Adaptive network approach to exploration-exploitation trade-off in reinforcement learning

Mohammadamin Moradi,[1] Zheng-Meng Zhai,[1] Shirin Panahi,[1] and Ying-Cheng Lai[1,2]
[1)]*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, Arizona 85287, USA*
[2)]*Department of Physics, Arizona State University, Tempe, Arizona 85287, USA*

(*Electronic mail: Ying-Cheng.Lai@asu.edu)

(Dated: 25 September 2024)

A foundational machine-learning architecture is reinforcement learning, where an outstanding problem is achieving an optimal balance between exploration and exploitation. Specifically, exploration enables the agents to discover optimal policies in unknown domains of the environment for gaining potentially large future rewards, while exploitation relies on the already acquired knowledge to maximize the immediate rewards. We articulate an approach to this problem, treating the dynamical process of reinforcement learning as a Markov decision process that can be modeled as a nondeterministic finite automaton and defining a subset of states in the automaton to represent the preference for exploring unknown domains of the environment. Exploration is prioritized by assigning higher transition probabilities to these states. We derive a mathematical framework to systematically balance exploration and exploitation by formulating it as a mixed integer programming problem to optimize the agent's actions and maximize the discovery of novel preferential states. Solving the MIP problem provides a trade-off point between exploiting known states and exploring unexplored regions. We validate the framework computationally with a benchmark system and argue that the articulated automaton is effectively an adaptive network with a time-varying connection matrix, where the states in the automaton are nodes and the transitions among the states represent the edges. The network is adaptive because the transition probabilities evolve over time. The established connection between the adaptive automaton arising from reinforcement learning and the adaptive network opens the door to applying theories of complex dynamical networks to address frontier problems in machine learning and artificial intelligence.

In the development of artificial intelligence, reinforcement learning (RL) has been playing a foundational role. For example, AlphaGo, a computer program that plays the board game Go, is based mostly on RL. In RL, agents learn optimal decision-making policies through trial and error. In particular, an RL agent interacts with the environment that it is in, receives feedback in the form of rewards or penalties, and updates its strategy accordingly. The dynamical process of RL has two major ingredients: exploration and exploitation. More specifically, exploration is necessary for discovering potentially beneficial actions or states that may yield higher rewards in the long run, but exploitation involves utilizing the already acquired knowledge to maximize the immediate rewards. While exploration may lead to states with better rewards, excessive exploration can impede the agent's ability to exploit the known profitable actions, in addition to the risk that no such states can be found. Further, excessive exploitation can lead to suboptimal policies due to insufficient exploration of the state space. For efficient and effective learning in RL, a balance between exploration and exploitation is essential. In spite of previous efforts, there is no definite solution to the exploration-exploitation balance problem. This article presents an approach to this problem based on the automata theory - a mathematical framework for treating systems with discrete states and transitions. The basic idea is that the dynamical process of RL can be viewed as a Markov decision process that can be represented as a nondeterministic finite automaton. Accordingly, a subset of states in the automaton can be defined to represent the preference for exploring unknown domains of the environment. The RL agent is encouraged to prioritize exploration when higher transition probabilities are assigned to these states, thereby realizing control of the dynamical behavior of the RL agent. A formalism is derived, providing a systematic way to balance exploration and exploitation. The framework is verified computationally through a benchmark system, where an optimal trade-off point is explicitly demonstrated. A key message pertinent to this Focus Issue is that the articulated automaton for achieving an optimal balance between exploration and exploitation can be viewed as an adaptive network with a time-varying connection matrix, where the states in the automaton are effectively nodes in the network and the transitions among the states are edges connecting the nodes. The transition probabilities evolve over time depending on factors such as the current state and action. While explicit equations describing the couplings among the nodes cannot be explicitly written down, they are implicitly defined by the interactions of the RL agent with the environment through actions. The present work establishes a natural "bridge" between a frontier problem in RL and adaptive networks, suggesting the possibility that some well developed theories in the field of complex dynamical networks may be exploited to benefit machine learning and artificial intelligence in terms of understanding and optimization.

I. INTRODUCTION

Complex systems in the real world are often described by interactive networks comprising interconnected dynamical units. Adaptivity or plasticity emerges as a predominant trait within such networks, where both the connectivity structure of the network and the dynamics of individual nodes evolve over time[1–16]. The adaptivity enables the network to respond to changing conditions, leading to emergent properties and phenomena that do not arise in static networks. Adaptive dynamical networks encompass a diverse spectrum of systems capable of adjusting their connectivity according to their dynamic states[17]. Indeed, incorporating adaptivity into static networks allows a wide range of systems across diverse domains to be studied, from biological neural circuits to social networks and beyond[15,16]. While static networks have gained significant attention over recent decades, it remains challenging to understand the holistic dynamical behaviors of adaptive networks due to the complex interplay between the network structure and dynamics.

In this paper, we aim to establish a connection or a "bridge" between adaptive networks and a basic and open problem in reinforcement learning (RL) - a fundamental machine-learning paradigm that has revolutionized artificial intelligence. In RL, agents learn optimal decision-making policies through trial and error, and RL algorithms learn by interacting with an environment, receiving feedback in the form of rewards or penalties, and updating their actions or behavior accordingly[18–20]. In spite of the widespread use of RL in science, engineering and technologies, a challenging problem is the exploration-exploitation trade-off[21] that arises from the need to balance between gathering new information about the environment (exploration) and exploiting the existing knowledge to maximize rewards (exploitation). Specifically, exploration is necessary to discover potentially beneficial actions or states that may yield higher rewards in the long run, but exploitation simply utilizes the already acquired knowledge to maximize the immediate reward. Striking a balance between exploration and exploitation is crucial for RL agents to learn efficiently, because overly excessive exploration can impede the agent's ability to exploit the known profitable actions while excessive exploitation can lead to suboptimal policies due to insufficient exploration of the state space[22–26].

There were previous works on exploration-exploitation trade-off in RL. A common approach to exploration in RL is the so-called ε-greedy strategy[19], where the agent selects the action with the highest estimated reward value with the probability ε and explores a random action with the probability $1-\varepsilon$. While this approach is simple to implement and provides a basic level of exploration, there is a lack of adaptability because the fixed exploration rate as determined by the value of ε does not account for changes in the environment or agent's learning progress, leading to either excessive or insufficient exploration. A widely used technique is the upper confidence bound (UCB) algorithm[27,28], which balances exploration and exploitation based on the uncertainty of the action values. UCB algorithms assign exploration bonuses to actions based on their uncertainty estimates, encouraging the agent to explore less certain actions. While UCB methods offer a principled approach to exploration, they often rely on strong assumptions about the reward distribution and may not scale well to large state spaces due to the computational complexity. Thompson sampling[29–31] is a Bayesian-based method that addresses exploration by maintaining a posterior distribution over the parameters of the RL model. Instead of relying on explicit uncertainty estimates, Thompson sampling selects actions probabilistically according to the posterior distribution. This approach naturally balances exploration and exploitation and has shown promising results in various contexts, but it can be computationally demanding, especially in complex environments with high-dimensional state and action spaces[32,33]. Another line of research focused on intrinsic motivation, which augments the external reward signal with additional intrinsic rewards based on the agent's curiosity or novelty of experiences. Curiosity-driven exploration methods, such as the intrinsic motivation module[34–36] or the curiosity reward approach[37–39], encourage the agent to explore novel or uncertain states, fostering a more diverse exploration behavior. However, these methods often require a careful tuning of hyperparameters and may introduce additional complexities such as the need for separate reward models. Finally, there is a growing recent interest in leveraging formal methods and automata theory to address the exploration challenge in RL[40–43]. Existing works in this area have primarily focused on certain specific aspects of exploration, such as option discovery or novelty detection. A systematic framework for balancing exploration and exploitation is still lacking[44].

With respect to complex dynamical systems, recent years have witnessed widespread adoption of machine learning including RL across various areas such as evolutionary game dynamics[45], control systems[46,47], epidemic spreading[48], crisis prediction[49–51], and artificial complex dynamical memory[52]. There are also recent efforts in developing adaptive networks that integrate RL with complex dynamical systems[45]. Our approach to addressing the exploration-exploitation balance problem is unique in that we exploit the principles of the automata theory[53,54] *from the point of view of adaptive networks*. In particular, the theory of automata provides a mathematical framework for modeling and analyzing systems with discrete states and transitions. By modeling the RL problem as a Markov Decision Process (MDP) and representing it as a nondeterministic finite automaton (NDFA), we arrive at a formalism to explicitly capture and manipulate the exploration preference of the RL agents. Our approach entails defining a subset of states in the NDFA that represent the preference for exploring unknown domains of the environment. These states are characterized by higher transition probabilities, thereby encouraging the agent to prioritize exploration. By assigning appropriate transition probabilities within this subset, the exploration behavior of the RL agent can be controlled. The formalism provides a means to systematically balance exploration and exploitation in RL and offers a unique perspective for understanding the dynamics of exploration within the RL framework, leading to a better exploration-exploitation balance. We computationally validate our framework in terms of exploration efficiency and the overall learning performance.

