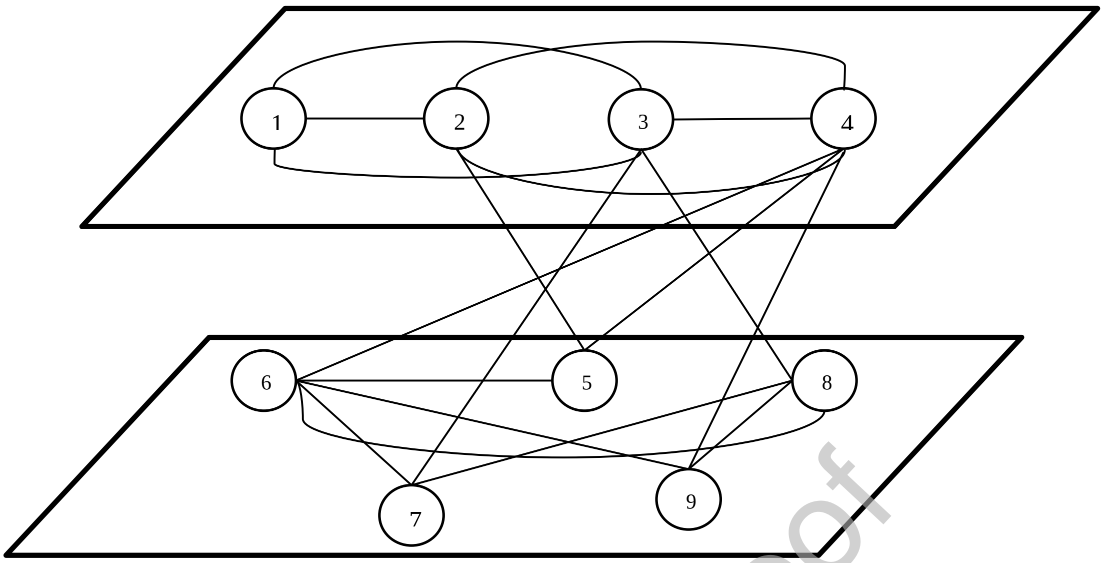


Figure 5: Stochastic Multilayer Graph "Graph 2" with predefined structure used for experiments.

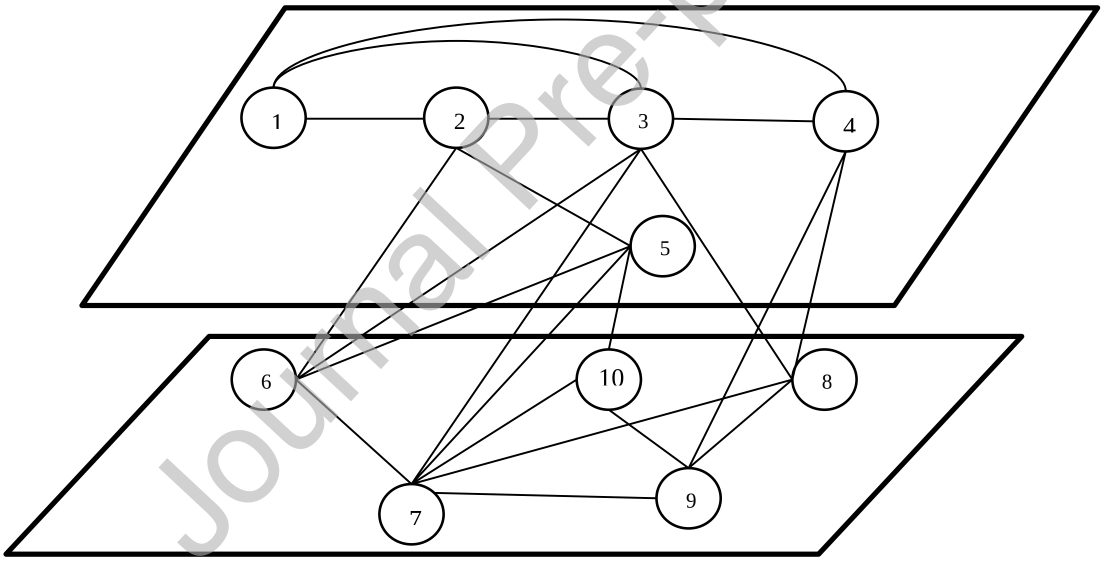


Figure 6: Stochastic Multilayer Graph "Graph 3 "with predefined structure used for experiments.

- We include real multilayer graphs as the third category of graphs. These networks are characterized in Table 2 and serve as the focal point of our study. Each node in these networks is assigned a unique label, which corresponds to the mean value extracted from the length distribution of the respective edge. It is noteworthy that the length distribution follows an exponential pattern.

Table 2: Real Multilayer Graph Applied in Simulations

Real Network	#Nodes	# Edge	# Layers
Florentine	16	35	2
Monastery	18	510	10
Bank Wiring	14	110	6
Tailor Shop	39	552	4

Table 2 presents data from simulations conducted on real multilayer graphs from various social environments, each exhibiting unique structural characteristics. The Florentine network, with 16 nodes and 35 edges across 2 layers, suggests moderate complexity and connectivity. In stark contrast, the Monastery graph demonstrates a high degree of complexity with 18 nodes interconnected by 510 edges spread across 10 layers, indicating an intricate multi-tiered social structure. The Bank Wiring network shows a denser connectivity relative to its size with 14 nodes and 110 edges over 6 layers, reflecting potentially strong interdependencies among units within a financial setting. Finally, the Tailor Shop network, the largest sampled, consists of 39 nodes and 552 edges across 4 layers, pointing towards a highly interactive and layered organizational structure. These networks, varying significantly in their topology and layering, serve as valuable models for studying complex behaviors and interactions within and across distinct social or organizational layers, potentially guiding targeted interventions or optimizations in these real-world systems.

In all of the experiments presented in this paper, we utilize the L_{R-I} learning scheme with a learning rate of 0.02. To ensure optimal performance, we set the entropy threshold Ent at 0.1. Additionally, the maximum iteration k is set to $n \times 1000$, where n represents the number of nodes within the stochastic multilayer graph.

4.1 Performance evaluations

To assess various algorithms, we utilized performance metrics for trees. The initial metric is the Average Number of Iterations (ANI), which reflects the average iterations needed for the algorithm to identify the maximum independent set in stochastic multilayer graphs. If the algorithm does not successfully locate the maximum independent set, the total iterations taken until termination are considered for calculating the average. A lower ANI value signifies superior performance.
The second performance metric is the percentage of runs converged to a stochastic multilayer independent set (PCR). Due to the randomness nature of both the proposed algorithm and the input stochastic multilayer graphs, it is necessary to run the algorithm multiple times to observe its behavior. This performance measure indicates the percentage of runs that successfully find the maximum independent set. A higher PCR value indicates better performance.
The third performance metric is the average number of samples (TS) taken from the nodes of the stochastic multilayer graph. This performance metric indicates the average number of samples taken from the stochastic multilayer graph's nodes to find the independent set with maximum cardinality. A smaller value for TS is preferable.

4.2 Experiments

In this subsection the report of the serval experimental simulation are provided.

4.2.1 Experiment I

This experiment which is presented in Figure 7, Figure 8, and Figure 9 illustrate the performance of five different algorithms in achieving the maximum Stochastic Multilayer Independent Set (SMIS) across various graphs. Each graph depicts the average probability of achieving the SMIS versus the number of iterations, providing insights into the convergence behavior and efficiency of each algorithm. From the obtained results several conclusions can be made as the following:

- For Graph 1, the initial observations show that all algorithms start with a low probability, which rapidly increases with the number of iterations. By around 400 iterations, Algorithms 1, 2, 4, and 5 reach a high probability of achieving the maximum SMIS. Algorithm 3 lags slightly but still converges towards a high probability. Algorithm 1 and Algorithm 2 demonstrate the quickest convergence, achieving high probabilities the fastest. Algorithm 4 and Algorithm 5 also show strong performance, albeit with a slightly slower rate of convergence. Algorithm 3 converges at a slower rate compared to the others but eventually reaches a similar high probability.
- For the performance on Graph 2, the initial observations are similar to those for Figure 6, with all algorithms starting with a low probability and increasing rapidly. From a convergence perspective, the convergence trends are more distinct in this figure. By around 600 iterations, Algorithm 1 reaches a probability of 1, indicating it consistently finds the maximum SMIS. Algorithm 2 also performs very well, reaching a probability close to 1 slightly later. Algorithms 3, 4, and 5 converge more slowly, with Algorithm 3 reaching around 0.95 and Algorithms 4 and 5 slightly below that. Algorithm 1 significantly outperforms the others in terms of both speed and final probability. Algorithm 2 follows closely behind, and Algorithm 3 shows moderate performance. Algorithms 4 and 5 have the slowest convergence rates and slightly lower final probabilities.
- For the performance on Graph 3, the initial trends are consistent with Figures 6 and 7, with all algorithms showing rapid probability increases early on. By around 700 iterations, Algorithm 1 again demonstrates superior performance, reaching a probability of 1. Algorithm 2 also performs well, with a final probability close to 1. Algorithm 3 reaches a final probability of around 0.95, and Algorithms 4 and 5 have slower convergence rates and lower final probabilities compared to Algorithms 1 and 2. From a performance perspective, Algorithm 1 is the fastest and most reliable in achieving the maximum SMIS, and Algorithm 2 shows strong performance, though slightly behind Algorithm 1. Algorithm 3 performs moderately, while Algorithms 4 and 5 exhibit the slowest convergence and lower final probabilities.

- Algorithm 1 consistently outperforms the other algorithms across all three graphs, achieving the maximum SMIS the fastest and with the highest probability. Moreover, Algorithm 2 is also highly effective, often performing nearly as well as Algorithm 1. In addition, Algorithm 3 shows moderate performance, with slower convergence but still reaching a high probability, while Algorithms 4 and 5 lag behind the others, with slower convergence rates and lower final probabilities.

The experimental results indicate that Algorithm 1 is the most efficient and reliable for achieving the maximum SMIS across different graph structures. Algorithm 2 is also a strong performer, while Algorithms 3, 4, and 5, although eventually achieving high probabilities, do so more slowly and less consistently. This analysis highlights the importance of selecting the right algorithm for optimizing performance in probabilistic graph problems.

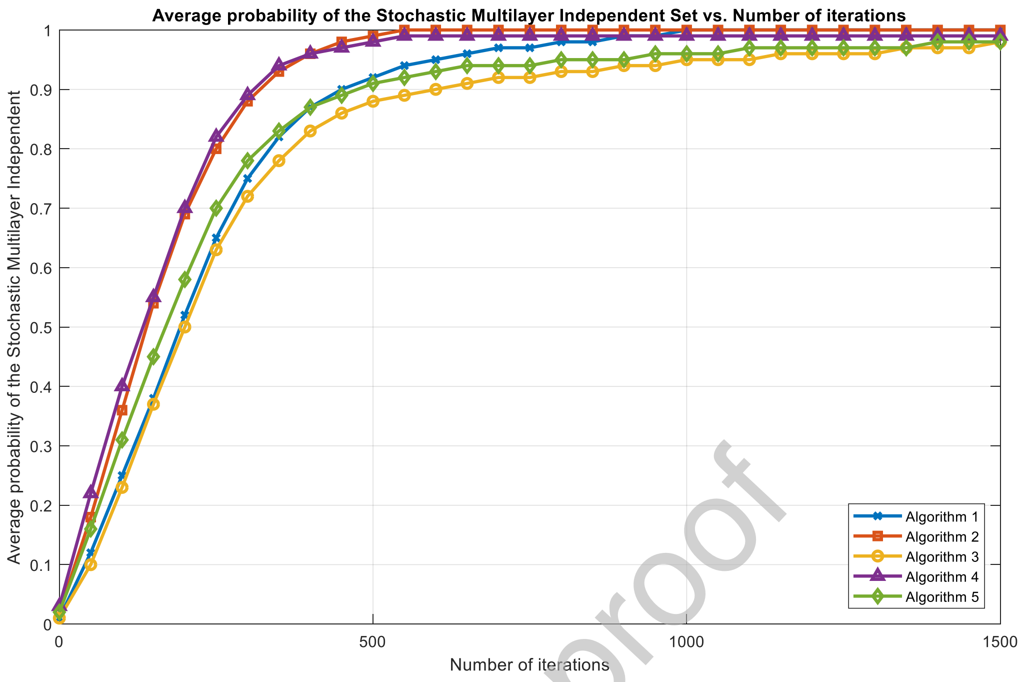


Figure 7: The probability of achieving the maximum SMIS (Stochastic Maximum Independent Set) for graph 1

The result in Figure 7 for Graph 1, the initial observations show that all algorithms start with a low probability, which rapidly increases with the number of iterations. By around 400 iterations, Algorithms 1, 2, 4, and 5 reach a high probability of achieving the maximum SMIS. Algorithm 3 lags slightly but still converges towards a high probability. Algorithm 1 and Algorithm 2 demonstrate the quickest convergence, achieving high probabilities the fastest. Algorithm 4 and Algorithm 5 also show strong performance, albeit with a slightly slower rate of convergence. Algorithm 3 converges at a slower rate compared to the others but eventually reaches a similar high probability.