In Sec. II, we present our formulation of the adaptive network approach to exploration-exploitation balance in RL by introducing the NDFA, justifying it as an adaptive network, and formulating the satisfaction of the exploration preference as a mixed integer programming (MIP) problem. In Sec. III, we study a concrete example to illustrate the preference induced MDP implementation and formulation of the MIP. In Sec. IV, we demonstrate that our automata theory-based framework can lead to an optimal exploration-exploitation balance in RL through a benchmark system: the movements of a taxi on a spatial grid. In Sec. V (Discussion), we point out the limitations of the framework and offer a future perspective. In Appendix A, we present four potential application examples: robotic maintenance in a power grid, sensor calibration, circuit design optimization, and control system optimization, and describe how each problem can be formulated using our adaptive automaton or adaptive network framework.

II. ADAPTIVE NETWORK APPROACH TO EXPLORATION-EXPLOITATION BALANCE IN REINFORCEMENT LEARNING: THEORETICAL FORMULATION

We argue and show that the exploration-exploitation balance problem can be formulated as an MIP problem based on linear temporal-logic constraints. In particular, a fundamental concept in computer science is abstract machines capable of transitioning between different states based on inputs. These machines, known as automata, provide a mathematical framework for modeling and analyzing systems with discrete states and transitions. Further, temporal logic is a formal logic used to reason about events and their relationships over time. It models and analyzes the dynamic behavior of RL training process, verifies control systems, predicts outcomes and identifies vulnerabilities. In our work, the theory of automata is used as a foundational tool for modeling the exploration-exploitation balance in RL. In the following, we introduce finite automata as a class of adaptive networks and show how linear temporal-logic formulas quantify preferences to address the exploration-exploitation balance problem.

A. Non-deterministic finite automaton (NDFA)

A nondeterministic finite state machine (NDFA) is a theoretical model used in computer science to recognize patterns within input strings of symbols. Unlike a deterministic finite state machine (DFA) where each state and input symbol determines a single, unique next state, an NDFA can transition to multiple possible states for a given input. This allows the NDFA to explore multiple potential state sequences simultaneously. When given an input string of symbols, the NDFA processes each symbol by moving through its states, which are connected by transitions. These transitions are defined based on the symbols of the string, but the key feature of an NDFA is that, for any given state and symbol, there may be several possible next states, or even none. In some cases, an NDFA can also transition to a new state without consuming any input symbols. The NDFA accepts the string if there exists at least one sequence of transitions leading from the initial state to an accepting (or final) state that corresponds to the entire input string. If no such sequence exists, the NDFA rejects the string. This nondeterminism means that the automaton does not follow a single path; instead, it explores multiple paths simultaneously and determines if any path leads to success. NDFAs are particularly useful in theory because they are conceptually simpler and often easier to design than DFAs. Importantly, for any NDFA, there exists an equivalent DFA that recognizes the same language, even though the DFA might have more states[53,55]. Based on the NDFA, preferences over accepting conditions can be modeled[56]. A DFA is a five-tuple

$$\langle \tilde{S}, \Sigma, \delta, \tilde{s}_0, \phi \rangle,$$

where $\tilde{S}$ represents the set of automaton states, Σ is the set of possible automaton symbols/actions, δ is the transition function, $\tilde{s}_0$ is the initial automaton state, and ϕ represents the preference formula(s). To incorporate preferences into an MDP, we define the *preference induced MDP*, which is a 4-tuple:

$$\langle \tilde{S} \times S, A, d, \Delta \rangle,$$

where S denotes the set of MDP states, A is the set of MDP actions, and d is the probability of initial state distribution. The transition function Δ is defined as

$$\Delta((\tilde{s}', s')|(\tilde{s}, s), a) = P(s'|s, a) * \mathbf{1}_{\{\tilde{s}'\}}\{\delta(L(s'), \tilde{s})\}, \quad (1)$$

which means that from MDP state s and automaton state $\tilde{s}$, under action a, the probability of going to MDP state s' and automaton state $\tilde{s}'$ is equal to the sum of the probabilities of going into s' from s taking action a for all possible transitions. In our problem, $\tilde{S} = S$ and A only include one action: mandatory movement, because in each step of RL, an agent must take an action. The distribution of initial state d is uniform. Without any loss of generality, given that $\tilde{S}$ is equivalent to S, we proceed to use "s" instead of "$(\tilde{s}, s)$" for simplicity. In Eq. (1), L is the labeling function that maps MDP states to their correspondent automaton actions, which is the identity map in our case.

To explain the definition (1), we consider an example: a Taxi. As illustrated in Fig. 1(a), a taxi is situated within a 2 by 2 grid environment and is engaged in the pursuit of accomplishing specific objectives using four actions: going up, down, left, and right. Assume the taxi driver has a deterministic policy π derived from previous training as going up in states $\{0\}$, $\{1\}$, and $\{3\}$, and going left in state $\{2\}$. Further, assume that the taxi employs the ε-greedy method for exploration (with the probability $1-\varepsilon$) and exploitation (with the probability ε). Table I lists the transition matrix for this preference-induced MDP. Taking into account the training procedure of the RL agent as well as the objective of exploring unexplored territories while simultaneously reaching the goal, we establish the following preferences:

- *PA*: getting to the star cell is the final goal;

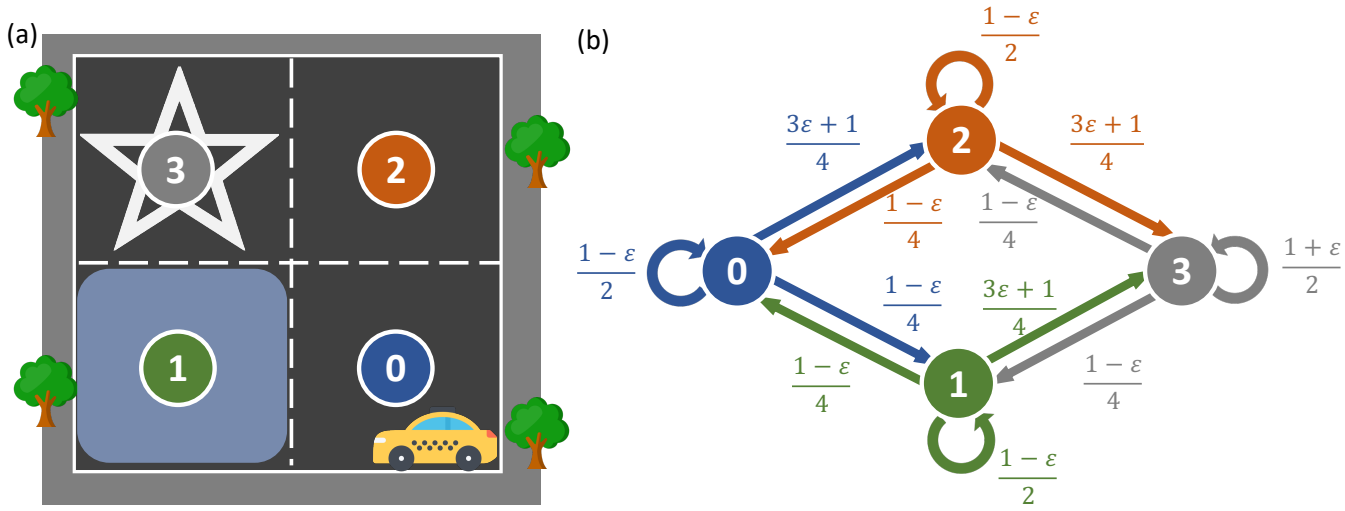

FIG. 1. An example illustrating exploration-exploitation in RL and the emergence of an automaton. (a) Within a bounded 2 by 2 grid environment, a taxi is positioned and tasked with achieving predefined objectives through a set of four distinct actions: moving upwards, downwards, leftwards, and rightwards. The goal is to successfully navigate towards the star cell. Within the grid, the highlighted blue cell indicates the unexplored cell to be explored during the training phase. (b) The resulting automaton based on the probabilities of different transitions using the ε-greedy approach.

- *PB*: exploring the unexplored area first is preferred to getting to the star cell;
- *PC*: getting to the goal is preferred to getting to any other cell except the unexplored cells;
- *PD*: getting to the unexplored cell is preferred to getting to the star cell and getting to the star cell is preferred to getting to the other cells;
- *PE*: exploring the unexplored area first is preferred to getting to the other cells.

The resulting automaton is illustrated in Fig. 1(b). The automaton so constructed has four states, where the preferences can be implemented as the following logic formulas:

- $\phi_{PA}: F\{3\}$
- $\phi_{PB}: \{1\} \succeq \{3\}$
- $\phi_{PC}: \{3\} \succeq \{0,2\}$
- $\phi_{PD}: \{1\} \succeq \{3\} \succeq \{0,2\}$
- $\phi_{PE}: \{1\} \succeq \{0,2,3\}$

Note that $F\psi$ ("eventually" operator) denotes that ψ occurs at some point in the future and $A \succeq B$ is read as "A is preferred to B." Transforming preferences into temporal logic formulas in the form $\{P\} \preceq \{P'\}$ enables us to formulate an MIP problem to find an optimal policy to maximally satisfy the exploration-exploitation balance, where exploration is incorporated into the problem by employing ϕ_{PE} as the preference, while the exploitation requirement is fulfilled by establishing a criterion based on the frequency of the agent reaching the star cell during training.

A concrete example of an automaton is presented in Sec. III.

TABLE I. Transition matrix ($\Delta(s'|s,a)$) based on the automaton in Fig. 1(b)

Current\Next	{0}	{1}	{2}	{3}
{0}	$\frac{1}{2}(1-\varepsilon)$	$\frac{1}{4}(1-\varepsilon)$	$\frac{1}{4}(3\varepsilon+1)$	0
{1}	$\frac{1}{4}(1-\varepsilon)$	$\frac{1}{2}(1-\varepsilon)$	0	$\frac{1}{4}(3\varepsilon+1)$
{2}	$\frac{1}{4}(1-\varepsilon)$	0	$\frac{1}{2}(1-\varepsilon)$	$\frac{1}{4}(3\varepsilon+1)$
{3}	0	$\frac{1}{4}(1-\varepsilon)$	$\frac{1}{4}(1-\varepsilon)$	$\frac{1}{2}(1+\varepsilon)$

B. Automaton as an adaptive network

Adaptive networks represent a frontier in the study of complex systems, offering a dynamic framework that can capture the complicated interplay between structure and function[1–16]. In network science, adaptivity is referred to as the dynamic evolution of network topology, where nodes and edges adjust their connections and strengths based on the system's dynamics or external stimuli[57]. Conventionally, a general class of N-dimensional adaptive dynamical networks can be defined by the following set of coupled differential equations:

$$\dot{\boldsymbol{x}}_i(t) = \boldsymbol{F}(\boldsymbol{x}_i(t)) + \sigma \sum_j k_{ij}(t) \boldsymbol{H}[\boldsymbol{x}_i(t), \boldsymbol{x}_j(t-\tau)], \quad (2)$$

$$\dot{k}_{ij}(t) = G[k_{ij}(t), \boldsymbol{x}_i(t), \boldsymbol{x}_j(t), \eta], \quad (3)$$

where the dynamical function F governs the dynamics of each individual node described by an m-dimensional state vector x_i, H is the coupling function, σ is the overall coupling strength, and k_{ij} are the elements of the dynamical adjacency matrix $\mathscr{K}$ that describes the time dependence of the topological connectivity of the network. The entries of this weighted matrix evolve over time according to a general adaptation evolution function G, and η in G is the adaptation parameter.

The dynamic topology inherent in adaptive networks engenders a spectrum of collective behaviors such as complete or cluster synchronization[58,59], where complete synchronization denotes a state wherein all nodes within the network achieve identical dynamical states, irrespective of their initial conditions or interaction complexity. This global coherence facilitates efficient information propagation and system-wide coordination. The master stability function approach[60–65] can be used to study the stability of the complete synchronous solution in adaptive networks[66,67], when the mathematical functions governing the network dynamics are available. However, in real-world applications, the exact equations governing the system dynamics are unknown. In fact, real-world systems often can be viewed as a black box - a situation similar to nondeterministic automata, where various actions evoke different responses, rendering inapplicable the master stability function approach.

In formulating an adaptive automaton as an adaptive network with a time-varying transition matrix, even though the exact equations of the system are not available, they are implicitly defined as the interactions of the agent with the environment through actions. The states and transitions of the automaton can thus be interpreted within the framework of a dynamic network describing the RL exploration-exploitation generative procedure, where the states in the automaton are treated as nodes in a network and the transitions between the states are viewed as edges that effectively connect nodes in the network. The adaptive nature of the problem is encapsulated by the dynamical transition matrix with time-varying transition probabilities. These probabilities evolve over time depending on factors such as the current state and the action taken.

C. Formulating the satisfaction of the preferences as an MIP problem

In an MIP problem[68], some variables of the system to be optimized are integers with a linear objective function, subject to linear constraints. In our work, the objective function is the value of the preference satisfaction. The definition of the value of the preference satisfaction (v_{ps}) is closely related to the probability of occurrence of the corresponding preference, where v_{ps} for a preference formula $X_0 \preceq X_1 \preceq ... \preceq X_n$ is defined as[56] $P(X_i)$ if there exists some i such that $P(X_i) \geq P(X_{i-1})$ while for all $k \geq i$, $P(X_k) < P(X_{k-1})$ holds and is zero otherwise. The definition of v_{ps} can be illustrated using the example in Fig. 1. Assume that for the preference formula ϕ_{PD}, the two derived policy samples induce the probabilities as listed in Table II. From the definition of v_{ps}, policy π_1 satisfies the preference ϕ_{PD} by 80% whereas policy π_2 satisfies the preference ϕ_{PD} by 70%. Note that v_{ps} is not necessarily equal to the probability of most preferred state.

To obtain the constraints for the MIP problem, we employ a supporting variable $y(t,s,a)$, defined as the probability of visiting the state s at time t and taking action a, so $y(T,P)$ is the probability of the automaton being at state P at time $t = T$ (regarded as the ending point). Using the supporting variable and from the definition of v_{ps} for the preference formula $\{P\} \preceq \{P'\}$, we have the first set of MIP constraints as

TABLE II. Probabilities of automaton ending in specific states for two random policies

Automaton State\Policy	P_{π_1}	P_{π_1}
{1}	0.1	0.7
{3}	0.8	0.2
{0,2}	0.1	0.1

$$0 \leq v_{ps} \leq B, \tag{4}$$

$$B - 1 \leq v_{ps} - y(T,P') \leq 0, \tag{5}$$

$$B(1+\xi) + 1 \leq y(T,P') - y(T,P) \leq B(1+\xi) - \xi, \tag{6}$$

$$B \text{ is a binary}, \tag{7}$$

$$\text{all } y \text{ are non-negative}, \tag{8}$$

where ξ is a small positive number and B is a binary variable: it is one if there exists a policy that satisfies the preference and zero otherwise. From Eqs. (4) and (5), it can be seen that if there exists no such policy satisfying the preference, then $v_{ps} = 0$, otherwise its value will be between 0 and 1 and is equal to $y(T,P')$. We enforce the events' probabilities and their difference to be between 0 and 1 using Eqs. (6) and (8). The second set of the MIP constraints is responsible for maintaining the consistence between the supporting variable $y(t,s,a)$ and MDP elements d and Δ. For all possible states and for all time steps, we have

$$\sum_{a \in A} y(0,s,a) = d(s), \tag{9}$$

$$\sum_{a \in A} y(t,s',a) = \sum_{a \in A} \sum_{s \in S} \Delta(s'|s,a) y(t-1,s,a). \tag{10}$$

Equation (9) ensures that the probabilities denoted by the supporting variable y at time $t = 0$ are consistent with the distribution of initial state d. Equation (10) asserts that the probability of getting to a state at time t is equal to the sum of probabilities that other possible states take the specific action to reach the target state at time $t-1$. Taking into consideration these constraints, we formulate the maximum preference satisfaction problem for the preference formula $\{P\} \preceq \{P'\}$ as

$$\max_{B,y,v_{ps}} v_{ps} \text{ subject to Eqs. } (4-10), \tag{11}$$

where B, y and v_{ps} are the optimization variables and v_{ps} is the objective function. Equation (11) represents an MIP problem because it includes an integer constraint [Eq. (7)], otherwise it is a standard linear programming problem. By resolving the optimization problem outlined in Eq. (11), we obtain the variable v_{ps}, which serves as a decisive criterion for effectively exploring the uncharted cells within the grid environment. It is worth noting that the feasibility of the optimization problem introduced in our study is contingent upon various factors that encompass the inherent dynamics of the Markov decision process, the level of stringency inherent within the preference formula, and even the choice of solver employed to tackle the

optimization problem. As a result, the attainability of a feasible solution can be influenced by these considerations. In the following, we present a concrete example of formulating a preference induced MDP into MIP and obtain the solution.

III. A CONCRETE EXAMPLE OF AUTOMATON

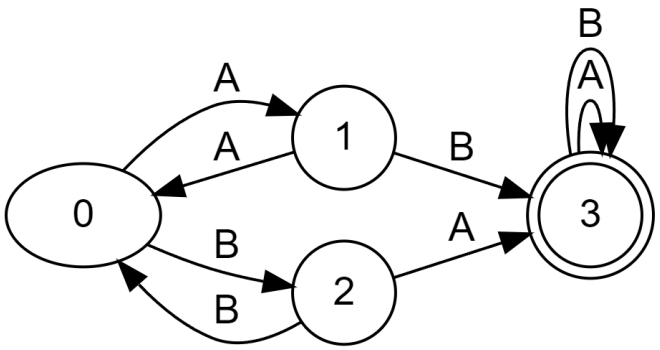

FIG. 2. A concrete example of automaton. There are four states with #0 being the starting state and #3 being the final state. The preference $p := \{1\} \succeq \{2\}$ stipulates that ending in state #1 is preferred to ending in state #2. As indicated in Tab. IV, using the derived policy, 76.89% of the time the system will end in state #1 compared with ending in state #2 which takes place 0.48% of the time, indicating that the preference is optimally satisfied.