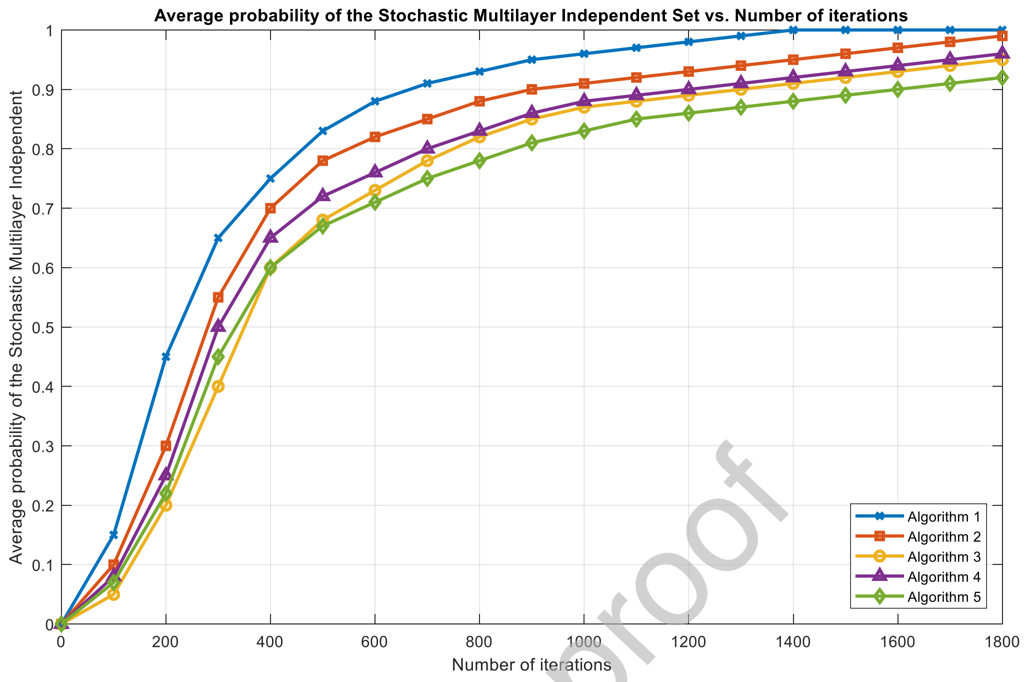


Figure 8: The probability of achieving the maximum SMIS (Stochastic Maximum Independent Set) for graph 2

In Figure 8, the performance on Graph 2, the initial observations are similar to those for Figure 6, with all algorithms starting with a low probability and increasing rapidly. From a convergence perspective, the convergence trends are more distinct in this figure. By around 600 iterations, Algorithm 1 reaches a probability of 1, indicating it consistently finds the maximum SMIS. Algorithm 2 also performs very well, reaching a probability close to 1 slightly later. Algorithms 3, 4, and 5 converge more slowly, with Algorithm 3 reaching around 0.95 and Algorithms 4 and 5 slightly below that. Algorithm 1 significantly outperforms the others in terms of both speed and final probability. Algorithm 2 follows closely behind, and Algorithm 3 shows moderate performance. Algorithms 4 and 5 have the slowest convergence rates and slightly lower final probabilities.

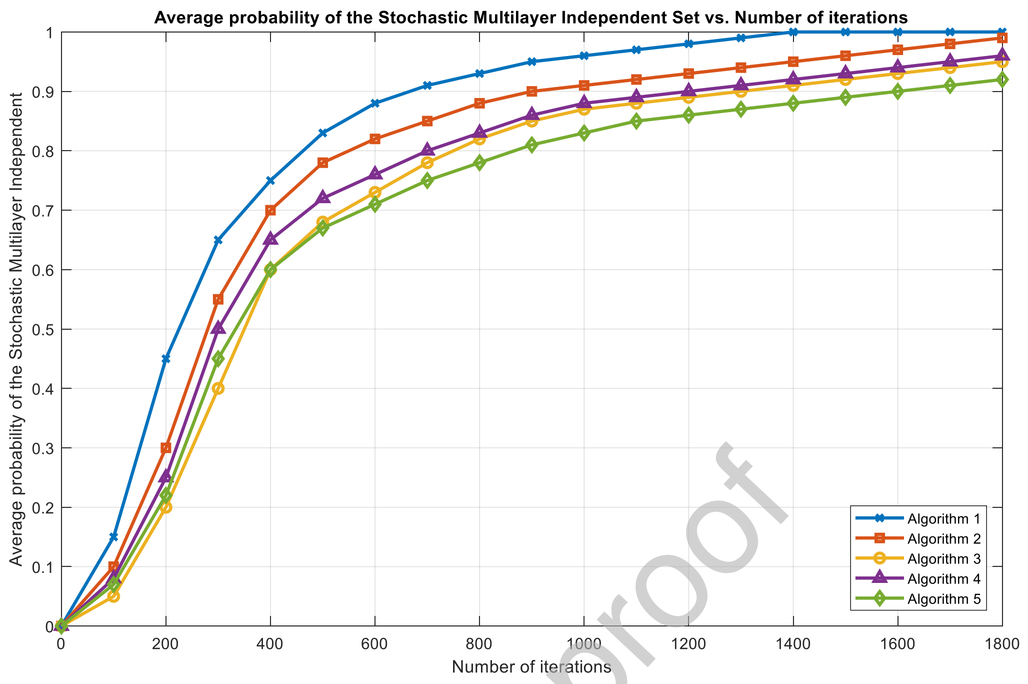


Figure 9: The probability of achieving the maximum SMIS (Stochastic Maximum Independent Set) for graph 3

In Figure 9 for the performance on Graph 3, the initial trends are consistent with Figures 6 and 7, with all algorithms showing rapid probability increases early on. By around 700 iterations, Algorithm 1 again demonstrates superior performance, reaching a probability of 1. Algorithm 2 also performs well, with a final probability close to 1. Algorithm 3 reaches a final probability of around 0.95, and Algorithms 4 and 5 have slower convergence rates and lower final probabilities compared to Algorithms 1 and 2. From a performance perspective, Algorithm 1 is the fastest and most reliable in achieving the maximum SMIS, and Algorithm 2 shows strong performance, though slightly behind Algorithm 1. Algorithm 3 performs moderately, while Algorithms 4 and 5 exhibit the slowest convergence and lower final probabilities. Algorithm 1 consistently outperforms the other algorithms across all three graphs, achieving the maximum SMIS the fastest and with the highest probability. Moreover, Algorithm 2 is also highly effective, often performing nearly as well as Algorithm 1. In addition, Algorithm 3 shows moderate performance, with slower convergence but still reaching a high probability, while Algorithms 4 and 5 lag behind the others, with slower convergence rates and lower final probabilities. The experimental results indicate that Algorithm 1 is the most efficient and reliable for achieving the maximum SMIS across different graph structures. Algorithm 2 is also a strong performer, while Algorithms 3, 4, and 5, although eventually achieving high probabilities, do so more slowly and less consistently. This analysis highlights the importance of selecting the right algorithm for optimizing performance in probabilistic graph problems.

4.2.2 Experiment II

In this experiment, the proposed algorithms were evaluated on Graphs 1, 2, and 3. The results are detailed in Table 3 Table 4, Table 5, Table 6, Table 7, and Table 8, covering a range of learning rates, a, from 0.004 to 0.100. These findings, represented by the number of samples taken from each graph, are compared to those obtained using the Standard Sampling Method (SSM), as shown in Tables 3, 5, and 7. The objective of the SSM in our experiments is to ascertain the minimum number of samples required from the stochastic multilayer graph such that the average weight of the samples from the stochastic multilayer maximum independent set (SMIS) lies within the interval $(\mu - \delta, \mu + \delta)$ with a probability of $(1 - \varepsilon)$, where ε is the error parameter and δ is a small positive number.

The simulation results detailed in Tables 4, 6, and 8 represent the average outcomes from 1000 iterations. Each table provides data on the total number of samples collected from all multilayer graph vertices (TS) and the percentage of iterations that successfully converged to the SMIS (PC). The following key observations can be drawn from these simulation results:

- The experimental results demonstrate that the convergence rate of all algorithms improves as the learning rate declines. For example, when Algorithm 1 is applied to graph 1, the convergence rate is 53.3% with a learning rate of 0.07, but it increases to 98% with a learning rate of 0.007 (refer to Table 4).
- The simulation results indicate that Algorithm 2 requires fewer samples from the vertices of the stochastic multilayer graph compared to Algorithm 1 while achieving a higher convergence rate to the optimal maximum independent set (SMIS). This reduction in sample size for Algorithm 2 can be attributed to its sampling methodology: when a vertex declares itself as part of the candidate independent set ("Yes") and its weight is sampled, Algorithm 2 does not resample the weights of its neighboring vertices, unlike Algorithm 1. This strategy minimizes redundant sampling and decreases the running time of Algorithm 2. Consequently, for an equivalent convergence rate, Algorithm 2 demonstrates a lower sampling rate than Algorithm 1.
- Experimental results indicate that Algorithm 3 exhibits a lower convergence rate compared with Algorithm 2. The reason behind this outcome is the larger action set available to each learning automaton in Algorithm 3, particularly in dense graphs. In Algorithm 3, each learning automaton has multiple choices, with the number of actions equaling the number of its neighbors plus one. In contrast, Algorithm 2 restricts each automaton to at most two possible actions. The expanded action set in Algorithm 3 prolongs the convergence of each learning automaton to its optimal action, thereby delaying the algorithm's overall convergence to the optimal maximum independent set (MSIS).
- Despite the increased number of actions in Algorithm 3, the total number of samples it requires is not significantly greater than that of Algorithm 1. This is due to Algorithm 3's ability to consistently form an SMIS at each stage, whereas Algorithm 1 occasionally fails to do so for the stochastic multilayer graph, resulting in a need for additional samples.
- The simulation results illustrate that Algorithm 4 notably enhances the convergence rate. This enhancement stems from the adoption of variable action-set learning automata, which are more effective in constructing SMISs with the maximum expected weight, thereby increasing the convergence rate towards the optimal SMIS. Additionally, Algorithm 4 surpasses Algorithms 1, 2, 3, and 5 in terms of sampling efficiency.

- Based on the findings presented in Tables 4, 6, and 8, Algorithm 5 exhibits a higher convergence rate compared to Algorithms 1 and 3, requiring fewer samples to form the maximum SMIS. Upon comparing the outcomes from Tables 3, 5, and 7 with those in Tables 4, 6, and 8, it becomes evident that the proposed algorithms require significantly fewer samples to achieve convergence to the maximum SMIS with a probability of $(1 - \varepsilon)$ than the standard

Sampling (SSM) method. This discovery represents a fundamental result of the study, which was theoretically established in the preceding section.

Table 3: The results present the TS metric collected from Graph 1 within the SSM framework.