We study a concrete example to illustrate the preference induced MDP implementation and formulation of the MIP. The specific finite automaton is illustrated in Fig. 2. The setting is

- Preference: $p := (P' =)\{1\} \succeq (P =)\{2\}$
- Episode length: $T = 3$
- MDP states:

$$S = \{[1,1] := 0, [1,0] := 1, [0,1] := 2, [0,0] := 3\}$$

- Automaton states: $\tilde{S} = \{0,1,2,3\}$
- Actions: $AS = \{A, B\}$
- Initial state-pair probability distribution:

$$d(\tilde{s},s) = \begin{cases} 0.9 & \text{if } (\tilde{s},s) = (0,0) \\ 0.05 & \text{if } (\tilde{s},s) = (1,1) \\ 0.05 & \text{if } (\tilde{s},s) = (2,2) \\ 0 & \text{else} \end{cases}$$

TABLE III. Definition of supporting variables $y(t,(\tilde{s},s),a)$

t	$(\tilde{s},s)$	A	B
t=0	(0,0)	y_1	y_2
t=0	(1,1)	y_3	y_4
t=0	(2,2)	y_5	y_6
t=0	(3,3)	y_7	y_8
t=1	(0,0)	y_9	y_{10}
t=1	(1,1)	y_{11}	y_{12}
t=1	(2,2)	y_{13}	y_{14}
t=1	(3,3)	y_{15}	y_{16}
t=2	(0,0)	y_{17}	y_{18}
t=2	(1,1)	y_{19}	y_{20}
t=2	(2,2)	y_{21}	y_{22}
t=2	(3,3)	y_{23}	y_{24}
t=3	(0,0)	y_{25}	y_{26}
t=3	(1,1)	y_{27}	y_{28}
t=3	(2,2)	y_{29}	y_{30}
t=3	(3,3)	y_{31}	y_{32}

- State-pair transition probability distribution:

$$\begin{aligned}
&\Delta((1,1)|(0,0),A) = 0.8 \\
&\Delta((2,2)|(0,0),A) = 0.1 \\
&\Delta((3,3)|(0,0),A) = 0.05 \\
&\Delta((0,0)|(0,0),A) = 0.05 \\
&\Delta((1,1)|(0,0),B) = 0.1 \\
&\Delta((2,2)|(0,0),B) = 0.8 \\
&\Delta((3,3)|(0,0),B) = 0.05 \\
&\Delta((0,0)|(0,0),B) = 0.05 \\
&\Delta((0,0)|(1,1),A) = 0.05 \\
&\Delta((1,1)|(1,1),A) = 0.95 \\
&\Delta((3,3)|(1,1),B) = 1 \\
&\Delta((3,3)|(2,2),A) = 1 \\
&\Delta((0,0)|(2,2),B) = 0.05 \\
&\Delta((2,2)|(2,2),B) = 0.95 \\
&\Delta((3,3)|(3,3),A) = 1 \\
&\Delta((3,3)|(3,3),B) = 1
\end{aligned}$$

The supporting variable $y(t,(\tilde{s},s),a)$ is defined as the probability of visiting the state pair $(\tilde{s},s)$ at time t and taking action a. We use the variables listed in Tab. III with the purpose of keeping the equations concise. For example, y_{11} is the probability of being in the state pair $(1,1)$ at time $t = 1$ under action A. With these definitions and the initial state-pair probability distribution ($d(\tilde{s},s)$), we construct the following constraints from Eq. (9):

$$y_1 + y_2 = 0.9, \tag{12}$$
$$y_3 + y_4 = 0.05, \tag{13}$$
$$y_5 + y_6 = 0.05, \tag{14}$$
$$y_7 + y_8 = 0. \tag{15}$$

Moreover, using the state-pair transition probability distribution defined above $[\Delta((\tilde{s}',s')|(\tilde{s},s),a)]$, we obtain the follow-

ing constraints from Eq. (10):

$$y_9 + y_{10} = 0.05y_1 + 0.05y_2 + 0.05y_6 + 0.05y_3, \quad (16)$$
$$y_{11} + y_{12} = 0.8y_1 + 0.1y_2 + 0.95y_3, \quad (17)$$
$$y_{13} + y_{14} = 0.1y_1 + 0.8y_2 + 0.95y_6, \quad (18)$$
$$y_{15} + y_{16} = 0.05y_1 + 0.05y_2 + y_4 + y_5 + y_7 + y_8, \quad (19)$$
$$y_{17} + y_{18} = 0.05y_9 + 0.05y_{10} + 0.05y_{14} + 0.05y_{11}, \quad (20)$$
$$y_{19} + y_{20} = 0.8y_9 + 0.1y_{10} + 0.95y_{11}, \quad (21)$$
$$y_{21} + y_{22} = 0.1y_9 + 0.8y_{10} + 0.95y_{14}, \quad (22)$$
$$y_{23} + y_{24} = 0.05y_9 + 0.05y_{10} + y_{12} + y_{13} + y_{15} + y_{16}, \quad (23)$$
$$y_{25} + y_{26} = 0.05y_{17} + 0.05_{18} + 0.05_{22} + 0.05y_{19}, \quad (24)$$
$$y_{27} + y_{28} = 0.8y_{17} + 0.1y_{18} + 0.95y_{19}, \quad (25)$$
$$y_{29} + y_{30} = 0.1y_{17} + 0.8y_{18} + 0.95y_{22}, \quad (26)$$
$$y_{31} + y_{32} = 0.05y_{17} + 0.05y_{18} + y_{20} + y_{21} + y_{23} + y_{24}. \quad (27)$$

The following relations are useful:

$$y(T,P) = y_{29} + y_{30},$$
$$y(T,P') = y_{27} + y_{28}.$$

Equations (4-8) then lead to the following constraints:

$$0 \le v_{PS} \le B, \quad (28)$$
$$B - 1 \le v_{PS} - y_{27} - y_{28} \le 0, \quad (29)$$
$$B(1+\varepsilon) + 1 \le y_{27} + y_{28} - y_{29} - y_{30} \le B(1+\varepsilon) - \varepsilon, \quad (30)$$
$$B \text{ is a binary}, \quad (31)$$
$$\text{all } y \text{ are non-negative}. \quad (32)$$

The standard form of the MIP optimization problem for this example is

$$\max_{B,y,v_{PS}} v_{PS} \text{ subject to Eqs. } (12-31).$$

Using a standard equation solver, such as intlinprog in MATLAB, we obtain the solution to this optimization problem as $v_{PS} = 0.7689$, $B = 1$, with the values of the y parameters listed in Table IV. This solution describes the best possible way to satisfy the preference $p := \{1\} \succeq \{2\}$. Following this policy for each state pair at each time step, it can be ensured that 76.89% of the time the preference will be satisfied - the highest satisfaction possible for this problem.

For each value of the exploration parameter ε ranging from 0 to 1, we follow a systematic procedure to formulate and solve the MIP problem based on the preference-induced MDP (inferred from the RL environment) in three steps:

Step 1: At each ε value, we formulate the MIP problem by incorporating the preference-induced MDP. Once the MIP problem is set up, the objective is to derive the value of preference satisfaction, v_{PS}, which quantifies how well the agent explores the under-explored environment. This value serves as an *exploration indicator*, reflecting how successfully the agent balances exploration according to the defined preferences within the MDP. The higher the v_{PS}, the better the agent is at exploring unknown states.

TABLE IV. Solution for the MIP problem of *Example 1* $v_{PS} = 0.7689$ and $B = 1$.

t	$(\tilde{s},s)$	A	B
t=0	(0,0)	$y_1 = 0.9$	$y_2 = 0$
t=0	(1,1)	$y_3 = 0.05$	$y_4 = 0$
t=0	(2,2)	$y_5 = 0$	$y_6 = 0.05$
t=0	(3,3)	$y_7 = 0$	$y_8 = 0$
t=1	(0,0)	$y_9 = 0.05$	$y_{10} = 0$
t=1	(1,1)	$y_{11} = 0.7675$	$y_{12} = 0$
t=1	(2,2)	$y_{13} = 0$	$y_{14} = 0.1375$
t=1	(3,3)	$y_{15} = 0$	$y_{16} = 0.0450$
t=2	(0,0)	$y_{17} = 0.0478$	$y_{18} = 0$
t=2	(1,1)	$y_{19} = 0.7691$	$y_{20} = 0$
t=2	(2,2)	$y_{21} = 0.1356$	$y_{22} = 0$
t=2	(3,3)	$y_{23} = 0$	$y_{24} = 0.0475$
t=3	(0,0)	$y_{25} = 0$	$y_{26} = 0.0408$
t=3	(1,1)	$y_{27} = 0$	$y_{28} = 0.7689$
t=3	(2,2)	$y_{29} = 0$	$y_{30} = 0.0048$
t=3	(3,3)	$y_{31} = 0$	$y_{32} = 0.1855$

Step 2: In parallel, we define the criterion for measuring the agent's ability to achieve its primary goal (i.e., exploitation). The exploitation indicator v_{GS} quantifies the frequency with which the agent reaches the desired goal states during the training episode. This is mathematically defined by the sum of the probabilities of visiting goal states at time t across all actions a:

$$v_{GS} = \sum_{t=1}^{T} \sum_{s \in S_{goal}} \sum_{a \in A} y(t,s,a) \quad (33)$$

where $y(t,s,a)$ denotes the probability of being in state s and taking action a at time t. This value serves as an *exploitation indicator*, where a higher v_{GS} reflects more frequent achievement of the task or goal.

Step 3: To determine the optimal balance between exploration and exploitation, we track the trajectories of both v_{PS} (exploration) and v_{GS} (exploitation) as ε varies from 0 to 1. The goal is to identify the *trade-off point* where the two trajectories intersect, representing the optimal value of ε that balances both exploration and exploitation. This intersection point provides critical insight into how much exploration is necessary before the agent should start exploiting the learned knowledge to maximize immediate rewards.

This step-by-step approach ensures that the exploration-exploitation trade-off is systematically analyzed, with the MIP formulation and solution serving as the foundation for deriving key performance metrics for both exploration and exploitation.

IV. DEMONSTRATION OF OPTIMAL EXPLORATION-EXPLOITATION BALANCE

To test and validate our automata theory-based framework to achieve an optimal exploration-exploitation balance in RL,

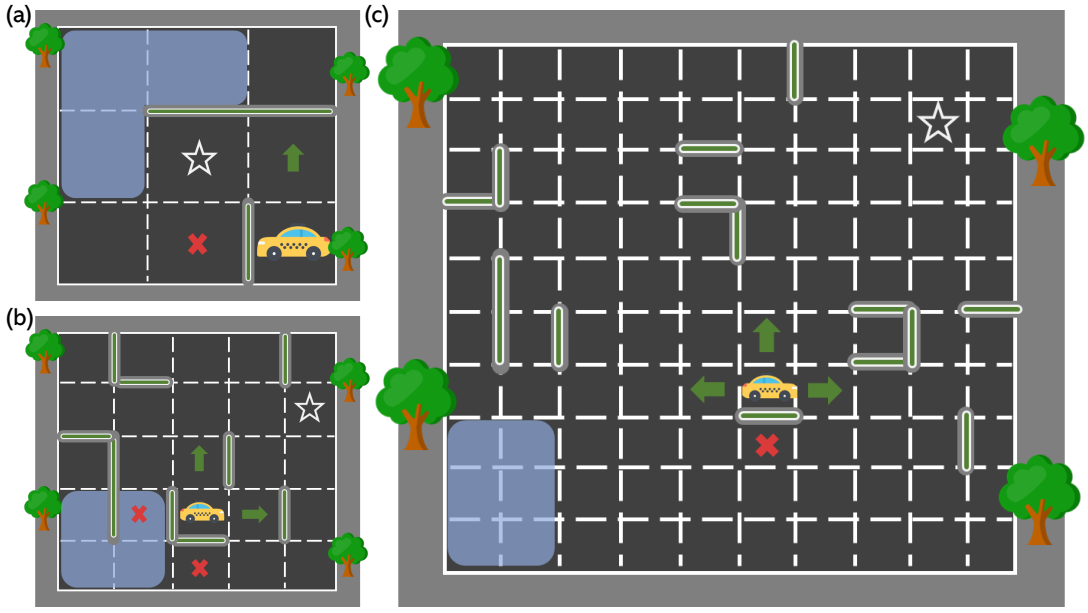

FIG. 3. Experimental RL environments. Shown are: (a) 3×3, (b) 5×5, and (c) 10×10 grids. Random walls are positioned at various locations. The RL goal for the taxi agent is to get to the star cell. The unexplored cells are highlighted with blue. Three elements are subject to random generation: the placement of walls within the grid, the initial policy of the RL agent, and the probability distribution of the initial position of the taxi. The random values are generated using a designated random seed (7, 8, and 0 for the 3×3, 5×5, and 10×10 grids, respectively).

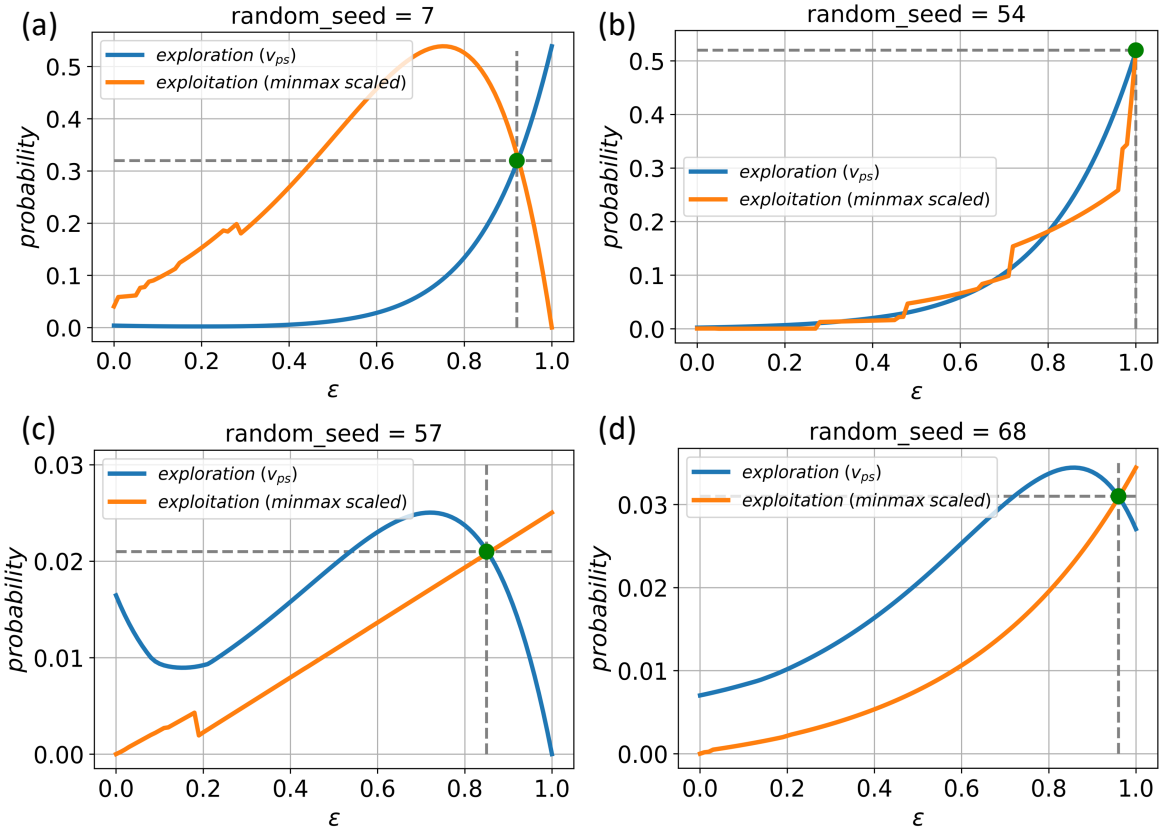

FIG. 4. Determining the trade-off point. (a-d) The intersecting point of the trajectory of the value of preference satisfaction (v_{ps}) with the trajectory of goal achievement (v_{gs}) for different values of ε, for the 3×3 grid with four different random seeds, respectively.

we conduct experiments in grid environments of three representative sizes: 3×3, 5×5, and 10×10, as shown in Fig. 3.

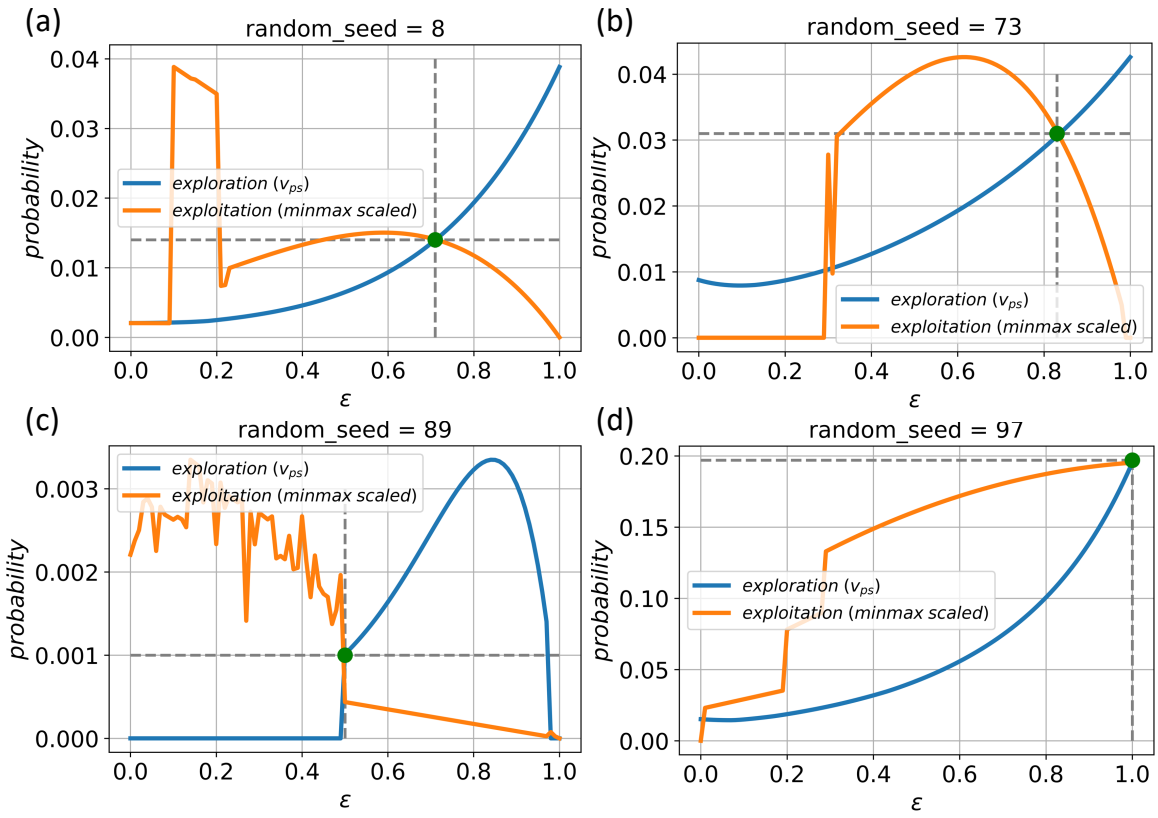

FIG. 5. Determining the trade-off point for the 5×5 grid. (a-d) The intersecting point of the trajectory of the value of preference satisfaction (v_{ps}) with the trajectory of goal achievement (v_{gs}) for different values of ε, for the 5×5 grid with four different random seeds, respectively.