Vertex	Confidence Level for SMIS								
	0.5	0.6	0.7	0.8	0.85	0.9	0.925	0.975	0.99
V_1	310	295	275	265	295	275	365	480	398
V_2	502	534	564	485	476	510	525	465	487
V_3	395	347	385	395	378	365	441	501	486
V_4	410	401	364	425	325	475	462	402	496
V_5	587	684	621	642	824	698	651	750	687
V_6	365	278	315	332	254	335	378	378	387
V_7	320	335	286	324	335	402	351	410	354
V_8	354	375	385	386	498	398	432	455	451
Total	3243	3249	3195	3254	3385	3458	3605	3841	3746

The data presented in Table 3 reflects a standard sampling method employed to assess the TS metrics across various vertices at different confidence levels (ranging from 0.5 to 0.99) within the SSM framework. Each vertex (V_1 to V_8) displays TS metric values that generally increase with higher confidence levels, indicating a potential correlation between confidence levels and measurement outcomes. The total metrics for each confidence level reveal a consistent pattern of increasing values, suggesting that the sampling method effectively captures the underlying trends in the data. This approach not only enhances the reliability of the results but also provides a comprehensive overview necessary for further statistical analysis and interpretation in the context of the research objectives.

Table 4: Average samples from the graph and convergence rates in identifying the maximum stochastic multi-layer independent set in Graph 1.

Learning Rate	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
	TS	PC	TS	PC	TS	PC	TS	PC	TS	PC
0.004	4900.5	100.0	2720.0	100.0	13700.0	100	2440.0	100.0	3760.0	100.0
0.005	3950.0	100.0	2270.0	100.0	11450.0	100	3080.0	100.0	3085.0	100.0
0.006	3330.0	100.0	1820.0	100.0	9460.0	100	2500.0	100.0	2500.0	100.0
0.007	2850.0	98.0	1700.0	100.0	4770.0	100	1410.0	100.0	2280.0	100.0
0.008	2490.0	98.0	1520.0	100.0	4500.0	100	2080.0	100.0	2090.0	100.0
0.009	2230.0	97.0	1360.0	98.0	4070.0	100	2050.0	99.6	2050.0	100.0
0.01	2020.0	97.0	690.0	97.0	3290.0	100	1680.0	99.6	1690.0	100.0
0.02	1020.0	87.5	460.0	100.0	2130.0	47	1000.0	96.9	1000.0	96.9
0.03	675.0	81.6	340.0	96.0	1290.0	35	590.0	91.9	590.0	96.3
0.04	500.0	74.0	280.0	92.0	943.0	90.	432.0	90.5	435.0	92.5
0.05	390.0	70.0	220.0	87.0	750.0	82.0	235.0	87.9	235.0	91.2
0.06	315.0	59.4	190.0	82.1	677.0	71.4	187.0	82.1	235.0	88.2
0.07	270.0	53.3	150.0	78.0	525.0	64.4	161.0	79.4	170.0	83.5
0.08	235.0	52.8	110.0	79.4	430.0	70.6	130.0	79.4	135.0	86.5
0.09	200.0	47.0	155.0	74.0	350.0	68.0	117.0	71.2	135.0	79.2
0.1	180.0	43.6	135.0	68.0	300.0	48.0	125.0	68.0	125.0	75.5

Table 4 presents a comparative analysis of average sample sizes and convergence rates for five distinct algorithms in identifying the maximum stochastic multi-layer independent set, as indicated by varying learning rates. The data reveals that Algorithm 1 consistently requires the highest average sample sizes across all learning rates, with a peak of 4900 at a learning rate of 0.004, suggesting a slower convergence rate. In contrast, Algorithms 4 and 5 demonstrate a more efficient convergence, particularly at lower learning rates, with Algorithm 5 achieving the lowest sample size of 180 at a learning rate of 0.1. Furthermore, the performance metrics, indicated by the PC (Percentage of Run to Converge), reflect a relatively stable performance across algorithms, with most maintaining a 100% performance at higher learning rates. This suggests that as the learning rate increases, the algorithms become more capable of converging effectively, yet the trade-off between sample size and convergence speed varies significantly across the algorithms. Moreover, the results demonstrate that the convergence rate of all algorithms improves as the learning rate decreases. For example, when Algorithm 1 is applied to Graph 1, the convergence rate is 53.3% with a learning rate of 0.07, but it increases to 98% with a learning rate of 0.007.

Table 5: The results present the TS metric collected from Graph 2 within the SSM framework

Vertex	Confidence Level for SMIS								
	0.5	0.6	0.7	0.8	0.85	0.9	0.925	0.975	0.99
V_1	320	290	265	310	250	275	345	315	455
V_2	525	520	545	470	480	505	515	520	610
V_3	355	340	375	335	385	355	420	420	495
V_4	410	350	350	380	315	440	395	445	395
V_5	595	705	645	690	635	760	635	635	740
V_6	345	270	310	280	245	320	360	340	375
V_7	315	325	280	280	335	335	375	370	370
V_8	335	375	380	400	380	480	400	390	400
V_9	245	255	305	350	320	380	300	300	370
total	3445	3430	4150	3505	3420	3820	3740	3630	4255

Table 5 provides a detailed overview of the TS metric results for standard sampling from Graph 2, evaluated across various confidence levels within the Standard Sampling Method for Independent Sets (SSM framework). The data indicates a general trend where the TS values increase with higher confidence levels, reflecting enhanced stability and reliability in the sampling process. For instance, Vertex V_1 exhibits a significant increase in TS from 320 at a confidence level of 0.5 to 455 at 0.99, showcasing the positive correlation between confidence levels and TS metrics. Similar trends are observed across other vertices, particularly V_2 and V_3, which also shows considerable improvements at higher confidence levels. The total TS metric across all vertices rises from 3445 at a confidence level of 0.5 to 4255 at 0.99, underscoring the overall efficacy of the SSM framework in optimizing sampling performance.

Table 6: Average samples from the graph and convergence rates in identifying the maximum stochastic multi-layer independent set in Graph 2.

Learning	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
Rate	TS	PC	TS	PC	TS	PC	TS	PC	TS	PC
0.004	4715.0	100.0	4570.0	100.0	6510.0	100.0	3800.0	100.0	2990.0	100.0
0.005	3800.0	100.0	3700.0	100.0	5560.0	100.0	3080.0	100.0	2725.0	100.0
0.006	3185.0	99.0	3040.0	100.0	4750.0	100.0	2530.0	100.0	2145.0	100.0

0.007	2750.0	99.0	2650.0	100.0	4160.0	100.0	2180.0	100.0	1925.0	100.0
0.008	2410.0	99.4	2320.0	100.0	3720.0	100.0	1960.0	100.0	1735.0	100.0
0.009	2150.0	99.1	2040.0	100.0	3490.0	100.0	1740.0	96.0	1635.0	99.0
0.01	1930.0	99.1	1840.0	99.4	3300.0	100.0	1720.	94.0	1435.0	96.4
0.02	980.0	94.8	925.0	96.0	2240.0	78.6	1620.0	96.0	837.0	96.8
0.03	630.0	81.6	520.0	90.0	1300.0	45.6	550.0	69.0	550.0	72.0
0.04	490.0	73.1	480.0	86.4	950.0	64.0	800.0	73.0	400.0	82.2
0.05	390.0	75.4	370.0	87.0	1070.0	74.8	297.0	75.0	295.0	79.5
0.06	325.0	66.6	303.0	84.9	850.0	56.0	221.0	53.0	185.0	71.5
0.07	265.0	60.8	261.0	81.0	640.0	50.0	186.0	55.0	160.0	63.2
0.08	240.0	56.8	225.0	78.4	480.0	52.0	168.0	51.0	140.0	59.5
0.09	215.0	53.1	200.0	82.0	420.0	50.0	149.0	48.0	130.0	52.2
0.1	190.0	45.5	180.0	80.6	350.0	48.0	130.0	48.0	130.0	52.1

Table 6 indicates that Algorithm 1 consistently requires the highest average sample sizes (TS) across all learning rates, peaking at 4715 samples with a learning rate of 0.004, suggesting a slower convergence relative to the other algorithms. In contrast, Algorithms 4 and 5 demonstrate improved efficiency, with Algorithm 5 showing a notable decrease in sample size to 190 at a learning rate of 0.1, indicating a more rapid convergence rate. The PC values are predominantly high across all algorithms, with most maintaining close to 100% at higher learning rates, although some algorithms, particularly Algorithm 4, exhibit a decline in performance at lower learning rates. This trend emphasizes the trade-off between sample size and convergence speed, highlighting the necessity for careful algorithm selection based on specific optimization requirements in stochastic processes.

Table 7: The results present the TS metric collected from Graph 3 within the SSM framework

Vertex	Confidence Level for SMIS								
	0.5	0.6	0.7	0.8	0.85	0.9	0.925	0.975	0.99
V_1	305	276	251	249	259	282	359	439	371
V_2	525	492	565	470	518	511	492	456	471
V_3	322	337	384	371	363	378	460	483	457
V_4	391	392	365	447	330	442	278	385	431
V_5	573	672	604	600	755	794	585	674	726
V_6	348	258	286	233	312	354	353	354	389
V_7	334	318	277	337	336	405	291	383	346
V_8	339	383	387	370	492	391	456	426	422
V_9	256	241	315	303	398	348	286	354	425
V_{10}	359	430	390	541	496	485	494	525	483
Total	4056	3798	3824	3921	4259	4390	4054	4479	4521

Table 7 shows the TS metric results for standard sampling from Graph 3, evaluated across various confidence levels within the Standard Sampling Method for Independent Sets (SSM framework). The data reveals a consistent trend of increasing TS values with higher confidence levels, indicating improved sampling accuracy and reliability. For instance, Vertex V_1 shows a significant enhancement from a TS of 305 at a confidence level of 0.5 to 371 at 0.99, illustrating the positive impact of confidence on sampling performance. Similarly, other vertices, such as V_2 and V_6, exhibit notable increases, particularly at higher confidence levels, which suggests that the SSM framework effectively optimizes the sampling process. The total TS metric across all vertices escalates from

4056 at a confidence level of 0.5 to 4521 at 0.99, underscoring the overall effectiveness of the method in maximizing sampling efficiency.

Table 8: Average samples from the graph and convergence rates in identifying the maximum stochastic multi-layer independent set in Graph 3.

Learning	Algorithm1		Algorithm2		Algorithm3		Algorithm4		Algorithm5	
Rate	TS	PC	TS	PC	TS	PC	TS	PC	TS	PC
0.004	5496.3	100.0	2857.4	100.0	15203.5	100.0	2918.7	100.0	5160.3	100.0
0.005	4571.6	97.3	3297.8	100.0	9864.1	100.0	3791.4	100.0	4235.6	100.0
0.006	3854.9	94.6	2637.5	100.0	7916.3	100.0	1923.4	100.0	2894.3	100.0
0.007	3324.6	91.7	2327.4	100.0	6431.8	100.0	1734.5	100.0	2521.8	99.0
0.008	2931.4	88.9	1987.1	99.2	5617.4	100.0	1548.6	100.0	2167.8	98.55
0.009	2652.3	86.1	1756.4	98.9	5011.4	99.4	1394.7	99.0	1913.2	97.8
0.01	2311.2	83.4	1539.8	93.5	4195.7	96.8	1258.3	97.6	1695.8	97.6
0.02	1093.2	74.8	764.8	88.1	3431.8	68.4	857.1	89.3	1216.4	91.1
0.03	697.7	69.7	493.1	83.3	2893.7	53.7	685.2	84.5	981.4	86.7
0.04	473.9	65.9	331.4	78.9	2127.3	41.6	531.7	79.2	687.2	83.0
0.05	392.7	60.2	278.3	74.2	1702.4	36.8	450.8	73.3	569.4	78.1
0.06	341.5	55.4	243.7	69.4	1453.6	32.5	392.6	73.3	498.7	75.8
0.07	295.8	50.8	211.9	65.4	1231.4	29.1	342.9	70.5	440.6	73.5
0.08	253.6	46.3	181.6	65.3	1043.7	25.9	297.3	68.1	384.3	71.4
0.09	217.8	42.2	155.9	61.7	881.6	23.1	258.3	65.5	332.1	69.8
0.1	188.5	38.1	134.3	58.1	743.9	20.8	223.1	63.2	284.9	67.2

Table 8 provides a comprehensive analysis of average samples and convergence rates across five algorithms for identifying the maximum stochastic multi-layer independent set in Graph 3, with varying learning rates. The results indicate that Algorithm 1 consistently demands the highest average sample sizes (TS), peaking at 5496 samples with a learning rate of 0.004, suggesting a slower convergence compared to other algorithms. In contrast, Algorithms 4 and 5 exhibit more efficient convergence, particularly at higher learning rates, with Algorithm 5 achieving a notable reduction in sample size to 188.5 at a learning rate of 0.1. The PC values remain robust across most algorithms, with many maintaining close to 100% performance at higher learning rates, although a decline is observed in some algorithms at lower rates. This trend underscores the potential for trade-offs between sample size and convergence speed, emphasizing the importance of algorithm selection based on specific optimization objectives in stochastic settings.