Figures 4, 5 and 6 show the respective results for the three grid environments. The simulations are performed on computers each with GPU GeForce RTX 4090 and CPU 13th Gen Intel(R) Core(TM) i9-13900KS, using Python (SpyderIDE). We use the *optimize.milp* module of the *scipy* package to solve the MIP problems. For the 10×10 grid, on average it takes 1.2 seconds to generate and solve the MIP.

The simulation setting described in Sec. II is applied to all experiments. In particular, to introduce a higher level of complexity to the environment, random walls are positioned at various locations. For example, when a wall is situated to the right of the taxi and the taxi intends to move in that direction, it will remain in its current cell due to the obstruction. In the simulations, three elements are subject to random generation: the placement of walls within the grid, the initial policy of the RL agent, and the probability distribution governing the initial position of the taxi. The random values are generated using a designated random seed, each engendering distinct scenarios and facilitating analysis of the trade-off across diverse situations. The random seeds are themselves randomly generated from a uniform distribution. The criterion for quantifying the frequency of goal attainment by the agent during the training phase is indicated by Eq. 33.

We focus on determining the exploration-exploitation trade-off point by intersecting the trajectory of value of preference satisfaction (v_{ps}) and the trajectory of goal achievement (v_{gs}) for different values of ε. To evaluate the exploration behavior of the RL agents, we derive the trajectory value of preference satisfaction, which quantifies the degree to which the exploration of unexplored cells preference defined by the automaton is satisfied. Note that higher values of ε indicate a stronger inclination towards exploitation. We compare this trajectory with the trajectory of achieving the RL goal, which represents the agent's ability to accomplish the task at hand. By examining the intersection point of these trajectories for different ε values, we identified the optimal trade-off point for the ε value that balances exploration and exploitation. Figures 4, 5, and 6 illustrate the results for the 3×3, 5×5, and 10×10 grids, respectively, demonstrating the effectiveness of our framework in achieving the exploration-exploitation balance.

Through simulations, we identify certain instances where an increase in ε, signifying a greater emphasis on exploitation, correlates with a simultaneous increase in the trajectory value of preference satisfaction. This indicates a heightened level of exploration within previously unexplored cells. The finding suggests that in certain scenarios, the initial policy and positioning of walls pose challenges for the agent to effectively explore uncharted areas. As a result, even following a suboptimal policy can yield higher satisfaction of preferences.

The ε trade-off point that strikes an optimal balance between exploration and exploitation can be found by identifying the intersection point between the trajectories of preference satisfaction and goal achievement. This point represents

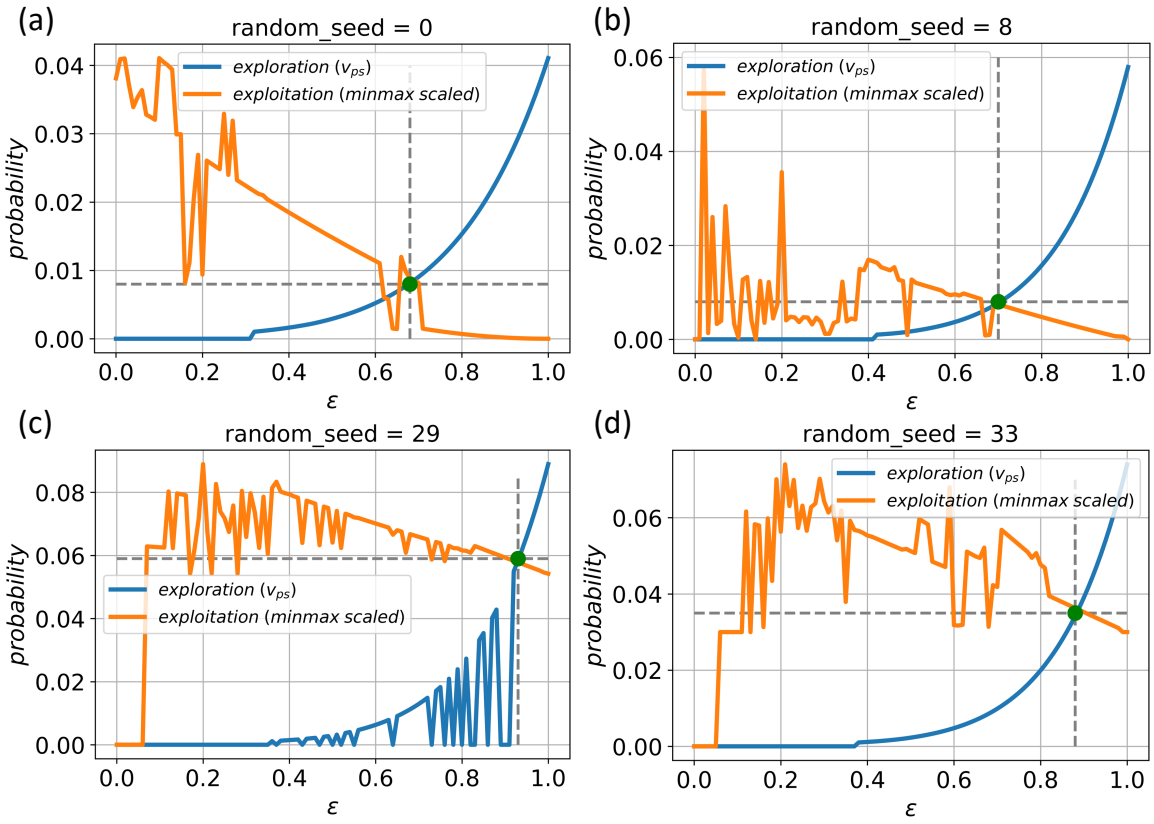

FIG. 6. Determining the trade-off point for the 10×10 grid. (a-d) The intersecting point of the trajectory of the value of preference satisfaction (v_{ps}) with the trajectory of goal achievement (v_{gs}) for different values of ε, for the 10×10 grid with four different random seeds, respectively.

the ideal value of ε that maximizes both the agent's exploration and its ability to achieve the desired goal. Our approach provides a systematic and principled methodology for finding this trade-off point. We find that the optimal trade-off point is influenced by the grid environment size. In a smaller grid environment, such as the 3×3 grid, the trade-off point tends to have a higher value of ε, indicating a greater emphasis on exploration. As the grid size increases, the trade-off point often shifts towards lower values of ε, indicating a stronger focus on exploitation. This observation suggests an effect of the complexity and size of the environment on the optimal balance between exploration and exploitation.

The simulations demonstrate the efficacy of our automata theory-based approach to achieving an optimal exploration-exploitation balance in RL, where the optimal exploration strategy can be found by the ε trade-off point through the intersection of preference satisfaction and goal achievement trajectories. Our framework offers a systematic methodology to control and balance exploration and exploitation, enhancing the learning capabilities of RL agents in a variety of grid environments.

V. DISCUSSION

We have presented an automata-theory approach to addressing the exploration-exploitation balance in RL. Modeling the RL problem as an MDP and subsequently representing the MDP as an NDFA enable the dynamic nature of the RL environment to be effectively described. Our approach incorporates the preference for exploring unexplored areas in the RL environment by designating specific states in the NDFA. The exploration-exploitation balance problem can be formulated as an MIP optimization problem, allowing the optimal policies that satisfy the defined preferences to be found. Leveraging the automata theory has led to a principled framework for balancing exploration and exploitation in RL tasks. Simulations have demonstrated the efficacy of the proposed framework in achieving a favorable exploration-exploitation trade-off. The results indicate that, in certain scenarios, even suboptimal policies can lead to a higher satisfaction of exploration preferences due to the challenges posed by wall positioning and suboptimal initial policies.

It is worth stressing the connection between adaptive automaton and adaptive network. Viewing an automaton as an adaptive network with a time-varying transition matrix entails interpreting the states and transitions of the automaton in a dynamic network framework. In an automaton, states represent different conditions or configurations of the system.

Each state can be regarded as a node in the network. Transitions in the automaton denote the movement from one state to another based on certain conditions or inputs. These transitions can then be viewed as edges connecting the nodes in the network. In a traditional automaton, the transition matrix is fixed, meaning the probabilities of transitioning from one state to another remain constant. However, in an adaptive network, the transition probabilities are time-varying, which can change over time according to various factors such as external inputs, system dynamics, or the learning mechanism. The transition probabilities in the time-varying transition matrix can adapt based on the current state of the system and its interactions with the environment. The adaptation may be driven by learning algorithms, feedback mechanisms, or any other dynamic processes. By treating the automaton as an adaptive network, we can analyze its behavior over time in terms of network dynamics. This includes studying how the states evolve, how transitions occur, and how the system adapts to changing conditions.

A limitation is the scalability associated with the MIP formulation. As the size of the environment increases, the number of constraints in the MIP problem grows exponentially, requiring the development of techniques to reduce the computational complexity of the MIP formulation. Potential solutions include exploring approximation algorithms, constraint relaxation methods, and decomposition approaches to efficiently handle larger RL environments. One solution is to leverage advanced optimization techniques, such as parallel computing, heuristics, and metaheuristic algorithms, to enhance the efficiency of solving large-scale MIP problems encountered in expansive environments. Exploring problem-specific reformulation or decomposition approaches can also be beneficial to reducing computational burdens.