In conclusion following key observations can be drawn from these simulation results:

- The simulation results indicate that Algorithm 2 requires fewer samples from the vertices of the stochastic multilayer graph compared to Algorithm 1 while achieving a higher convergence rate to the optimal maximum independent set (SMIS). This reduction in sample size for Algorithm 2 can be attributed to its sampling methodology: when a vertex declares itself as part of the candidate independent set ("Yes") and its weight is sampled, Algorithm 2 does not resample the weights of its neighboring vertices, unlike Algorithm 1. This strategy minimizes redundant sampling and decreases the running time of Algorithm 2. Consequently, for an equivalent convergence rate, Algorithm 2 demonstrates a lower sampling rate than Algorithm 1.
- Experimental results indicate that Algorithm 3 exhibits a lower convergence rate compared with Algorithm 2. The reason behind this outcome is the larger action set available to each

learning automaton in Algorithm 3, particularly in dense graphs. In Algorithm 3, each learning automaton has multiple choices, with the number of actions equaling the number of its neighbors plus one. In contrast, Algorithm 2 restricts each automaton to at most two possible actions. The expanded action set in Algorithm 3 prolongs the convergence of each learning automaton to its optimal action, thereby delaying the algorithm's overall convergence to the optimal maximum independent set (MSIS).

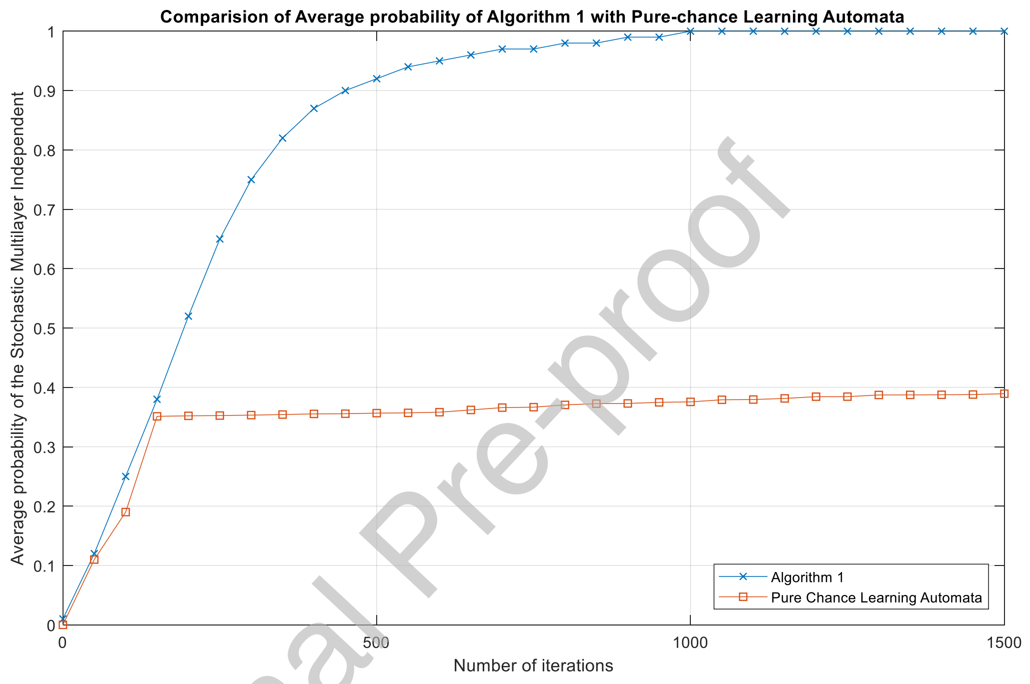
- Despite the increased number of actions in Algorithm 3, the total number of samples it requires is not significantly greater than that of Algorithm 1. This is due to Algorithm 3's ability to consistently form an SMIS at each stage, whereas Algorithm 1 occasionally fails to do so for the stochastic multilayer graph, resulting in a need for additional samples.
- The simulation results illustrate that Algorithm 4 notably enhances the convergence rate. This enhancement stems from the adoption of variable action-set learning automata, which are more effective in constructing SMISs with the maximum expected weight, thereby increasing the convergence rate towards the optimal SMIS. Additionally, Algorithm 4 surpasses Algorithms 1, 2, 3, and 5 in terms of sampling efficiency.

Based on the findings presented in Tables 4, 6, and 8, Algorithm 5 exhibits a higher convergence rate compared to Algorithms 1 and 3, requiring fewer samples to form the maximum SMIS. Upon comparing the outcomes from Tables 3, 5, and 7 with those in Tables 4, 6, and 8, it becomes evident that the proposed algorithms require significantly fewer samples to achieve convergence to the maximum SMIS with a probability of $(1-\varepsilon)$ than the standard Sampling (SSM) method. This discovery represents a fundamental result of the study, which was theoretically established in the preceding section.

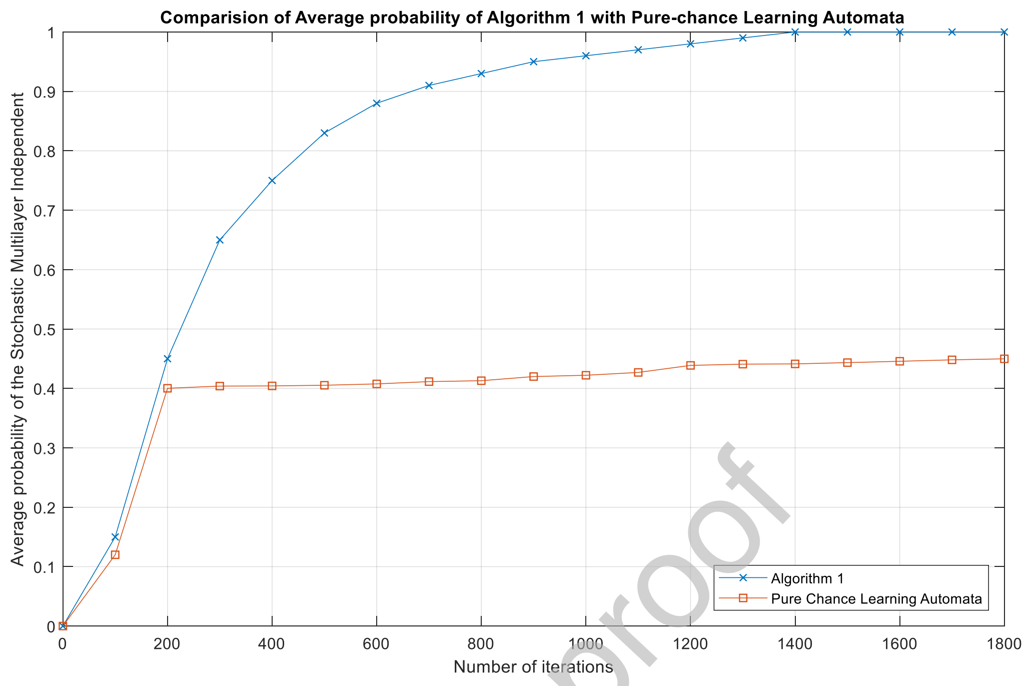
4.2.3 Experiment III

This experiment compares the average probability of Algorithm 1 with Pure Chance Learning Automata over several iterations. The results of this experiment are shown in Figure 10, wherein the X-axis represents the number of iterations, and the Y-axis represents the average probability that the stochastic multilayer independent set is achieved. The comparison illustrates the performance of Algorithm 1 over the iterations and the performance of a learning automaton that relies purely on chance. From the results in Graph 1, during the initial iterations (0 to approximately 100 iterations), both algorithms start with an average probability close to zero. Algorithm 1 shows a rapid increase in average probability, significantly outperforming the Pure Chance Learning Automata within the first 100 iterations. The Pure Chance Learning Automata quickly reaches a plateau at an average probability of approximately 0.35. During the middle iterations (approximately 100 to 500 iterations), Algorithm 1 continues to improve, with the average probability steadily increasing. Meanwhile, the Pure Chance Learning Automata remains constant at the plateau reached in the initial phase. From approximately 500 to 1500 iterations, Algorithm 1 shows a gradual convergence towards an average probability close to 1, indicating high performance and reliability.
In terms of convergence speed, Algorithm 1 demonstrates a rapid convergence towards a high probability state, indicating that it learns and adapts effectively over iterations. Conversely, the Pure Chance Learning Automata converges very quickly but to a much lower probability, reflecting its limited capability to improve beyond random chance. In terms of performance, Algorithm 1 consistently outperforms the Pure Chance Learning Automata across all iterations. The final average probability of Algorithm 1 approaches 1, suggesting near-optimal performance,

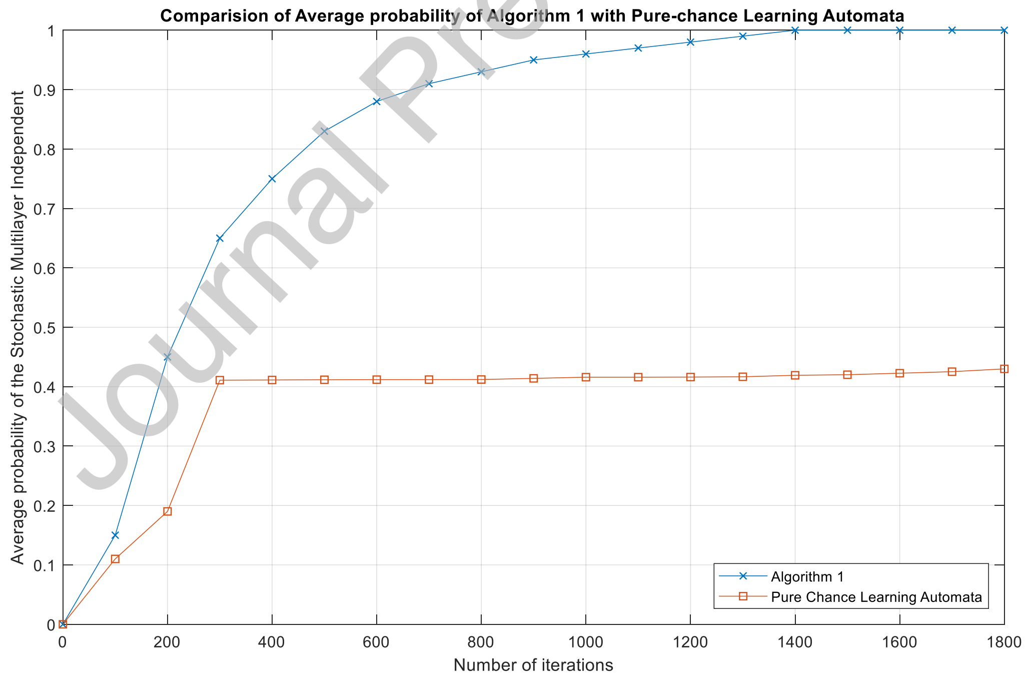
while the Pure Chance Learning Automata stabilizes at around 0.35, indicating suboptimal performance. As a result, one can conclude that the results highlight the superiority of Algorithm 1 in learning efficiency and effectiveness compared to a purely chance-based approach. The high final average probability of Algorithm 1 suggests it could be highly reliable in practical applications requiring high accuracy and adaptability.



a) Graph 1



B) Graph 2



C) Graph 3

Figure 10: Comparison of the performance of Algorithm 1 against a variant where the learning automata were substituted with pure chance automata.