Another limitation pertains to the criterion employed for assessing goal achievement and determining the exploration-exploitation trade-off. While our current criterion provides a reasonable evaluation metric, there is room for improvement. Future research could strive to develop more refined and adaptive criteria that capture the balance between exploration and exploitation. This could involve incorporating additional factors, such as uncertainty estimation, information gain, or dynamic adaptation of exploration rates based on environmental characteristics, requiring the agent's adaptability, robustness to environmental changes, or convergence speed to optimal solutions to be measured. Such metrics could provide deeper insights into the effectiveness of our framework to enable a more comprehensive comparison with existing methods.

ACKNOWLEDGMENTS

This work was supported by AFOSR under Grant No. FA9550-21-1-0438. We would like to attribute "Freepik" and "Nikita Golubev" for their provision of the "tree" and "taxi" icons utilized in Figs. 1 and 3.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon request.

Appendix A: Four application examples

The methods and frameworks presented in this paper provide a versatile approach to addressing complex, real-world problems, particularly in the context of dynamic and adaptive environments. The RL exploration-exploitation balance, tackled through MDP models and automata theory, has implications across a wide range of domains. These applications span from autonomous systems and robotics to network optimization and system control, where intelligent agents must navigate complex, unpredictable environments while making optimal decisions. Here in Appendix, we explore several real-world applications, demonstrating how our framework can be translated into practical settings. Ranging from robotic maintenance in power grids to sensor calibration and control system optimization, these examples illustrate the broad applicability of our approach and its potential to significantly enhance decision-making processes in complex environments. More specifically, we describe these applications and show how they can be translated into the MDP and automata frameworks.

a. Robotic maintenance in a power grid

Consider a robotic device navigating a grid-like environment to perform maintenance tasks on electrical components. This example can be formulated as an MDP and represented using automata as well. In the MDP formulation, the state space comprises the locations of the robotic device in the grid-like environment and the status of nearby electrical components. Each state s includes the robot's current position and information about nearby components. Actions a correspond to movement actions and repair actions. The agent has the following movement actions: "Move Up" (a_1), "Move Down" (a_2), "Move Left" (a_3), and "Move Right" (a_4). The agent also has the "Repair" action (a_5) to fix nearby malfunctioning components. The transition dynamics capture the stochasticity of robot movement and repair success. Movement actions lead to changes in the robot's position, and repair actions result in fixing components. The transition functions are defined as follows:

- For movement actions:
  - $s' = (x', y')$ with probability $P(s'|s, a_i)$
- For the repair action:
  - Component status changes with probability $P(\text{Fixed}|\text{Malfunctioning}, a_5)$

Immediate rewards $R(s,a)$ are based on the robot's actions; for example, repairing a component yields a positive reward. The main goal is to explore the grid efficiently to discover malfunctioning components and optimize repair actions, while minimizing negative rewards.

For Automata representation, the automaton states represent different behavior modes of the robot:

- State $\tilde{s}_1$ ("Patrol"): Exploring the grid for components
- State $\tilde{s}_2$ ("Repair"): Repairing malfunctioning components

b. Sensor calibration

We consider an exploration scenario involving the calibration of sensors in an environment. Each MDP state s includes accuracy (A_s) and precision (P_s) levels of the pressure sensor. Actions correspond to adjusting the calibration settings of the sensors. These actions influence the sensors' measurements and their uncertainty. Adjusting the calibration settings introduces uncertainties in the sensor measurements. The agent can take actions a to adjust the calibration settings of the pressure sensor. The agent has two choices: "Increase Accuracy" (a_1) and "Increase Precision" (a_2). The transition dynamics capture how the calibration process affects the sensors' accuracy and precision. The transition dynamics are probabilistic and involve uncertainty in calibration adjustments. Immediate rewards are based on the quality of the measurements after performing a calibration action. Achieving highly accurate and precise measurements leads to positive rewards. The immediate reward function $R(s,a)$ is defined as the sum of accuracy and precision. The goal is to explore the space of possible calibration settings to find the optimal configuration that maximizes the accuracy and precision of sensor measurements.

The automaton states represent the different stages of calibration. States could include "Initial Calibration," "Fine-Tuning," "High Precision," etc. Transitions between states are based on the agent's choices of calibration actions. For example, transitioning from "Initial Calibration" to "Fine-Tuning" could occur when the agent decides that the initial settings need to be refined. The automaton states represent different calibration stages, for example:

- State $\tilde{s}_1$ ("initial calibration"): $A_{\tilde{s}_1} = 50$, $P_{\tilde{s}_1} = 40$
- State $\tilde{s}_2$ ("fine tuning"): $A_{\tilde{s}_2} = 50$, $P_{\tilde{s}_2} = 40$
- State $\tilde{s}_3$ ("optimized calibration"): $A_{\tilde{s}_3} = 50$, $P_{\tilde{s}_3} = 40$

An accepting state might signify the achievement of the best possible calibration configuration with high accuracy and precision.

c. Circuit design optimization

In a circuit design problem, each MDP state represents the current configuration of a circuit design, including the values of various components (resistors, capacitors, etc.) and their connections. Actions correspond to modifying the values of circuit components, reconfiguring connections, and making design choices. Moreover, changing component values and connections affects the behavior of the circuit. The transition dynamics capture how modifications influence the circuit's performance and behavior. Immediate rewards are based on the circuit's performance metrics, such as signal quality, power consumption, or noise levels. Achieving desired circuit behavior yields positive rewards. The primary goal is to explore different circuit design configurations to optimize specific performance criteria while meeting design specifications.

The automaton states represent different design stages, such as "Initial Configuration," "Parameter Tuning," "Validation," etc. Further, transitions between states occur as the agent makes design modifications. For example, transitioning from "Initial Configuration" to "Parameter Tuning" could happen when the agent decides to fine-tune component values. An accepting state could signify the successful optimization of the circuit design, where the design meets the desired performance criteria. The agent's decisions are driven by immediate rewards tied to circuit performance and the long-term exploration goal of achieving the best design configuration.

d. Control system optimization for a pendulum

In a control system problem, each MDP state represents the current state of a pendulum system, including the pendulum's angle, angular velocity, and other relevant parameters. Actions correspond to control inputs that affect the pendulum's motion, such as applying torques or forces. The dynamics of the pendulum system are affected by the control inputs. The transition dynamics capture how the pendulum's state changes based on the applied control actions. Immediate rewards are based on the stability and performance of the control system. Keeping the pendulum balanced and minimizing oscillations yield positive rewards. The goal is to explore different control strategies to stabilize the pendulum and optimize its performance.

The automaton states represent different control modes, such as "Stabilization," "Trajectory Tracking," "Control Tuning," etc. Transitions between states occur as the agent selects different control strategies. For example, transitioning from "Stabilization" to "Trajectory Tracking" could happen when the agent wants to move the pendulum to a specific position. An accepting state could signify the successful optimization of the control system, where the pendulum is stabilized and follows desired trajectories accurately.

[1] L. Zhu, Y.-C. Lai, F. C. Hoppensteadt, and J. He, "Cooperation of spike timing-dependent and heterosynaptic plasticities in neural networks: A Fokker-Planck approach," Chaos **16**, 023105 (2006).
[2] C. Kuehn, "Time-scale and noise optimality in self-organized critical adaptive networks," Phys. Rev. E **85**, 026103 (2012).
[3] O. V. Popovych, S. Yanchuk, and P. A. Tass, "Self-organized noise resistance of oscillatory neural networks with spike timing-dependent plasticity," Sci. Rep. **3**, 2926 (2013).
[4] D. V. Kasatkin, S. Yanchuk, E. Schöll, and V. I. Nekorkin, "Self-organized emergence of multilayer structure and chimera states in dynamical networks with adaptive couplings," Phys. Rev. E **96**, 062211 (2017).
[5] L. Horstmeyer, C. Kuehn, and S. Thurner, "Network topology near criticality in adaptive epidemics," Phys. Rev. E **98**, 042313 (2018).
[6] L. Horstmeyer and C. Kuehn, "Adaptive voter model on simplicial complexes," Phys. Rev. E **101**, 022305 (2020).
[7] R. Berner, J. Sawicki, and E. Schöll, "Birth and stabilization of phase clusters by multiplexing of adaptive networks," Phys. Rev. Lett. **124**, 088301 (2020).

[8] R. Berner, S. Vock, E. Schöll, and S. Yanchuk, "Desynchronization transitions in adaptive networks," Phys. Rev. Lett. **126**, 028301 (2021).

[9] R. Berner, S. Yanchuk, and E. Schöll, "What adaptive neuronal networks teach us about power grids," Phys. Rev. E **103**, 042315 (2021).

[10] L. Horstmeyer, C. Kuehn, and S. Thurner, "Balancing quarantine and self-distancing measures in adaptive epidemic networks," Bull. Math. Biol. **84**, 79 (2022).

[11] D. Schlager, K. Clauβ, and C. Kuehn, "Stability analysis of multiplayer games on adaptive simplicial complexes," Chaos **32**, 053128 (2022).

[12] M. A. Gkogkas, C. Kuehn, and C. Xu, "Continuum limits for adaptive network dynamics," Commun. Math. Sci. **21**, 83–106 (2023).

[13] B. Jüttner and E. A. Martens, "Complex dynamics in adaptive phase oscillator networks," Chaos **33**, 053106 (2023).

[14] K. Klemm and E. A. Martens, "Bifurcations in adaptive vascular networks: Toward model calibration," Chaos **33**, 093135 (2023).

[15] J. Sawicki, R. Berner, S. A. Loos, M. Anvari, R. Bader, W. Barfuss, N. Botta, N. Brede, I. Franović, D. J. Gauthier, *et al.*, "Perspectives on adaptive dynamical systems," Chaos **33** (2023).

[16] R. Berner, T. Gross, C. Kuehn, J. Kurths, and S. Yanchuk, "Adaptive dynamical networks," Phys. Rep. **1031**, 1–59 (2023).

[17] O. V. Maslennikov and V. I. Nekorkin, "Adaptive dynamical networks," Phys. Uspekhi **60**, 694 (2017).

[18] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," Journal of artificial intelligence research **4**, 237–285 (1996).

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction* (MIT press, 2018).

[20] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," Adapt. Learning Opt. **12**, 729 (2012).

[21] H. Liu, A. Kumar, W. Yang, and B. Dumoulin, "Explore-exploit: A framework for interactive and online learning," arXiv preprint arXiv:1812.00116 (2018).

[22] M. Moradi, Y. Weng, and Y.-C. Lai, "Defending smart electrical power grids against cyberattacks with deep Q-learning," PRX Energy **1**, 033005 (2022).

[23] S. Song, J. Weng, H. Su, D. Yan, H. Zou, and J. Zhu, "Playing fps games with environment-aware hierarchical reinforcement learning." in *IJCAI* (2019) pp. 3475–3482.

[24] S. Curi, F. Berkenkamp, and A. Krause, "Efficient model-based reinforcement learning through optimistic policy search and planning," Adv. Neu. Info. Proc. Sys. **33**, 14156–14170 (2020).

[25] M. Moradi, Y. Weng, J. Dirkman, and Y.-C. Lai, "Preferential cyber defense for power grids," PRX Energy **2**, 043007 (2023).

[26] M. Moradi, S. Panahi, Z.-M. Zhai, Y. Weng, J. Dirkman, and Y.-C. Lai, "Heterogeneous reinforcement learning for defending power grids against attacks," APL Mach. Learn. **2**, 026121 (2024).

[27] Y. Zhang, P. Cai, C. Pan, and S. Zhang, "Multi-agent deep reinforcement learning-based cooperative spectrum sensing with upper confidence bound exploration," IEEE Access **7**, 118898–118906 (2019).

[28] P. Auer, "Using upper confidence bounds for online learning," in *Proceedings 41st Annual Symposium on Foundations of Computer Science* (IEEE, 2000) pp. 270–279.

[29] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, Z. Wen, *et al.*, "A tutorial on thompson sampling," Found. Trends Mach. Learning **11**, 1–96 (2018).

[30] Y. Ouyang, M. Gagrani, A. Nayyar, and R. Jain, "Learning unknown Markov decision processes: A Thompson sampling approach," Adv. Neu. Info. proc. Sys. **30** (2017).

[31] T. Zhang, "Feel-good thompson sampling for contextual bandits and reinforcement learning," SIAM J. Math. Data Sci. **4**, 834–857 (2022).

[32] A. Gopalan and S. Mannor, "Thompson sampling for learning parameterized markov decision processes," in *Conference on Learning Theory* (PMLR, 2015) pp. 861–898.

[33] I. Osband and B. Van Roy, "Bootstrapped thompson sampling and deep exploration," arXiv preprint arXiv:1507.00300 (2015).

[34] A. G. Barto, "Intrinsic motivation and reinforcement learning," Intrin. Motiv. Learning Nat. Artif. Sys. , 17–47 (2013).

[35] A. Aubret, L. Matignon, and S. Hassas, "A survey on intrinsic motivation in reinforcement learning," arXiv preprint arXiv:1908.06976 (2019).

[36] N. Chentanez, A. Barto, and S. Singh, "Intrinsically motivated reinforcement learning," Adv. Neu. Info. proc. Sys. **17** (2004).

[37] J. Li, X. Shi, J. Li, X. Zhang, and J. Wang, "Random curiosity-driven exploration in deep reinforcement learning," Neurocomputing **418**, 139–147 (2020).

[38] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," arXiv preprint arXiv:1804.00456 (2018).

[39] L. Zheng, J. Chen, J. Wang, J. He, Y. Hu, Y. Chen, C. Fan, Y. Gao, and C. Zhang, "Episodic multi-agent reinforcement learning with curiosity-driven exploration," Adv. Neu. Info. proc. Sys. **34**, 3757–3769 (2021).

[40] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," Sci. Robot. **4**, eaay6276 (2019).

[41] N. Fulton and A. Platzer, "Verifiably safe off-model reinforcement learning," in *Tools and Algorithms for the Construction and Analysis of Systems: 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part I* (Springer, 2019) pp. 413–430.

[42] B. Könighofer, F. Lorber, N. Jansen, and R. Bloem, "Shield synthesis for reinforcement learning," in *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles: 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, Rhodes, Greece, October 20–30, 2020, Proceedings, Part I 9* (Springer, 2020) pp. 290–306.

[43] G. Amir, M. Schapira, and G. Katz, "Towards scalable verification of deep reinforcement learning," in *2021 formal methods in computer aided design (FMCAD)* (IEEE, 2021) pp. 193–203.

[44] J. Yang, I. Borovikov, and H. Zha, "Hierarchical cooperative multi-agent reinforcement learning with skill discovery," arXiv preprint arXiv:1912.03558 (2019).

[45] C. Du, Y. Lu, H. Meng, and J. Park, "Evolution of cooperation on reinforcement-learning driven-adaptive networks," Chaos **34** (2024).

[46] S. E. Razavi, M. A. Moradi, S. Shamaghdari, and M. B. Menhaj, "Adaptive optimal control of unknown discrete-time linear systems with guaranteed prescribed degree of stability using reinforcement learning," Int. J. Dyn. Cont. **10**, 870–878 (2022).

[47] Z.-M. Zhai, M. Moradi, L.-W. Kong, B. Glaz, M. Haile, and Y.-C. Lai, "Model-free tracking control of complex dynamical trajectories with machine learning," Nat. Commun. **14**, 5698 (2023).

[48] Y. Lu, Y. Wang, Y. Liu, J. Chen, L. Shi, and J. Park, "Reinforcement learning relieves the vaccination dilemma," Chaos **33** (2023).

[49] L.-W. Kong, H.-W. Fan, C. Grebogi, and Y.-C. Lai, "Machine learning prediction of critical transition and system collapse," Phys. Rev. Res. **3**, 013090 (2021).

[50] L.-W. Kong, H.-W. Fan, C. Grebogi, and Y.-C. Lai, "Emergence of transient chaos and intermittency in machine learning," J. Phys. Complex. **2**, 035014 (2021).

[51] S. Panahi and Y.-C. Lai, "Adaptable reservoir computing: a paradigm for model-free data-driven prediction of critical transitions in nonlinear dynamical systems," Chaos **34**, 051501 (2024).

[52] L.-W. Kong, G. A. Brewer, and Y.-C. Lai, "Reservoir-computing based associative memory and itinerancy for complex dynamical attractors," Nat. Commun. **15**, 4840 (2024).

[53] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd ed. (Addison-Wesley, New York, 2006).

[54] B. Khoussainov and A. Nerode, *Automata Theory and Its Applications* (Springer Science & Business Media, 2012).

[55] M. O. Rabin and D. Scott, "Finite automata and their decision problems," IBM J. Res. Develop. **3**, 114–125 (1959).

[56] J. Fu, "Probabilistic planning with preferences over temporal goals," in *2021 American Control Conference (ACC)* (IEEE, 2021) pp. 4854–4859.

[57] R. Berner, *Patterns of Synchrony in Complex Networks of Adaptively Coupled Oscillators* (Springer Nature, 2021).

[58] C. Zhou and J. Kurths, "Dynamical weights and enhanced synchronization in adaptive complex networks," Phys. Rev. Lett. **96**, 164102 (2006).

[59] F. Sorrentino and E. Ott, "Adaptive synchronization of dynamics on evolving complex networks," Phys. Rev. Lett. **100**, 114101 (2008).

[60] L. M. Pecora and T. L. Carroll, "Master stability functions for synchronized coupled systems," Phys. Rev. Lett. **80**, 2109 (1998).

[61] T. Nishikawa, A. E. Motter, Y.-C. Lai, and F. C. Hoppensteadt, "Hetero-

geneity in oscillator networks: Are smaller worlds easier to synchronize?" Phys. Rev. Lett. **91**, 014101 (2003).
[62]L. Huang, K. Park, Y.-C. Lai, L. Yang, and K. Yang, "Abnormal synchronization in complex clustered networks," Phys. Rev. Lett. **97**, 164101 (2006).
[63]X. Wang, L. Huang, Y.-C. Lai, and C. H. Lai, "Optimization of synchronization in gradient clustered networks," Phys. Rev. E **76**, 056113 (2007).
[64]L. Huang, Y.-C. Lai, and R. A. Gatenby, "Dynamics-based scalability of complex networks," Phys. Rev. E **78**, 045102 (2008).
[65]L. Huang, Q. Chen, Y.-C. Lai, and L. M. Pecora, "Generic behavior of master-stability functions in coupled nonlinear dynamical systems," Phys. Rev. E **80**, 036204 (2009).
[66]F. Sorrentino, G. Barlev, A. B. Cohen, and E. Ott, "The stability of adaptive synchronization of chaotic systems," Chaos **20**, 013103 (2010).
[67]R. Berner, S. Vock, E. Schöll, and S. Yanchuk, "Desynchronization transitions in adaptive networks," Phys. Rev. Lett. **126**, 028301 (2021).
[68]N. Pisaruk, *Mixed Integer Programming: Models and Methods* (Belarus State University, 2019).