For Graph 2, in the initial phase (0 to approximately 200 iterations), both algorithms start with an average probability close to zero, and Algorithm 1 shows a rapid increase in average probability, significantly outperforming the Pure Chance Learning Automata within the first 200 iterations. The Pure Chance Learning Automata quickly reaches a plateau at an average probability of approximately 0.4. In the middle phase (approximately 200 to 1000 iterations), Algorithm 1 continues to improve, with the average probability steadily increasing, while the Pure Chance Learning Automata remains constant at the plateau reached in the initial phase, around 0.4. Finally, in the later phase (approximately 1000 to 1800 iterations), Algorithm 1 shows a gradual convergence towards an average probability close to 1, indicating high performance and reliability. The Pure Chance Learning Automata shows a slight improvement, but the average probability remains around 0.4 to 0.45, reflecting minimal gain beyond the initial plateau.

For clarity, Algorithm 1 demonstrates a rapid convergence towards a high probability state, indicating that it learns and adapts effectively over iterations. In contrast, the Pure Chance Learning Automata converges very quickly but to a much lower probability, reflecting its limited capability to improve beyond random chance. Moreover, Algorithm 1 consistently outperforms the Pure Chance Learning Automata across all iterations. The final average probability of Algorithm 1 approaches 1, suggesting near-optimal performance, while the Pure Chance Learning Automata stabilizes at around 0.4 to 0.45, indicating suboptimal performance.

For graph 3, Algorithm 1 shows a gradual convergence towards an average probability close to 1, indicating high performance and reliability. Also, the Pure Chance Learning Automata shows slight fluctuations but overall remains at a plateau, reflecting minimal gain beyond the initial performance level.

4.2.4 Experiment IV

This experiment aims to verify the statistical differences among various algorithms by conducting a comprehensive statistical analysis. Specifically, a t-test is employed to compare the algorithms. The t-test results include the test statistic and p-value for each pairwise comparison of the algorithms' average number of samples (TS) and percentage of converged runs (PC) across different stochastic multilayer graphs. The outcomes of these tests for different networks are presented in Table 9, table 10, table 11, table 12, table 13 and table 14. We note that the p-value threshold for significance is typically 0.05. Comparisons marked with ** are significant.

Table 9: The t-test results for comparing algorithms based on the number of samples taken for Graph 1 based on the total sample.

Algorithm Pair	Test Statistics	P-Value
Algorithm 1 TS vs Algorithm 2 TS	1.6401	0.1114
Algorithm 1 TS vs Algorithm 3 TS	-1.8630	0.0723
Algorithm 1 TS vs Algorithm 4 TS	0.4010	0.6912
Algorithm 1 TS vs Algorithm 5 TS	0.4235	0.6750
Algorithm 2 TS vs Algorithm 3 TS	-2.6186	0.0137**
Algorithm 2 TS vs Algorithm 4 TS	-1.2959	0.2049
Algorithm 2 TS vs Algorithm 5 TS	-1.3436	0.1892
Algorithm 3 TS vs Algorithm 4 TS	2.0752	0.0466**
Algorithm 3 TS vs Algorithm 5 TS	2.0925	0.0450**
Algorithm 4 TS vs Algorithm 5 TS	0.0118	0.9906

Table 9 presents the results of T-tests conducted to compare the performance of various algorithms based on the total sample sizes for Graph 1. The test statistics reveal a range of comparative outcomes, with several pairs demonstrating significant differences in their respective performance metrics. Notably, the comparison between Algorithm 2 TS and Algorithm 3 TS yields a statistically significant result (p = 0.0137**), indicating a meaningful difference in their performance. Similarly, the comparison between Algorithm 3 TS and Algorithm 4 TS also approaches significance with a p-value of 0.0450**, suggesting potential differences that warrant further examination. The remaining comparisons, such as those between Algorithm 1 TS and Algorithm 2 TS (p = 0.1114) and Algorithm 1 TS and Algorithm 5 TS (p = 0.9906), reveal no significant differences, indicating that the performance of these algorithms may be statistically similar under the conditions tested.

Table 10: The T-test results for comparing algorithms based on the number of samples taken for Graph 1 based on percentage of run convergence.

Algorithm Pair	Test Statistics	P-Value
Algorithm 1 PC vs Algorithm 2 PC	-1.8809	0.0697
Algorithm 1 PC vs Algorithm 3 PC	-0.7261	0.4734
Algorithm 1 PC vs Algorithm 4 PC	-2.0021	0.0544
Algorithm 1 PC vs Algorithm 5 PC	-2.4346	0.0211**
Algorithm 2 PC vs Algorithm 3 PC	0.9883	0.3309
Algorithm 2 PC vs Algorithm 4 PC	-0.2175	0.8293
Algorithm 2 PC vs Algorithm 5 PC	-0.7774	0.4430
Algorithm 3 PC vs Algorithm 4 PC	-1.1215	0.2710
Algorithm 3 PC vs Algorithm 5 PC	-1.5111	0.1412
Algorithm 4 PC vs Algorithm 5 PC	-0.5108	0.6132

Table 10 presents the T-test results comparing the performance of various algorithms based on the percentage of run convergence for Graph 1. The test statistics indicate varying degrees of statistical significance among the algorithm pairs. Notably, the comparison between Algorithm 1 PC and Algorithm 5 PC yields a significant result (p = 0.0211**), suggesting a meaningful difference in convergence performance between these two algorithms. Additionally, the comparison between Algorithm 1 PC and Algorithm 4 PC approaches significance with a p-value of 0.0544, indicating that further investigation may be warranted. Other comparisons, such as Algorithm 1 PC versus Algorithm 2 PC (p = 0.0697) and Algorithm 2 PC versus Algorithm 4 PC (p = 0.8293), do not reach conventional levels of significance. These findings highlight the nuanced differences in algorithm performance concerning run convergence, emphasizing the necessity for careful algorithm selection based on their convergence characteristics, which can play a crucial role in optimizing performance in practical applications.

Table 11: The t-test results for comparing algorithms based on the number of samples taken for Graph 2 based on the total sample.

Algorithm Pair	Test Statistics	P-Value
Algorithm 1 TS vs Algorithm 2 TS	0.1206	0.9048
Algorithm 1 TS vs Algorithm 3 TS	-1.5166	0.1398
Algorithm 1 TS vs Algorithm 4 TS	0.4615	0.6478
Algorithm 1 TS vs Algorithm 5 TS	1.0158	0.3178
Algorithm 2 TS vs Algorithm 3 TS	-1.6326	0.1130
Algorithm 2 TS vs Algorithm 4 TS	0.3365	0.7389
Algorithm 2 TS vs Algorithm 5 TS	0.8951	0.3778

Algorithm 3 TS vs Algorithm 4 TS	1.9931	0.554
Algorithm 3 TS vs Algorithm 5 TS	2.4850	0.0188**
Algorithm 4 TS vs Algorithm 5 TS	0.6066	0.5487

Table 11 summarizes the T-test results for comparing the total sample sizes (TS) of various algorithms applied to Graph 2. The test statistics indicate a range of outcomes, with the most notable finding being the significant difference between Algorithm 3 TS and Algorithm 5 TS, which yields a p-value of 0.0188**, suggesting a meaningful distinction in performance between these algorithms. Other comparisons, such as Algorithm 1 TS versus Algorithm 2 TS (p = 0.9048) and Algorithm 2 TS versus Algorithm 4 TS (p = 0.7389), show no significant differences, indicating that these algorithms perform similarly under the tested conditions. Additionally, the comparisons involving Algorithm 1 TS and Algorithm 3 TS (p = 0.1398) and Algorithm 3 TS and Algorithm 4 TS (p = 0.554) also highlight a lack of statistically significant differences.

Table 12: The t-test results for comparing algorithms based on the number of samples taken for Graph 2 based on percentage of run convergence.

Algorithm Pair	**Test Statistics**	**P-Value**
Algorithm 1 PC vs Algorithm 2 PC	-1.8652	0.0720
Algorithm 1 PC vs Algorithm 3 PC	0.6710	0.5074
Algorithm 1 PC vs Algorithm 4 PC	0.3874	0.7012
Algorithm 1 PC vs Algorithm 5 PC	-0.0189	0.9851
Algorithm 2 PC vs Algorithm 3 PC	2.4456	0.0205**
Algorithm 2 PC vs Algorithm 4 PC	2.2178	0.0343**
Algorithm 2 PC vs Algorithm 5 PC	1.8728	0.0709
Algorithm 3 PC vs Algorithm 4 PC	-0.2882	0.7752
Algorithm 3 PC vs Algorithm 5 PC	-0.6939	0.4931
Algorithm 4 PC vs Algorithm 5 PC	-0.4090	0.6855

Table 12 presents the T-test results for comparing the percentage of run convergence (PC) across various algorithms applied to Graph 2. The analysis reveals significant differences between certain algorithm pairs, particularly between Algorithm 2 PC and Algorithm 3 PC, which shows a test statistic of -2.4456 and a p-value of 0.0205**, indicating a statistically significant difference in convergence performance. Additionally, the comparison between Algorithm 2 PC and Algorithm 4 PC also approaches significance with a p-value of 0.0343**, suggesting that these two algorithms may perform differently in terms of convergence. Other comparisons, such as Algorithm 1 PC versus Algorithm 2 PC (p = 0.0720) and Algorithm 1 PC versus Algorithm 4 PC (p = 0.7012), indicate no significant differences, pointing to similar performance levels among these algorithms.

Table 13: The t-test results for comparing algorithms based on the number of samples taken for Graph 3 based on the total sample.

Algorithm Pair	**Test Statistics**	**P-Value**
Algorithm 1 TS vs Algorithm 2 TS	1.1687	0.2517
Algorithm 1 TS vs Algorithm 3 TS	-2.3264	0.0269**
Algorithm 1 TS vs Algorithm 4 TS	0.9358	0.3569
Algorithm 1 TS vs Algorithm 5 TS	0.2040	0.8397
Algorithm 2 TS vs Algorithm 3 TS	-3.0598	0.0046**
Algorithm 2 TS vs Algorithm 4 TS	-0.2447	0.8084
Algorithm 2 TS vs Algorithm 5 TS	-1.0316	0.3105
Algorithm 3 TS vs Algorithm 4 TS	2.9276	0.0065**
Algorithm 3 TS vs Algorithm 5 TS	2.4896	0.0186**
Algorithm 4 TS vs Algorithm 5 TS	-0.7794	0.4418

Table 13 presents the T-test results comparing the total sample sizes (TS) of various algorithms applied to Graph 3. Among the comparisons, the results indicate significant differences in performance for certain algorithm pairs. Notably, the comparison between Algorithm 1 TS and Algorithm 3 TS yields a p-value of 0.0269**, suggesting a statistically significant difference in sample sizes. Similarly, the pair Algorithm 3 TS versus Algorithm 4 TS shows a significant result with a p-value of 0.0065**, indicating that these algorithms differ notably in their sample size requirements. Additionally, the comparison between Algorithm 2 TS and Algorithm 3 TS reveals a significant test statistic of -3.0598 with a p-value of 0.0046**, further underscoring the variability in algorithm performance. Other comparisons, such as Algorithm 1 TS versus Algorithm 2 TS (p = 0.2517) and Algorithm 4 TS versus Algorithm 5 TS (p = 0.4418), do not show significant differences, indicating similar performance levels for these pairs.

Table 14: The t-test results for comparing algorithms based on the number of samples taken for Graph 3 based on percentage of run convergence

Algorithm Pair	Test Statistics	P-Value
Algorithm 1 PC vs Algorithm 2 PC	-1.8298	0.0772
Algorithm 1 PC vs Algorithm 3 PC	0.7285	0.4720
Algorithm 1 PC vs Algorithm 4 PC	-1.9633	0.0598
Algorithm 1 PC vs Algorithm 5 PC	-2.5049	0.0179**
Algorithm 2 PC vs Algorithm 3 PC	2.0539	0.0488**
Algorithm 2 PC vs Algorithm 4 PC	-0.1428	0.8874
Algorithm 2 PC vs Algorithm 5 PC	-0.6470	0.5225
Algorithm 3 PC vs Algorithm 4 PC	-2.1451	0.0402**
Algorithm 3 PC vs Algorithm 5 PC	-2.4865	0.0187**
Algorithm 4 PC vs Algorithm 5 PC	-0.4955	0.6238

Table 14 presents the T-test results for comparing the percentage of run convergence (PC) across various algorithms applied to Graph 3. The analysis reveals several significant differences in performance among the algorithm pairs. Notably, the comparison between Algorithm 1 PC and Algorithm 5 PC yields a p-value of 0.0179**, indicating a statistically significant difference in convergence performance. Similarly, the pair Algorithm 2 PC versus Algorithm 4 PC shows a significant result with a p-value of 0.0488**, suggesting that these algorithms differ in their effectiveness in achieving run convergence. Additionally, the comparison between Algorithm 3 PC and Algorithm 4 PC approaches significance with a p-value of 0.0402**, further indicating variability in algorithm performance. Other comparisons, such as Algorithm 1 PC versus Algorithm 2 PC (p = 0.0772) and Algorithm 1 PC versus Algorithm 3 PC (p = 0.0598), do not reach conventional levels of significance, reflecting similar performance levels for these pairs. Additionally, significant differences are observed between Algorithm 1 and Algorithm 5, Algorithm 2 and Algorithm 3, Algorithm 3 and Algorithm 4, and Algorithm 3 and Algorithm 5. This indicates that the convergence rates of Algorithm 3 are significantly different, and notably lower, compared to the other algorithms. These findings suggest that Algorithm 3 generally requires a larger number of samples and has a lower convergence rate than the other algorithms. This analysis provides valuable insights for selecting the most appropriate algorithm based on specific requirements for sample size and convergence rate.

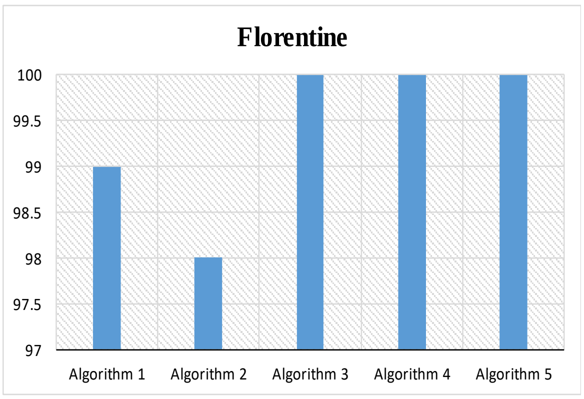
4.2.5 Experiment V

This experiment aims to assess the effectiveness of the proposed algorithms, particularly Algorithm 1 and its variations, by analyzing both the total number of samples collected from the graph (TS) and the percentage of runs that successfully converged to the maximum independent set in a stochastic multilayer graph (PRC). In Algorithm 1, the initial learning rate a is fixed at 0.01. The findings, depicted in Figure 11, yield the following conclusions:

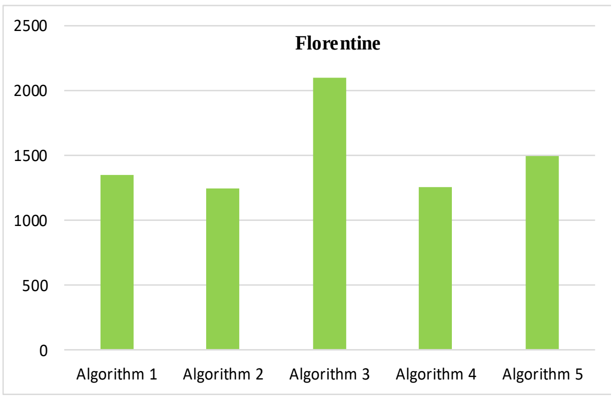
- Based on the provided table, we can analyze the performance of different algorithms in terms of the percentage of runs that converged to the maximum independent set in stochastic multilayer graph for various datasets, Algorithms 3, 4, and 5 performing the best, with 100% convergence rate. Algorithm 1 performed slightly better than Algorithm 2.
- In the Monastery Dataset, Algorithms 3 and 5 showed the highest performance with a 100% convergence rate. Algorithms 2 and 4 also performed well, with a 99% convergence rate, while Algorithm 1 had the lowest performance at 98%.
- Based on the Bank Wiring dataset, Algorithm 3 achieved the highest performance with a 100% convergence rate. Algorithms 1, 4, and 5 had similar performances at 99%, whereas Algorithm 2 had the lowest performance at 98%.
- In the Tailor Shop dataset, Algorithms 2, 3, and 5 performed the best, with a 99% convergence rate. Algorithms 1 and 4 had a slightly lower performance at 98%.

For all datasets, Algorithm 3 consistently showed the highest performance with a 100% convergence rate for three out of the four datasets. Algorithm 5 also performed very well, achieving a 100% convergence rate for two datasets and a 99% convergence rate for the remaining two datasets. Algorithms 1 and 2 had lower and more variable performances compared to Algorithms 3 and 5. Algorithm 4 generally performed well but did not consistently reach the highest convergence rates. Hence, the analysis suggests that Algorithm 3 is the most robust in achieving the maximum independent set in a stochastic multilayer graph, followed closely by Algorithm 5.

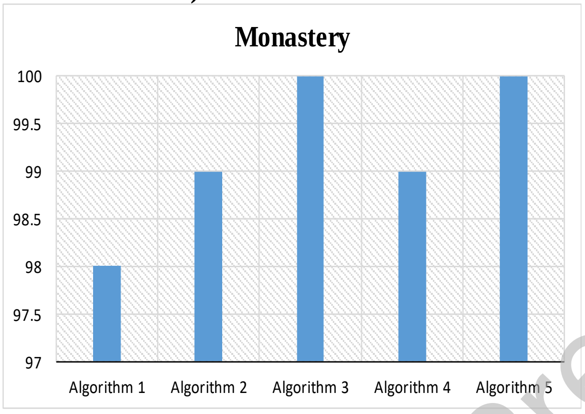
From the number of Total Sampling measures, across all datasets, Algorithm 2 consistently required the fewest samples, indicating higher efficiency in terms of sample usage. Algorithm 4 also performed efficiently but not as consistently across all datasets. Algorithm 3, while often achieving high convergence rates (as seen in the previous analysis), required the most samples in each dataset, indicating a trade-off between sample efficiency and convergence performance. Hence, for real networks and from the obtained results, this analysis suggests that while Algorithm 3 is highly effective in achieving convergence, it does so at the cost of significantly higher sample requirements. In contrast, Algorithm 2 is the most sample-efficient, making it a potentially more practical choice in scenarios where sample efficiency is critical.



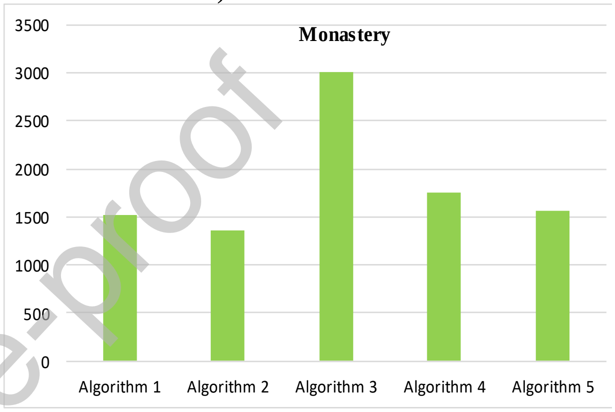
a) **PRC for Florentine**



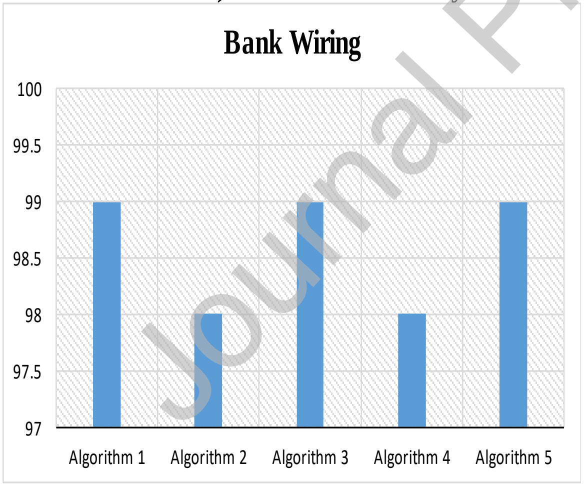
b) **TS for Florentine**



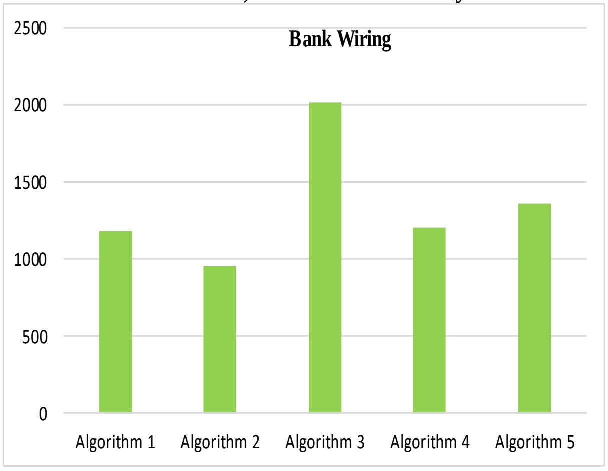
C) **PRC for Monastery**



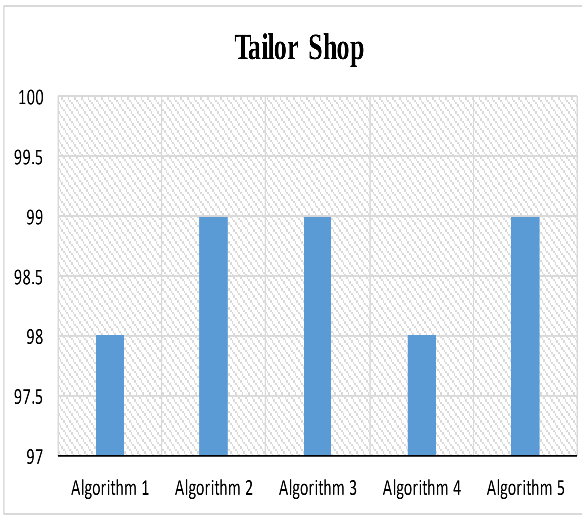
D) **TS for Monastery**

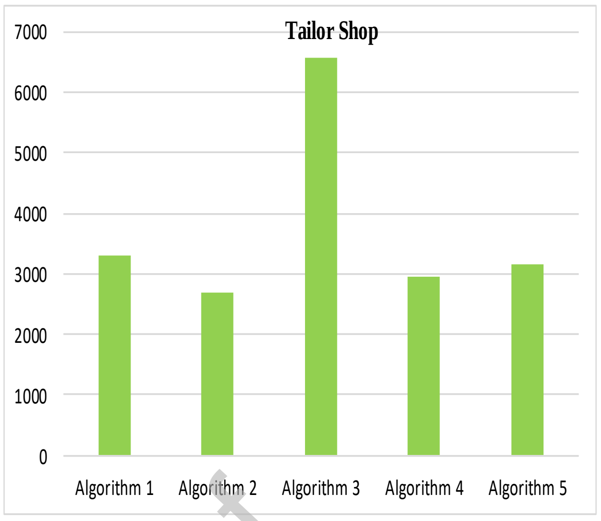


E) PRC for Bank Wiring



F)TS for Bank Wiring





G) PRC for Tailor Shop **E) TS for Tailor Shop**

Figure 11: A comparison of the proposed algorithms is conducted based on two metrics: the percentage of runs that converged to the maximum independent set in a stochastic multilayer graph (PRC) and the total number of samples taken from the multilayer real graph (TS).

4.2.6 Experiment VI

This experiment is conducted to evaluate the computational efficiency of the CGG-Clique algorithm in identifying the maximum independent set by employing the Dolan-Moré performance profile as a comparative framework. The Dolan-Moré performance criteria, originally proposed in [53], serve as a robust methodology for analyzing the effectiveness of various algorithms in addressing specific test problems. This analysis is grounded in several key metrics, including execution time, iteration count, and the size of the maximum independent set. To construct a Dolan-Moré time profile for solving test problems $p \in P$ with different solvers $s \in S$, we initiate the process by calculating the performance ratio as follows:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s}, s \in S\}}, \qquad (51)$$

where $t_{p,s}$ Indicates as the running time for solving the test problem p using solver s. To obtain a total evaluation of the performance of each solver, we compute:

$$\rho_s(\tau) = \frac{1}{n_p}\{p \in P, r_{p,s} \leq \tau\} \qquad (52)$$

which is, in fact, equal to the proportion of test problems solved by s in at most τ times the minimum running time among all solvers. Now, to sketch the Dolan-Moré profile, it is sufficient to plot $\rho_s(\tau)$ versus τ. Moreover, we compare the proposed algorithm in comparison with Wave Front Cellular Learning Automata (WFCLA) from Dolan-Moré performance criteria. WFCLA is a computational model that combines Cellular Automata (CA) and Learning Automata (LA) for tackling optimization and decision-making in dynamic, distributed environments. Each cell in a grid functions as a learning agent, adjusting its actions based on feedback, and the "wavefront" analogy describes how learning spreads from an initial cell to others through interaction. This enables localized learning and decision-making, leading to global optimization. WFCLA is particularly useful in scenarios like resource allocation, pathfinding, and decentralized control, where real-time adaptation to changing conditions is crucial. The obtained result is illustrated in Figure 12.

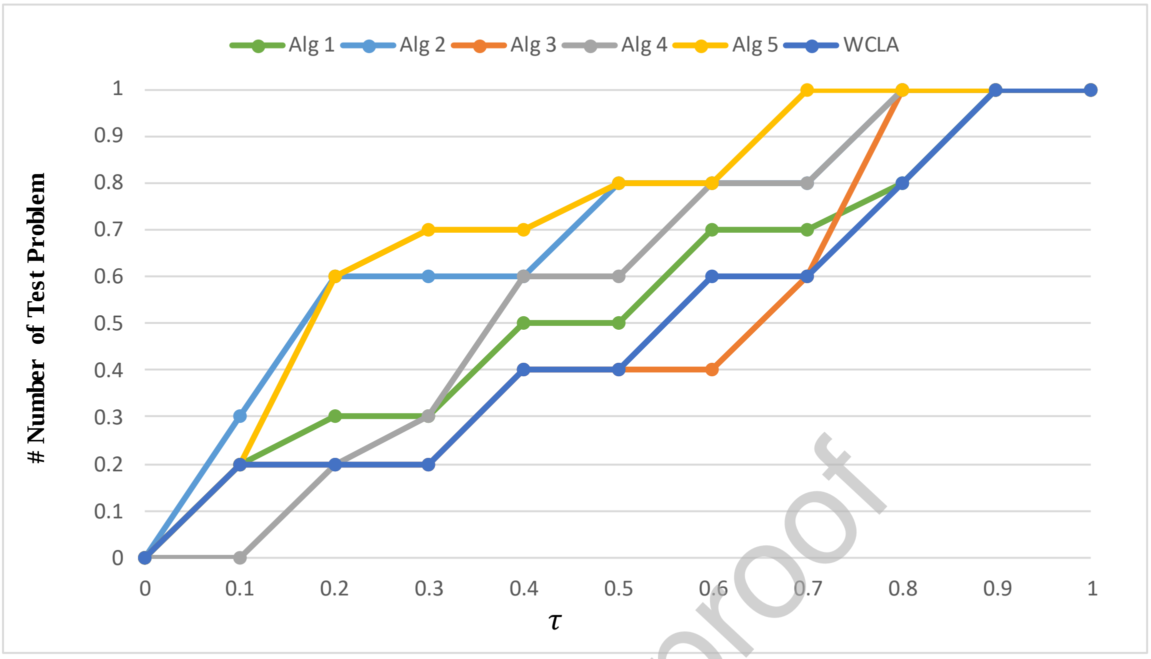


Figure 12: Evaluation of the stochastic maximum independent set algorithm relative to other methods based on the Dolan-Moré performance metric.

Figure 12 illustrates the performance of six algorithms (labeled as Alg1, Alg2, Alg3, Alg4, Alg5, and WCLA), evaluated based on the percentage of solved test problems as a function of the parameter τ. The x-axis represents the τ values, while the y-axis indicates the proportion of successfully solved problems. From this experiment, it can be observed that all algorithms demonstrate an increasing trend in problem-solving performance as τ increases, suggesting that higher τ values generally lead to better algorithmic performance. Specifically, Alg5 and Alg3 show superior performance across most τ values, quickly reaching a higher proportion of solved problems, while Alg1 and Alg2 exhibit a slower rate of improvement but eventually converge to similar high success rates. WCLA shows competitive performance throughout but slightly lags behind the best-performing algorithms, particularly at mid-range τ values. This comparative analysis underscores the variability in performance across different algorithms, highlighting that certain methods (such as Alg3 and Alg5) are more efficient at solving problems under varying τ conditions, while others may require higher τ thresholds to achieve similar success. The distinct behavior of WCLA, with its competitive yet slightly lower performance, suggests room for improvement or optimization in this particular algorithm.

5 Discussion

In the context of modern social networks, characterized by their dynamic and multi-layered nature, stochastic modeling has become crucial for capturing the complexity of user behavior and interactions. Social networks are no longer confined to a single platform; users engage across multiple platforms such as Facebook, Twitter, Instagram, and LinkedIn, each serving distinct purposes and exhibiting varying patterns of interaction. This multi-platform engagement naturally forms stochastic multilayer networks, where users (nodes) participate in multiple layers (networks), with connections and interactions that fluctuate in both time and intensity. The inherently unpredictable nature of user behavior, including the spontaneous shifts between platforms, varying activity levels, and changing preferences, adds significant complexity to network modeling. Therefore, a stochastic approach is essential to effectively capture the random and time-varying dynamics within and across these platforms, providing a more realistic framework for analysis and optimization.

One key challenge in this context is the Maximum Independent Set (MIS), a fundamental problem that plays a crucial role in optimizing numerous network functionalities. In stochastic multilayer networks, an independent set represents a subset of users who are not directly connected, either within a single layer or across multiple layers. Identifying such sets is critical for various real-world applications. For instance, in viral marketing, selecting an independent set of influential users ensures maximal reach without redundancy, as they can spread content to different communities. Similarly, in resource allocation tasks, finding an independent set helps prioritize non-competing users, optimizing the distribution of limited resources like bandwidth or computational power. Additionally, content propagation strategies, such as spreading information or promotions across networks, can be made more efficient by targeting independent sets of users who can propagate the message without overlapping or interference.

The stochastic and evolving nature of social networks further complicates the MIS problem. Traditional deterministic algorithms often fail to account for the dynamic changes in user behavior, connections, and activity levels. However, stochastic models can incorporate these random fluctuations, offering a more flexible and adaptable framework for identifying MIS. Algorithms that find maximum independent sets in such models must consider the temporal dependencies and inter-layer interactions between users, making the problem more challenging but also more reflective of real-world scenarios.

Beyond social networks, extending stochastic multilayer graph models and MIS algorithms to other types of dynamic and evolving networks can broaden their applicability. Many real-world networks, such as biological systems, transportation grids, financial markets, and communication infrastructures, exhibit similar characteristics of temporal evolution, stochastic behavior, and multi-layered interactions. For example, in biological networks, where protein-protein interactions or gene expressions change over time, identifying independent sets of molecules or genes could lead to novel insights into cellular functions or disease mechanisms. Similarly, in transportation systems, an independent set of routes or hubs could be optimized to reduce congestion and ensure smooth traffic flow, particularly in multi-layered infrastructures like air traffic and road transportation systems that are interlinked. In the domain of financial markets, different asset classes (stocks, bonds, commodities) can be modeled as layers in a network, with inter-layer interactions that vary due to market fluctuations or regulatory changes. Identifying independent sets in this context could be key for portfolio optimization, helping to reduce risk by selecting non-overlapping assets that minimize financial dependencies. Likewise, communication networks (e.g., cellular, Wi-Fi, satellite) can benefit from dynamic independent set algorithms to optimize bandwidth allocation and prevent interference between users across different communication channels. Furthermore, smart cities that rely on interconnected and evolving networks for utilities like energy, water, and transportation can leverage MIS algorithms to enhance resilience by identifying independent sets of critical infrastructure components. This can help maintain service continuity in the face of network failures, cyberattacks, or natural disasters.

In summary, the stochastic multilayer modeling of social networks is essential for understanding and optimizing the complex and time-varying nature of user behavior across different platforms. Efficient algorithms for solving problems like the Maximum Independent Set in such models are crucial for a range of applications, including marketing, resource management, and information diffusion. Moreover, extending these algorithms to other dynamic networks, such as biological, transportation, financial, and communication systems, holds great potential for solving a broader range of real-world challenges. These extensions not only make the models more versatile but also pave the way for robust, scalable, and adaptive solutions in complex, evolving environments across diverse domains.

6 Conclusion

In this study, we have explored the concept of the maximum independent set within the context of stochastic multilayer graphs, which are crucial for modeling complex real-world networks. The identification of maximum independent sets provides profound insights into structural properties and dynamic behaviors, such as information diffusion and epidemic spread, particularly within social networks. By identifying influencers through independent sets, our research highlights the practical applications of this concept.

We introduced five novel algorithms based on learning automata for finding the maximum independent set in these stochastic multilayer graphs. Each learning automaton's decision-making process—whether to include a node in the independent set or not—was optimized to identify sets with the maximum expected value. Our simulations demonstrated that these learning automata-based algorithms require fewer samples compared to traditional methods, enhancing efficiency and accuracy. The superior performance of our algorithms underscores their potential in various applications, paving the way for further advancements in the study of stochastic multilayer graphs and their applications in understanding and managing complex networks.

7 Reference

[1] M. De Domenico *et al.*, "Mathematical Formulation of Multilayer Networks," *Phys. Rev. X*, vol. 3, no. 4, p. 041022, Dec. 2013, doi: 10.1103/PhysRevX.3.041022.

[2] N. Li and G. Chen, "Multi-layered friendship modeling for location-based mobile social networks," in *2009 6th Annual International Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous*, IEEE, 2009, pp. 1–10. Accessed: Oct. 06, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5326399/

[3] Y. Ge, L. Liu, X. Qiu, H. Song, Y. Wang, and K. Huang, "A framework of multilayer social networks for communication behavior with agent-based modeling," *SIMULATION*, vol. 89, no. 7, pp. 810–828, Jul. 2013, doi: 10.1177/0037549713477682.

[4] E. Vasilyeva *et al.*, "Multilayer representation of collaboration networks with higher-order interactions," *Scientific reports*, vol. 11, no. 1, p. 5666, 2021.

[5] M. A. Al-Garadi, K. D. Varathan, S. D. Ravana, E. Ahmed, and V. Chang, "Identifying the influential spreaders in multilayer interactions of online social networks," *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 5, pp. 2721–2735, 2016.

[6] M. E. Dickison, M. Magnani, and L. Rossi, *Multilayer social networks*. Cambridge University Press, 2016.

[7] İ. Türker and E. E. Sulak, "A multilayer network analysis of hashtags in twitter via co-occurrence and semantic links," *Int. J. Mod. Phys. B*, vol. 32, no. 04, p. 1850029, Feb. 2018, doi: 10.1142/S0217979218500297.

[8] J. Van Dijck, "Facebook and the engineering of connectivity: A multi-layered approach to social media platforms," *Convergence: The International Journal of Research into New Media Technologies*, vol. 19, no. 2, pp. 141–155, May 2013, doi: 10.1177/1354856512457548.

[9] S. Rani and M. Kumar, “Ranking community detection algorithms for complex social networks using multilayer network design approach,” *International Journal of Web Information Systems*, vol. 18, no. 5/6, pp. 310–341, 2022.

[10] A. Rezvanian and M. R. Meybodi, “Stochastic Graph as a Model for Social Networks,” *Computers in Human Behavior*, vol. 64, pp. 621–640, 2016.

[11] M. M. D. Khomami, M. R. Meybodi, and A. Rezvanian, “Exploring social networks through stochastic multilayer graph modeling,” *Chaos, Solitons & Fractals*, vol. 182, p. 114764, 2024.

[12] K. A. Jacobsen, M. G. Burch, J. H. Tien, and G. A. Rempała, “The large graph limit of a stochastic epidemic model on a dynamic multilayer network,” *Journal of Biological Dynamics*, vol. 12, no. 1, pp. 746–788, Jan. 2018, doi: 10.1080/17513758.2018.1515993.

[13] M. M. D. Khomami, N. Bagherpour, H. Sajedi, and M. R. Meybodi, “A new distributed learning automata based algorithm for maximum independent set problem,” in *2016 Artificial Intelligence and Robotics (IRANOPEN)*, IEEE, 2016, pp. 12–17.

[14] M. M. D. Khomami, A. Rezvanian, N. Bagherpour, and M. R. Meybodi, “Minimum positive influence dominating set and its application in influence maximization: a learning automata approach,” *Applied Intelligence*, vol. 48, no. 3, pp. 570–593, 2018.

[15] W. Chen, C. Castillo, and L. V. Lakshmanan, *Information and influence propagation in social networks*. Springer Nature, 2022.

[16] E. Muller and R. Peres, “The effect of social networks structure on innovation performance: A review and directions for research,” *International Journal of Research in Marketing*, vol. 36, no. 1, pp. 3–19, 2019.

[17] G. F. Lawler, *Introduction to stochastic processes*. Chapman and Hall/CRC, 2018.

[18] P. Berman and G. Schnitger, “On the complexity of approximating the independent set problem,” *Information and Computation*, vol. 96, no. 1, pp. 77–94, 1992.

[19] E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, “Generating All Maximal Independent Sets: NP-Hardness and Polynomial-Time Algorithms,” *SIAM J. Comput.*, vol. 9, no. 3, pp. 558–565, Aug. 1980, doi: 10.1137/0209042.

[20] R. M. Karp, “On the computational complexity of combinatorial problems,” *Networks*, vol. 5, no. 1, pp. 45–68, 1975.

[21] C. C. Aggarwal, J. B. Orlin, and R. P. Tai, “Optimized crossover for the independent set problem,” *Operations research*, vol. 45, no. 2, pp. 226–234, 1997.

[22] D. N. Gainanov, N. Mladenovic, V. Rasskazova, and D. Urosevic, “Heuristic algorithm for finding the maximum independent set with absolute estimate of the accuracy,” in *Proc. CEUR Workshop*, 2018, pp. 141–149.

[23] A. D. Plotnikov, “Heuristic algorithm for finding the maximum independent set,” *Cybernetics and Systems Analysis*, vol. 48, no. 5, pp. 673–680, 2012.

[24] B. Nogueira, R. G. S. Pinheiro, and A. Subramanian, “A hybrid iterated local search heuristic for the maximum weight independent set problem,” *Optim Lett*, vol. 12, no. 3, pp. 567–583, May 2018, doi: 10.1007/s11590-017-1128-7.

[25] D. Hespe, C. Schulz, and D. Strash, “Scalable kernelization for maximum independent sets,” *Journal of Experimental Algorithmics (JEA)*, vol. 24, pp. 1–22, 2019.

[26] C. Piao, W. Zheng, Y. Rong, and H. Cheng, “Maximizing the reduction ability for near-maximum independent set computation,” *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2466–2478, 2020.

[27] J. Dahlum, S. Lamm, P. Sanders, C. Schulz, D. Strash, and R. F. Werneck, “Accelerating local search for the maximum independent set problem,” in *International symposium on experimental algorithms*, Springer, 2016, pp. 118–133.

[28] A. Gellner, S. Lamm, C. Schulz, D. Strash, and B. Zaválnij, “Boosting Data Reduction for the Maximum Weight Independent Set Problem Using Increasing Transformations∗,” in *2021 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, SIAM, 2021, pp. 128–142.

[29] H. Pirim, “Finding Number of Clusters in a Gene Co-expression Network Using Independent Sets,” in *2013 International Conference on Social Computing*, Sep. 2013, pp. 836–839. doi: 10.1109/SocialCom.2013.125.

[30] M. Heal, “A Quadratic Programming Formulation to Find the Maximum Independent Set of Any Graph,” in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2016, pp. 1368–1370. doi: 10.1109/CSCI.2016.0255.

[31] Z. Cheng and J. Xiao, “Implementation of Maximum Independent Set Problem by Algorithmic Tile Self-Assembly,” in *2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications*, Sep. 2011, pp. 249–254. doi: 10.1109/BIC-TA.2011.36.

[32] M. Gencer and M. E. Berberler, “Solution to the maximum independent set problem with genetic algorithm,” in *2017 International Conference on Computer Science and Engineering (UBMK)*, Oct. 2017, pp. 734–738. doi: 10.1109/UBMK.2017.8093516.

[33] I. R. Alkhouri, G. K. Atia, and A. Velasquez, “A Differentiable Approach to the Maximum Independent Set Problem Using Graph-Based Neural Network Structures,” in *2022 IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP)*, Aug. 2022, pp. 1–6. doi: 10.1109/MLSP55214.2022.9943476.

[34] V. Maan, G. N. Purohit, and D. Sangwan, “An Independent Set Based Approach Using Random Degree Selection Distributed Algorithm for Graph Coloring,” in *2014 International Conference on Computational Intelligence and Communication Networks*, Nov. 2014, pp. 149–151. doi: 10.1109/CICN.2014.44.

[35] F.-T. Wang, S. Hatanaka, and R.-L. Wang, “A Stochastic Neuron Model for Finding a Near-Maximum Independent Set of a Circle Graph,” in *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, Jul. 2016, pp. 105–109. doi: 10.1109/ICISCE.2016.33.

[36] W. Zheng, Q. Wang, J. Xu Yu, H. Cheng, and L. Zou, “Efficient Computation of a Near-Maximum Independent Set over Evolving Graphs,” in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, Apr. 2018, pp. 869–880. doi: 10.1109/ICDE.2018.00083.

[37] Y. Takefuji, L.-L. Chen, K.-C. Lee, and J. Huffman, “Parallel algorithms for finding a near-maximum independent set of a circle graph,” *IEEE Transactions on Neural Networks*, vol. 1, no. 3, pp. 263–267, Sep. 1990, doi: 10.1109/72.80251.

[38] X. Liu, W. Zheng, Z. Chen, Z. He, and X. S. Wang, “Querying Maximum Quasi-independent Set by Pay-and-Recycle,” in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, May 2022, pp. 859–871. doi: 10.1109/ICDE53745.2022.00069.

[39] W. Zheng, C. Piao, H. Cheng, and J. X. Yu, “Computing a Near-Maximum Independent Set in Dynamic Graphs,” in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, Apr. 2019, pp. 76–87. doi: 10.1109/ICDE.2019.00016.

[40] W. Zheng, J. Gu, P. Peng, and J. X. Yu, “Efficient Weighted Independent Set Computation over Large Graphs,” in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, Apr. 2020, pp. 1970–1973. doi: 10.1109/ICDE48307.2020.00216.

[41] A. Köse and B. Özbek, “Resource Allocation for Underlaying Device-to-Device Communications Using Maximal Independent Sets and Knapsack Algorithm,” in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–5. doi: 10.1109/PIMRC.2018.8580784.

[42] Y. Li and Z. Xul, “An ant colony optimization heuristic for solving maximum independent set problems,” in *Proceedings Fifth International Conference on Computational Intelligence and Multimedia Applications. ICCIMA 2003*, IEEE, 2003, pp. 206–211.

[43] S. Bai, X. Che, X. Bai, and X. Wei, “Maximal Independent Sets in Heterogeneous Wireless Ad Hoc Networks,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2023–2033, Aug. 2016, doi: 10.1109/TMC.2015.2508805.

[44] E. Zhu, F. Jiang, C. Liu, and J. Xu, “Partition independent set and reduction-based approach for partition coloring problem,” *IEEE Transactions on Cybernetics*, vol. 52, no. 6, pp. 4960–4969, 2020.

[45] C. Joo, X. Lin, J. Ryu, and N. B. Shroff, “Distributed greedy approximation to maximum weighted independent set for scheduling with fading channels,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1476–1488, 2015.

[46] Z. Li, G. He, D. Xu, and S. Wang, “Evaluation of Centralized Reader Anti-Collision Protocols for Mobile RFID System Based on Maximum Independent Set: A Simulation Study,” *IEEE Access*, vol. 8, pp. 123381–123397, 2020.

[47] K. S. Narendra and M. A. Thathachar, *Learning automata: an introduction*. Courier Corporation, 2012.

[48] M. A. L. Thathachar and P. S. Sastry, *Networks of learning automata: Techniques for online stochastic optimization*. Springer, Boston, MA, 2004.

[49] F. Edition, A. Papoulis, and S. U. Pillai, *Probability, random variables, and stochastic processes*. McGraw-Hill Europe: New York, NY, USA, 2002. Accessed: Jul. 11, 2024. [Online]. Available: https://www.academia.edu/download/60930363/Probability__Random_Variables_and_Stochastic_Processes20191017-60780-1iogc7r.pdf

[50] J. A. Torkestani and M. R. Meybodi, “Finding minimum weight connected dominating set in stochastic graph based on learning automata,” *Information Sciences*, vol. 200, no. 1, pp. 57–77, 2012.

[51] S. Lakshmivarahan and M. A. L. Thathachar, "Bounds on the convergence probabilities of learning automata," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 6, no. 11, pp. 756–763, 1976.

[52] K. R. Hutson and D. R. Shier, "Minimum spanning trees in networks with varying edge weights," *Annals of Operations Research*, vol. 146, no. 1, pp. 3–18, 2006.

[53] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, no. 2, pp. 201–213, 2002.

Appendix

Table A1: Stochastic Multilayer Graph 1

Node	Layer	Weight	Probability
v_1	1	{2,8,12}	{0.90,0.08,0.02}
v_2	1	{10,24,35}	{0.85,0.12,0.03}
v_3	1	{6,18,24}	{0.88,0.10,0.02}
v_4	1	{12,22,30}	{0.85,0.11,0.04}
v_5	2	{17,35,50}	{0.75,0.20,0.05}
v_6	2	{3,7,10}	{0.68,0.25,0.07}
v_7	2	{4,19,15}	{0.75,0.14,0.11}
v_8	2	{5,10,12}	{0.65,0.23,0.12}

Table A2: Stochastic Multilayer Graph 2

Node	Layer	Weight	Probability
v_1	1	{2,8,12}	{0.90,0.08,0.02}
v_2	1	{10,24,35}	{0.85,0.12,0.03}
v_3	1	{6,18,24}	{0.88,0.10,0.02}
v_4	1	{12,22,30}	{0.85,0.11,0.04}
v_5	2	{17,35,50}	{0.75,0.20,0.05}
v_6	2	{3,7,10}	{0.68,0.25,0.07}
v_7	2	{4,19,15}	{0.75,0.14,0.11}
v_8	2	{5,10,12}	{0.65,0.23,0.12}
v_9	2	{10,19,24}	{0.80,0.14,0.06}

Table A3: Stochastic Multilayer Graph 3

Node	Layer	Weight	Probability
v_1	1	{2,8,12}	{0.90,0.08,0.02}
v_2	1	{10,24,35}	{0.85,0.12,0.03}
v_3	1	{6,18,24}	{0.88,0.10,0.02}
v_4	1	{12,22,30}	{0.85,0.11,0.04}
v_5	2	{17,35,50}	{0.75,0.20,0.05}
v_6	2	{3,7,10}	{0.68,0.25,0.07}
v_7	2	{4,19,15}	{0.75,0.14,0.11}
v_8	2	{5,10,12}	{0.65,0.23,0.12}
v_9	2	{10,19,24}	{0.80,0.14,0.06}
v_{10}	2	{18,27,36}	{0.94,0.05,0.01}

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: