scientific reports

Check for updates

OPEN

A synthesis method for zero-sum mean-payoff asynchronous probabilistic games

Wei Zhao[1]✉, Wanwei Liu[2,5], Zhiming Liu[3,5]✉ & Tiexin Wang[4,5]

The traditional synthesis problem aims to automatically construct a reactive system (if it exists) satisfying a given Linear Temporal Logic (LTL) specifications, and is often referred to as a qualitative problem. There is also a class of synthesis problems aiming at quantitative properties, such as mean-payoff values, and this type of problem is called a quantitative problem. For the two types of synthesis problems, the research on the former has been relatively mature, and the latter also has received huge amounts of attention. System designers prefer to synthesize systems that satisfy resource constraints. To this end, this paper focuses on the reactive synthesis problem of combining quantitative and qualitative objectives. First, zero-sum mean-payoff asynchronous probabilistic games are proposed, where the system aims at the expected mean payoff in a probabilistic environment while satisfying an LTL winning condition against an adversarial environment. Then, the case of taking the wider class of Generalized Reactivity(1) (GR(1)) formula as an LTL winning condition is studied, that is, the synthesis problem of the expected mean payoffs is studied for the system with the probability of winning. Next, two symbolic algorithms running in polynomial time are proposed to calculate the expected mean payoffs, and both algorithms adopt uniform random strategies. Combining the probability of system winning, the expected mean payoffs of the system when it has the probability of winning is calculated. Finally, our two algorithms are implemented, and their convergence and volatility are demonstrated through experiments.

Traditional Linear Temporal Logic (LTL) synthesis aims to automatically construct a system that satisfy the given LTL specification, and is purely qualitative. In recent years, with the extensive application of formal methods in theoretical computer science and artificial-intelligence related fields, only considering the synthesis of qualitative objectives can not fully express the quantitative properties of the system, such as energy consumption and transmission efficiency. In general, a quantitative property can be regarded as a function that maps the execution of a system to a numerical value. Thus, researchers promote formal verification and system synthesis by combining qualitative properties with quantitative properties of the system[1]. Some studies focus on synthesizing strategies for quantitative goals under almost-sure winning conditions[2,3]. Other studies calculate the maximize (or minimize) mean payoff or ensure an almost-sure (or worst-case) threshold based on a mean-payoff game[4–8]. However, these studies do not consider the complex uncertain environment.

Even though there is an increasing interest in system synthesis with quantitative objectives, such synthesis problems are still challenging due to complex environmental changes. How to obtain the expected mean payoffs of the system in an uncertain environment needs further consideration. Generally, probabilistic games can reasonably describe the uncertainty of the environment. This study considers Asynchronous Probabilistic Games (APGs) equipped with the GR(1) winning objective and the expected mean-payoff condition, where the expected mean payoff is taken as a quantitative objective. The study of the synthesis problem of GR(1) specification focuses on how to obtain the expected mean payoffs of the system under the winning probability. First, a new model is constructed based on APGs, which is called zero-sum mean-payoff APGs. APGs are a class of models that exhibit stochastic and non-deterministic behaviors of systems and environments. Then, to obtain the expected mean payoffs, two algorithms are proposed to calculate the average mean payoffs for each player in the game that has the probability of winning. Based on the state properties, the first algorithm calculates the mean payoffs of each state when it satisfies the winning condition, and this algorithm is called *State-MP*. Specifically, it is assumed that there are n plays starting from a system state (or an environment state) that the

[1]The Department of Computer Engineering, Jiangsu University of Technology, Changzhou 213001, China. [2]The National University of Defense Technology, Changsha 410073, China. [3]Southwest University, Chongqing 400715, China. [4]The College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, Jiangsu, China. [5]Wanwei Liu, Zhiming Liu and Tiexin Wang contributed equally. ✉email: zhaowei618@jsut.edu.cn; zhimingliu88@swu.edu.cn

mean payoff is the average of the total payoffs of n plays (including loops play). The expected mean payoff that the system has a probability of winning is calculated by weighting the probability of the system winning and its mean payoff. If the expected mean payoff is larger than 0, the system obtains the value, and the environment obtains the negative of this value; if the expected mean payoff is smaller than 0, the environment obtains this value, and the system obtains the negative of the value. The other algorithm calculates the mean payoff of each state based on the path properties, and it is called *Path-MP*. In detail, it searches for and marks the plays that start from a system (or environment) state and do not reach the state satisfying the GR(1) winning condition. Based on the analysis of plays, the mean payoffs of the marked plays are calculated, and then the mean payoffs of the plays that satisfy the system winning condition are calculated. At this point, the mean reward for each state when the system has a probability of winning can be obtained. Meanwhile, the expected mean payoff is weighted by the probability of the system winning and its mean payoff. Then, the two participants (the system and the environment) obtain the mean payoff according to the allocation policy of the Path-MP algorithm. This study proves the feasibility of the two algorithms and uses the mean error method to compare their convergence and volatility. The comparison results indicate that the State-MP algorithm converges faster and is more stable than the Path-MP algorithm. Both algorithms can determine the mean payoff of the players in polynomial time.

Since there is no work on the expected mean-payoff synthesis of the system under the winning probability. This paper considers the zero-sum mean-payoff probabilistic game between the system and environment with GR(1) winning condition. The main contributions of this paper are as follows:

- The synthesis problem is investigated for mean-payoff objectives under the probability of system winning based on the zero-sum mean-payoff APGs with the GR(1) winning condition.
- Two efficient algorithms, namely State-MP and Path-MP, are proposed to calculate the expected mean payoff. Meanwhile, the mean error method is adopted to compare the convergence and volatility of the two proposed algorithms.

Related work

The synthesis problem was first proposed[9]. In 1989, Linear Temporal Logic (LTL) synthesis was again considered and solved by Pnueli[10]. However, the LTL synthesis was generally considered impractical due to its high computational complexity. After that, the researchers worked to reduce the computational complexity to an acceptable range. Meanwhile, it was found that the synthesis algorithm for a particular fragment of LTL was in polynomial time[11–15]. The representative studies are the efficient synthesis algorithms for various LTL formulas presented in[11,13]. In 2006, the research[15] has proposed the wider class of Generalized Reactivity(1) (GR(1) for short) formulas, which was described by a fragment of LTL. The synthesis problem taking the GR(1) formula as the specification can be solved with time complexity of $O(N^3)$, where N is the size of the state space. All these studies only deal with the plain qualitative synthesis problem, while this paper considers quantitative synthesis.

Mean payoff is one of the most classical quantitative properties of reactive systems. Generally, quantitative properties are constructed as a class of functions that map the executions of system to a real number. Also, mean-payoff games are a frequently-used model to analyze the quantitative properties. Mean-payoff games are usually turn-based two-player games on a finite graph with integer weights on the edges, and they were first studied[16]. Then, they have proposed that memoryless strategies can obtain the optimal mean payoff. Such games can be extended to many types of games, such as multi-player mean-payoff games[17] and two-player multi-mean-payoff games[18]. They all focus on formulating winning strategies for players, and the problem is NP$\cap$co-NP[17,18]. In general, the mean-payoff property can be studied based on zero-sum games. This type of game is the two-player games, and the gain of one player is equal to the loss of the other players, i.e., the net change in benefit or return is zero. However, zero-sum games cannot adequately express the interactions between the participants. Considering the properties of the above two games, this study considers zero-sum mean-payoff asynchronous probabilistic games, where the rewards are based on the player's choice of action in a probabilistic environment.

Another class of research focuses on quantitative synthesis under a probabilistic environment, where Markov decision processes (MDPs) can better exhibit both stochastic and non-deterministic behaviors. Naturally, mean-payoff MDP can be modeled as a game with a cost at each position. The strategy of minimizing the expected cost was investigated[19,20]. The difference between the two studies is the winning condition that needs to be satisfied. The work[21] presented finds a strategy to minimize the expected cost while satisfying the ω-regular condition. If the logic specification such as an ω-regular condition is defined as a "soft" constraint, then the costs (or rewards) can be defined as a "hard" constraint. When two constraints conflict, the worst-case strategy that satisfies the hard constraint is first selected, and then the strategy that minimizes the soft constraint is found. This alternating process can take place indefinitely. Moreover, recent research has examined a combination of deterministic and stochastic disturbances within a discrete-time Nash game framework, aiming to maximize the expected cost function for each player[22]. Our work studies how to calculate the expected mean payoff of the system satisfying the GR(1) winning condition in the opposition of the system and environment. Compared with the multidimensional mean-payoff MDPs[20] and multi-player mean-payoff concurrent games[23], our work focus more on the asynchronous game between the environment and the system.

There are other works[5,24–26] to analyze quantitative property based on MDPs and mean-payoff objectives. In particular, An algorithm was proposed to solve the problem of calculating the expected mean payoff over MDPs[25]. In their combination objective, they considered mean payoff[5] and parity conditions[17,26]. The mean payoff was combined with parity objectives as the objectives of MDPs and stochastic games[5,26]. The system synthesis aimed to satisfy the parity condition while the mean payoff was less than a given threshold. While the system's uncertainty is taken into account, the worst-case optimality of control is established in the form of a weighted sum derived from the discrete-time minimum principle[27]. None of these studies investigate how environment uncertainty affects system synthesis.

Preliminaries

GR(1): a fragment of LTL

LTL has become a classical logical language for describing the properties of reactive systems. Commonly used temporal operators include X ("next"), U ("until"), F ("finally"), and G ("globally"). The temporal operators GF and FG can be derived by the above temporal operators. These two temporal operators are often used to extend propositional logic.

Given a finite of Boolean variables *AP* and an infinite sequence $\pi = v_0 v_1 \cdots \in (2^{AP})^{\omega}$, π is a computation iff π assigns v_i to *AP*. This paper uses $\pi, i \models \varphi$ to indicate that the LTL formula φ satisfies at the *i*-th position of π, where $i \in \mathbb{N}$.

The time complexity of LTL synthesis algorithms is double-exponential. To reduce the computational complexity of these algorithms, a special LTL fragment called the GR(1) specification was discovered and studied[14].

Suppose *X* and *Y* are two finite sets of atomic propositions *AP* that satisfy $AP = X \cup Y$. The proposition sets *X* and *Y* are controlled by the environment and the system, respectively. The GR(1) specification contains the following elements:

1. θ^e and θ^s are not-temporal Boolean formulas defined on *X* and *Y*, respectively. θ^e and θ^s are used to describe the specification for the initial condition of the environment and the system, respectively.
2. φ_t^e is a conjunction of formulas of the form $\mathrm{G}A$, and *A* is a Boolean expression defined on $X \cup Y \cup X'$, where $X' = \{\bigcirc x | x \in X\}$. φ_t^e is used to describe the environment transition relation, which only limits the state of the environment to change.
3. φ_t^s is a conjunction of formulas of the form $\mathrm{G}B$, and *B* is Boolean expression defined on $X \cup Y \cup X' \cup Y'$, where $Y' = \{\bigcirc y | y \in Y\}$. φ_t^s is used to describe the system transition relationship, which limits the state of the system and the state of the environment to change.
4. φ_g^e and φ_g^s are the conjunctions of formulas of the form $\mathrm{GF}C$, and *C* is Boolean expression defined on $X \cup Y$. φ_g^e and φ_g^s are used to describe the goals of the environment and the system, respectively.

An LTL formula of the form $\psi = \varphi^e \Rightarrow \varphi^s$ is called an GR(1) formula, where $\varphi^e = \theta^e \wedge \varphi_t^e \wedge \varphi_g^e$ and $\varphi^s = \theta^s \wedge \varphi_t^s \wedge \varphi_g^s$. A GR(1) specification indicates an implication relation between a set of assumptions about the environment and a set of guarantees about the system, that is, the behavior of the system is guaranteed to satisfy φ^s if the assumption that the behavior of environments satisfies φ^e.

Asynchronous probabilistic games

This section introduces some concepts of asynchronous probabilistic games, including plays, winning conditions, strategies, etc.

Asynchronous probabilistic games combine the properties of a two-player turn-based game and Markov Decision Process (MDP). Two players move alternately in the arena. Each position (or state) is determined by the player's action, and the choice of the next position (or state) is probabilistic. An asynchronous probabilistic game (APG) is defined as follows:

An *Asynchronous Probabilistic Game (APG)* can be represented as a six-element tuple $\mathscr{G} = \langle AP, V, Act, \Theta_E, \Theta_S, L \rangle$, and each element is described as follows:

- *AP* is a nonempty, finite set of atomic propositions.
- *V* consists of two disjoint state sets V_E and V_S on the game arena, where $V = V_E \cup V_S$, and $V_E \cap V_S = \varnothing$. The sets V_E and V_S are defined as the environment and system state sets, respectively.
- *Act* is the union of the action sets of the system and environment. Let *Act*(*v*) denote the set of available actions at state *v*, where $v \in V$.
- $\Theta_E : V_E \times Act \to Dist(V_S)$ is an environment transition function that maps each pair $(v_e, a) \in V_E \times Act$ to a discrete probability distribution $Dist(V_S)$ on V_S, where $v_e \in V_E$ and $a \in Act$.
- $\Theta_S : V_S \times Act \to Dist(V_E)$ is a system transition function, which maps each pair $(v_s, a) \in V_S \times Act$ to a discrete probability distribution $Dist(V_E)$ on V_E, where $v_s \in V_S$ and $a \in Act$.
- $L : V \to 2^{AP}$ is a labeling function, and *L*(*v*) represents the set of atomic propositions that hold in *v*.

Given an asynchronous probabilistic game $\mathscr{G}$, if a finite or an infinite sequence of states $\pi = v_0, v_1, \ldots$ satisfies for every $i \geq 0$ that v_{i+1} is a successor of v_i, then π is called a *play*, and v_i is also called the precursor of v_{i+1} . That is,

1. if $v_i \in V_E$, for some $a \in Act$, $\Theta_E(v_i, a)(v_{i+1}) > 0$, and for all $b \in Act$, $\Theta_S(v_i, b)(v_{i+1}) = 0$ and
2. if $v_i \in V_S$, for all $a \in Act$, $\Theta_E(v_i, a)(v_{i+1}) = 0$ and for some $b \in Act$, $\Theta_S(v_i, b)(v_{i+1}) > 0$.

v_0 is called the *initial state* of play π. This paper denotes the set of all plays of $\mathscr{G}$ by $\Pi_{\mathscr{G}}$. For any state *v* of the set *V*, the set of plays with *v* as the initial state is denoted as Π^v.

For an APG, the system winning condition φ is represented with the fragment of the LTL described in the previous subsection. A play π is winning for the system with φ if

1. π is a finite play, and the last state v_n of π is in set V_E for which there is no action $a \in Act$ such that $\Theta_E(v_n, a)(v_s) > 0$, or
2. π is an infinite play, and π satisfies φ.

Otherwise, π is winning for the environment.

For the system player of the APG $\mathscr{G}$, a strategy is a function $f : V \times V_S \to Act$. For all finite sequences of states ending in a state of the system, $f(v_0 \dots v_n) \in Act(v_n)$ denotes the next action that the system should perform. For the environment player of the APG $\mathscr{G}$, the strategy is defined based on the function $g : V \times V_E \to Act$. F_S and F_E denote the sets of the system strategies and environment strategies of $\mathscr{G}$ respectively, and the strategies are memoryless. A strategy *f* is memoryless if it relies only on the current state. In this paper, only memoryless strategies are considered for the players.

Let $\Pi^s_{\mathscr{G}}$ and $\Pi^e_{\mathscr{G}}$ be the sets of finite plays with the last state in V_S and V_E, respectively. For a play $\pi = v_0 v_1 \dots$, it follows a system (or environment) strategy *f*, if for each finite prefix $\tau = v_0 v_1 \dots v_i \in \Pi^s_{\mathscr{G}}$ ($\Pi^e_{\mathscr{G}}$ resp.), it holds that $\Theta(\tau, f(\tau))(v_{i+1}) > 0$. For an APG $v \in V$ and a state proposition φ, $Pr_f(v \models \varphi)$ denotes the probability of the plays whose initial states satisfies φ under a strategy *f*.

Given an APG $\mathscr{G}$, a set *B* is a subset of *V* which satisfies a state proposition (or Boolean expression) φ, that is, there is a state *v* in *B* such that φ is true of *v*. The *reachability* property of *B* (or equivalently φ) is specified by the LTL formula $\mathrm{F}\varphi$. It explains that some states in *B* appear in the computation of $\mathscr{G}$, or equivalently φ holds in some states of the computation. The *fairness* property of *B* is expressed by the LTL formula $\mathrm{GF}\varphi$. It indicates that some states in the set *B* occur infinitely times in the computation of $\mathscr{G}$, or equivalently φ holds for infinitely times in the computation.

Reachability probability[28]
Given an APG $\mathscr{G}$, a strategy *f*, and a winning condition φ, $B \subseteq V$ is a set of states, and any play π starting from a state in *B* satisfies φ. This paper uses $v \models \mathrm{F}B$ to denote a play that starts from *v* in *V*, satisfies φ and reaches some states in *B*. Also, this paper uses $Pr(v \models \mathrm{F}B)$ to denote the probability of a play that starts from *v*, satisfies φ and reaches some states in *B*. For a strategy *f*, a play starting from *v* along *f* has the property of $Pr(v \models \mathrm{F}B)$, and the probability of such a play under strategy *f* is denoted as $Pr_f(v \models \mathrm{F}B)$.

For the sake of clarity, it is important to note that for the APG $\mathscr{G}$, the state space *V*, the winning condition φ, and a subset $B \subseteq V$, any play π staring from a state in *B* satisfies φ. In the remainder of this paper, the presence of the set *B* signifies its property in satisfying the winning condition φ.

Problem formulation

With the definitions and notations introduced in the previous section, this section introduces the design of the novel probabilistic model and formulates the synthesis problems.

A Mean-payoff Asynchronous Probabilistic Game (MpAPG, for short) combines an asynchronous probabilistic game with a mean-payoff game. The two players in the game are always hostile to each other, and they randomly obtain the mean payoff based on the total number of plays. The winning condition of MpAPG is defined by the GR(1) formula φ.

Formally, a MpAPG is a tuple $\mathscr{G}^{\mathrm{M}} = \langle AP, V, Act, \Theta_E, \Theta_S, L, W \rangle$. Since the definitions of other elements of $\mathscr{G}^{\mathrm{M}}$ have been given, only *W* is defined here. Assume that $W : V \times Act \to \mathbb{R}$ is the weight function that maps state-action pairs (*v*, *a*) of $\mathscr{G}^{\mathrm{M}}$ to a non-negative real *R*, where $v \in V, a \in Act$, $\mathbb{R}$ is a set of non-negative reals, and $R \in \mathbb{R}$. At each step of this game, players obtain payoffs by choosing an action to move to the next position (state). Here, payoffs can accumulate over time. This paper considers using uniform random strategies to obtain payoffs. Note that the next position (state) for each move is chosen based on probability. The gain of a strategy *f* for the system is the mean payoff of a random transition in which the system proceeds according to *f* in $\mathscr{G}^{\mathrm{M}}$.

For the two players of the MpAPG, under strict competition, the gain of one player indicates the loss of other players, and the total sum of the gains and losses of two players must be "zero" in the MpAPG. That is, for the system, the payoff of the system is equal to the loss of the environment, and the same is true for the environment. This shows that our model is a zero-sum game.

For a finite or an infinite play $\pi = v_0 v_1 \cdots$ of the game, this paper uses $\omega(\pi) = \omega(v_0, a_0)\omega(v_1, a_1) \cdots$ to define the sequence of weights on π, where $v_i \in V, a_i \in Act, i = 1, 2, \dots$. Meanwhile, the accumulative sum of the weights on a finite play π is denoted as $W(\pi)$.

Now, the value of the play property over MpAPG is defined:

Play payoff

Let $\mathscr{G}^{\mathrm{M}}$ be an MpAPG with a state space *V*. For an infinite play $\pi = v_0, v_1, \dots$ of the game $\mathscr{G}^{\mathrm{M}}$, the reward of the play along π is

$$P(\pi) = W(\pi) = \sum_{n=0}^{\infty} \omega(v_n, a_n).$$

For all finite play $\pi_i \in \Pi^v$ starting from the state $v \in V$, where $i = 1, \dots, n$, this paper defines $TP(v) = \sum_{i=1}^{n} P(\pi_i)$ and $MP(v) = \sum_{i=1}^{n} \frac{1}{n} P(\pi_i)$. Then, the *total payoff* is denoted as *TP*(*v*), and the *mean payoff* is denoted as *MP*(*v*). For infinite play $P(\pi_i)$, the (lim-sup) total payoffs and mean payoffs on an infinite play π_i are then defined as $TP(v) := lim\ sup_{n\to\infty} \sum_{i=1}^{n} P(\pi_i)$ and $MP(v) := lim\ sup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} P(\pi_i)$, respectively.

Given an MpAPG $\mathscr{G}^{\mathrm{M}}$, a state $v \in V$, a strategy *f*, and a state proposition φ, this paper uses $MP(v \models \varphi)$ to define the mean payoff of the plays whose initial states satisfy φ under the probability $Pr_f(v \models \varphi)$ along *f*.

Let $\mathscr{G}^{\mathrm{M}}$ be an MpAPG, and V be the state space over it. $B \subseteq V$ is a set of states, and all plays with the initial state in B satisfy the winning condition φ. This paper uses $P(v \models \mathrm{F}B)$ to define the play payoff earned along an infinite play π until it reaches some states in B for the first time. Meanwhile, the total payoff of starting from state $v \in V$ to any state in B is denoted as $TP(v \models \mathrm{F}B)$. The total payoff proves useful when the expected mean-payoff problem is taken into account.

Play mean-payoff

Let $\mathscr{G}^{\mathrm{M}}$ be an MpAPG. V is the state space on $\mathscr{G}^{\mathrm{M}}$, B is a subset of V, and all paths with an initial state of B satisfy φ, where φ is the winning condition. For $v \in V$, the mean payoff of starting from v to any state in B is denoted as

$$MP(v \models \mathrm{F}B).$$

Theorem 1 *Consider an MpAPG $\mathscr{G}^{\mathrm{M}}$ and a winning condition φ, and $B \subseteq V$ is a set that satisfies φ. Then, $MP(v \models \mathrm{F}B)$ can be calculated in polynomial time by a memory-less strategy.*

Expected mean payoff

Given a MpAPG $\mathscr{G}^{\mathrm{M}}$, V is the state space, f is a strategy, φ is a given winning condition, B is a subset of V, all paths with an initial state of B satisfy φ, and W is a weight function. This paper denotes the expected mean payoff of starting from v to some states in B while satisfying φ as

$$EM(v \models \mathrm{F}B) = Pr_f(v \models \mathrm{F}B) \cdot MP(v \models \mathrm{F}B),$$

where $Pr_f(v \models \mathrm{F}B)$ is the probability of the system winning along strategy f.

Problem

Given an MpAPG $\mathscr{G}^{\mathrm{M}}$ and a set of states $B \subseteq V$, all plays starting from B satisfy a given winning condition φ , and $Pr_f(v \models \mathrm{F}B)$ is the probability of the system winning under strategy f. This paper needs to calculate the expected mean payoff of the system at each winning strategy f when the system has a probability of winning. i.e.,

$$EM(v \models \mathrm{F}B) = Pr_f(v \models \mathrm{F}B) \cdot MP(v \models \mathrm{F}B).$$

Methods

This section presents the theory and approach to finding the expected mean payoff for the MpAPG $\mathscr{G}^{\mathrm{M}}$. Two algorithms are proposed to solve this problem. Here, an MpAPG $\mathscr{G}^{\mathrm{M}}$ with the winning condition φ is considered. W is a weight function, $B \subseteq V$ is a set of states, and all plays starting from B satisfy φ. φ is a GR(1) formula, which has the following form:

$$\bigwedge_{i=1}^{k} \mathrm{GF}J_i^e \Rightarrow \bigwedge_{j=1}^{m} \mathrm{GF}J_j^s$$

where J_i^e and J_j^s are Boolean formulas over the states in $\mathscr{G}^{\mathrm{M}}$, for $i = 1, \ldots, k$ and $j = 1, \ldots, m$.

First, a set B that satisfies the GR(1) formula φ is calculated. Then, a set Q in which all plays starting from Q do not satisfy φ is found and analyzed, where $Q \subseteq V$, i.e., if $v \in Q$, $Pr(v \models \mathrm{F}B) = 0$. This paper defines the union of sets B and Q as the target state set T, where $T = B \cup Q$ and $T \subseteq V$. The sets B, Q, and T can be obtained according to the modal μ-calculus operators over APGs. Since it is not the focus of this paper, it will not be described in detail here, and the specific details can be found in[28]. The value of the mean payoff is then calculated according to Theorem 2 and Theorem 3, respectively. Finally, the expected mean payoff for the system winning in the probabilistic environment is calculated.

Mean payoffs based on the states

To calculate the expected mean payoffs for the system winning in a probabilistic environment, the mean payoff for each state needs to be known. Here, our first algorithm called State-MP is presented, which calculates mean payoffs based on state properties.

Let $\mathscr{G}^{\mathrm{M}}$ be an MpAPG, φ be the GR(1) winning condition, $B \subseteq V$ be a set of states that satisfy φ, and T be the set of target states. We assume that for every state v in V, there are n plays starting from v, where $n \in \mathbb{Z}$. This paper assumes that there are plays π_i (including loops play) starting from v, where $\pi_i \in \Pi_{\mathscr{G}}^v$, $i = 1, 2, \ldots, n$.

Theorem 2 (State-MP) *There exists a uniform random memoryless strategy, $MP(v \models \mathrm{F}B) = \frac{1}{n}TP(v \models \mathrm{F}B)$ such that one of the following cases:*

1. *If all plays starting from v satisfy the winning condition φ, that is, n plays can reach a state in set B, then the mean payoff for the system is the value of $MP(v \models \mathrm{F}B)$, and the mean payoff of the environment is the negative of this value.*

2. *If all plays starting from v do not satisfy the winning condition φ, that is, none of n plays can reach a state in set B, then the mean payoff for the environment is the value of $MP(v \models \mathrm{F}B)$, and the mean payoff of the system is the negative of this value.*
3. *Whether all plays starting from v satisfy the winning condition φ cannot be determined, that is, it is uncertain whether n plays starting from v can reach any state in B. If the mean payoff $MP(v \models \mathrm{F}B)$ is larger than* 0, *the system obtains this value. If the mean payoff $MP(v \models \mathrm{F}B)$ is smaller than* 0, *the environment obtains this value.*

Proof Since $\mathscr{G}^{\mathrm{M}}$ is a zero-sum game, for the two players (the system and the environment) in the game, the payoff is balanced.

Assume that *n* is the number of plays starting from *v* falls into, and *m* of them satisfy the winning condition φ. If $m = n$, then the system wins (i.e., $Pr(v \models \mathrm{F}B) = 1$) and gains rewards. According to the property of the zero-sum game, the environment obtains a negative value of the rewards. So, case (i) holds.

Similarly, if $m = 0$, then the environment wins (i.e. $Pr(v \models \mathrm{F}B) = 0$) and gains rewards. The system is a loser and obtains a negative value of the rewards. So, case (ii) holds.

If $0 < m < n$, then the total payoffs for the $n - m$ plays that do not satisfy the winning condition are marked as negative. At this time, the value of $TP(v \models \mathrm{F}B)$ is calculated. Since $n > 0$, if $TP(v \models \mathrm{F}B) > 0$, then $MP(v \models \mathrm{F}B) > 0$, and the system obtains this value; if $TP(v \models \mathrm{F}B) < 0$, then $MP(v \models \mathrm{F}B) < 0$, and the environment obtain this value. So, the case (iii) holds.

Given an MpAPG $\mathscr{G}^{\mathrm{M}}$, define one step of the player moving from one position (or state) to another position (or state) as a time step *t*, where $t = 1, 2, \ldots$. Given a finite (or infinite) play $\pi = v_0 v_1 \cdots$ starting from *v*, if π reaches a state in *B* that satisfies the GR(1) condition φ, then the value of the total payoff $P(\pi)$ is marked as positive; if π does not reach a state in *B*, the value of the total payoff $P(\pi)$ is marked as negative. This paper denotes the total mean payoffs value of *n* plays starting from state *v* in this case as $TP_1(v \models \mathrm{F}B)$. To determine whether the environment or the system obtains the mean payoff $MP(v \models \mathrm{F}B)$, this paper denotes the total payoff without considering a negative value as $TP_2(v \models \mathrm{F}B)$. For example, if $TP_1(v \models \mathrm{F}B) = TP_2(v \models \mathrm{F}B)$, it is said that the system obtains the value of the mean payoff, and the environment obtains the negative of the mean payoff. The details of this algorithm are presented in Algorithm 1.

Input: A MpAPG $\mathscr{G}^{\mathrm{M}}$ with finite space V and a state set $B \subseteq V$, the target state set T, the iteration numbers k, the time step t;
Output: the mean-payoffs $MP(v \models \mathsf{F}B)$ for all states $v \in V$

For $v_i \in V, i = 1, 2, \cdots$
If $v_i \in T$
  return 0
EndIf
Initialize $TP_1(v_i \models \mathsf{F}B) = 0$ **and** $TP_2(v_i \models \mathsf{F}B) = 0$;
For $j = 1, 2, \cdots, k$
  Initialize $P(\pi) = 0, t = 0$;
  While(True)
  If $v_{i\_t} \in T$
    break
  EndIf
  If $v_{i\_t} \in V_E$
    $v_{i\_t+1} = \Theta_e(v_{i\_t}, a_{\_t})$;
    $P(\pi_j) += \omega(v_{i\_t})$;
  Else
    $v_{i\_t+1} = \Theta_s(v_{i\_t}, a_{\_t})$;
    $P(\pi_j) += \omega(v_{i\_t})$;
  EndIf
    $t = t + 1$
  EndWhile
  If $v_{i\_t} \in B$
    $TP_1(v_i \models \mathsf{F}B) += P(\pi_j), TP_2(v_i \models \mathsf{F}B) += P(\pi_j)$;
  Else
    $TP_1(v_i \models \mathsf{F}B) -= P(\pi_j)$, $TP_2(v_i \models \mathsf{F}B) += P(\pi_j)$;
  EndIf
EndFor
  If $(TP_1(v_i \models \mathsf{F}B) == TP_2(v_i \models \mathsf{F}B)$
    all plays reach a state in B, the system gains the value of mean-payoff $TP_1(v_i \models \mathsf{F}B)/k$
  ElsIf $(TP_1(v_i \models \mathsf{F}B) + TP_2(v_i \models \mathsf{F}B) == 0)$
    all plays no reach a state in B, the environment gains the value of mean-payoff $-TP_1(v_i \models \mathsf{F}B)/k$
  Else
    If$(TP_1(v_i \models \mathsf{F}B) >= 0)$
      the system get the value of mean-payoff
    Else
    the environment gain the value of the mean-payoff $TP_1(v_i \models \mathsf{F}B)/k$
    EndIf
  EndIf
EndFor

Algorithm 1. State-MP: Computing the mean-payoffs of all states v with $MP(v \models \mathrm{F}B)$

Mean payoff based on the paths

Now, our second algorithm called Path-MP is presented, which calculates mean payoffs based on path properties.

Consider an MpAPG $\mathscr{G}^{\mathrm{M}}$, the GR(1) winning condition φ, and $B \subseteq V$ is a set of states that satisfy φ. Assume that there are plays π_i (including loops play) starting from v, and m plays of them can reach some states in B, where $i = 1, 2, \ldots, n$, $m \leq n$.

Theorem 3 (Path-MP) *If there exists a uniform random strategy,* $MP(v \models \mathrm{F}B)$ *satisfies in one of the following cases*:

1. *If all plays starting from v do not satisfy the winning condition* φ*, that is, none of the n plays can reach a state in set B, then the mean payoff for the environment is the value of* $MP(v \models \mathrm{F}B)$*, and the mean payoff of the system is the negative of this value.*
2. *If all plays starting from v satisfy the winning condition* φ*, that is, n plays can reach a state in set B, then the mean payoff for the system is the value of* $MP(v \models \mathrm{F}B)$*, and the mean payoff of the environment is the negative of this value.*
3. *Whether all plays starting from v satisfy the winning condition* φ *can be determined, that is, it is uncertain whether n plays starting from v can reach a state in B. Let* $MP(v \models \mathrm{F}B) = \frac{1}{m}P(\pi_m) - \frac{1}{n-m}P(\pi_{n-m})$ *. If the mean payoff* $MP(v \models \mathrm{F}B)$ *is larger than* 0*, then the system obtains this value. If the mean payoff* $MP(v \models \mathrm{F}B)$ *is smaller than* 0*, then the environment obtains this value.*

Proof Since $\mathscr{G}^{\mathrm{M}}$ is a zero-sum game, the payoff for the environment and the system is balanced.

If $m = 0$, then the environment wins (i.e., $Pr(v \models \mathrm{F}B) = 0$) and gains rewards. The system is a loser and obtains a negative value of the rewards. So, case (*i*) holds.

Similarly, if $m = n$, then the system wins (i.e., $Pr(v \models \mathrm{F}B) = 1$) and gains rewards. According to the property of the zero-sum game, the environment obtains a negative value of the rewards. So, case (*ii*) holds.

If $0 < m < n$, let $l = n - m$, then $MP(v \models \mathrm{F}B) = \frac{1}{m}P(\pi_m) - \frac{1}{n-m}P(\pi_{n-m}) = \frac{1}{m}P(\pi_m) - \frac{1}{l}P(\pi_l)$, where $m \neq 0$ and $l \neq 0$. Since $P(\pi_m) > 0, P(\pi_l) > 0$, $\frac{1}{m}P(\pi_m) > 0$ and $\frac{1}{l}P(\pi_l) > 0$, if $\frac{1}{m}P(\pi_m) > \frac{1}{l}P(\pi_l)$, then $MP(v \models \mathrm{F}B) > 0$, and the system obtains this value; if $\frac{1}{m}P(\pi_m) < \frac{1}{l}P(\pi_l)$, then $MP(v \models \mathrm{F}B) < 0$, and the environment obtains this value. So, case (*iii*) holds.

Given an MpAGP $\mathscr{G}^{\mathrm{M}}$, $v \in V$, this paper denotes the total payoff value of n plays starting from state v as $TP(v \models \mathrm{F}B)$, where $B \subseteq V$ is a set of states that satisfy φ, and φ is the GR(1) winning condition, $n = 1, 2, \ldots$. To determine whether the environment or the system obtains the mean payoff $MP(v \models \mathrm{F}B)$, this paper denotes the number of plays to reach a state in set B and not reach a state in the set B as m and l respectively, where $m + l = n$. Meanwhile, this paper uses $TP_1(v \models \mathrm{F}B)$ and $TP_2(v \models \mathrm{F}B)$ to denote the total payoffs of the m plays to reach a state in set B that satisfies φ and the total payoffs of the l plays to not reach a state in set B that satisfies φ, respectively. For example, if $l = 0$, it is said that the system obtains the mean payoff, and the environment obtains the negative of the mean payoff. The details of this algorithm are presented in Algorithm 2.

Input: A MpAPG $\mathscr{G}^{\mathrm{M}}$ with finite space V and a state set $B \subseteq V$, the target set T, the iteration numbers k
Output: the mean-payoffs $MP(v \models \mathsf{F}B)$ for all $v \in V$

For $v_i \in V, i = 1, 2, \cdots$
If $v_{i\_1} \in T$
  return 0
EndIf
Initialize $TP(v_i \models \mathsf{F}B) = 0, TP_1(v_i \models \mathsf{F}B) = 0,$
$TP_2(v_i \models \mathsf{F}B) = 0, m = 0,$ **and** $l = 0;$
  For $j = 1, 2, \cdots, k$
  Initialize $P(\pi_j) = 0, t = 0;$
  While(True)
  If $v_{i\_t} \in T$
    break
  EndIf
  If $v_{i\_t} \in V_E$
    $v_{i\_t+1} = \Theta_e(v_{i\_t}, a\_t);$
    $P(\pi_j) += \omega(v_{i\_t});$
  Else
    $v_{i\_t+1} = \Theta_s(v_{i\_t}, a\_t);$
    $P(\pi_j) += \omega(v_{i\_t});$
  EndIf
    $t = t + 1$
  EndWhile
  If $v_{i\_t} \in B$
    $TP(v_i \models \mathsf{F}B) += P(\pi_j);$
    $TP_1(v_i \models \mathsf{F}B) += P(\pi_j);$
    $m += 1;$
  Else
    $TP(v_i \models \mathsf{F}B) -= P(\pi_j);$
    $TP_2(v_i \models \mathsf{F}B) -= P(\pi_j);$
    $l += 1;$
  EndIf
  EndFor
  If$(l == 0)$
    all plays reach a state in B, the system gains the value of mean-payoff $TP(v_i \models \mathsf{F}B)/k$
  ElsIf $(m == 0)$
    all plays no reach a state in B, the environment gains the value of mean-payoff $TP(v_i \models \mathsf{F}B)/k$
  Else
    If$(TP(v_i \models \mathsf{F}B) >= 0)$
      the system get the value of mean-payoff $\frac{TP_1(v_i \models \mathsf{F}B)}{m} + \frac{TP_2(v_i \models \mathsf{F}B)}{l}$
    Else
      the environment gain the value of the mean-payoff $\frac{TP_1(v_i \models \mathsf{F}B)}{m} + \frac{TP_2(v_i \models \mathsf{F}B)}{l}$
    EndIf
    EndIf
EndFor

Algorithm 2. Path-MP: Computing the mean-payoffs of states v with $MP(v \models \mathrm{F}B)$

Expected mean payoffs for the system winning

Corollary 1 *If the system (environment) gains the mean payoff of* $MP(v \models \mathrm{F}B)$*, then it also gains the expected mean-payoff of* $Pr_f(v \models \mathrm{F}B) \cdot MP(v \models \mathrm{F}B)$.

From the content in the previous section, it is known that the expected mean payoff $EM(v \models \mathrm{F}B) = Pr_f(v \models \mathrm{F}B) \cdot MP(v \models \mathrm{F}B)$ is the weight of the probability of system winning and the mean payoffs. Since the probability of system winning in each state is not less than zero, the expected mean payoff is distributed in the same way as the mean payoff. Our previous work has studied the probability of the system winning, and two algorithms are proposed in this section to calculate the mean payoff. So, the expected mean payoff under the winning probability of the system can be obtained by the two algorithms, respectively.

Case studies

This section presents two experiments: In the first experiment, a simple scenario of autonomous driving was simulated; in the second experiment, a robot patrol scenario was simulated. For the two experiments, training was performed for a specified number of plays ($n = 500, 1000, 5000, 10{,}000$).

Autonomous driving

This is an autonomous driving scenario where the system safety problem is explored. Specifically, the expected mean payoff of the system was calculated with the probability of winning. This paper considers the MpAPG $\mathscr{G}^{\mathrm{M}}$ in Fig. 1 to illustrate the interaction between the car (the system) and the environment. The circle states are controlled by the environment, and the square states are controlled by the system. At the environment state, the environment can prevent the car from reaching the intersection safely with a traffic jam or a pedestrian, and these two actions are denoted as *a* and *b*, respectively. Assume that the road condition is normal, and the environment has taken action *e*. At this time, the system can choose to brake, honk, or change lanes, and these three actions are denoted as *c*, *d*, and *f*, respectively. Assume that the car is running normally and the system takes action *g*. The decimal labels represent the probabilities of the enabled actions, and the integers represent payoffs. The winning condition of this experiment can be expressed with the formula $\varphi = \mathrm{GF}J$, where *J* is an atomic proposition *acc*.

Let there be plays π_i (including loops plays) starting from v in V, where $i = 1, 2, \ldots, n$. Through preprocessing of game states, it can be found that the set $B = \{v_6, v_7\}$ satisfies the winning condition φ.

This paper takes $n = 500$, $n = 1000$, $n = 5000$, and $n = 10{,}000$ as examples and gives the mean payoffs and the expected mean payoffs for each state when $n = 10{,}000$. According to Algorithm 1, the value of the mean payoffs is calculated for the system as a vector $[2.74090, 12.97610, 0, 4.55850, 9.38110, -4.19210, 0, 0, 0]$. This is the mean payoff for each state in which the system satisfies the winning condition. Then, the value of the expected mean payoff for the system is a vector $[0.26834, 12.06868, 0, 2.23143, 9.18429, -0.0, 0, 0, 0]$. This is the expected mean payoff for each state in which the system has a probability of winning. For example, in the state v_0, the expected mean payoff of the system at the probability of winning is 0.26834. Since $0.26834 > 0$, the expected mean payoff of the environment in the state v_0 is -0.26834.

According to Algorithm 2, the value of the mean payoff for the system in each state is calculated as a vector $[10.63551, 6.30843, 0, 10.94951, -7.73282, -4.19210, 0, 0, 0]$. The value of the expected mean payoff for the system in each state is also a vector $[1.04124, 5.86728, 0, 5.35990, -7.57060, -0.0, 0, 0, 0]$.

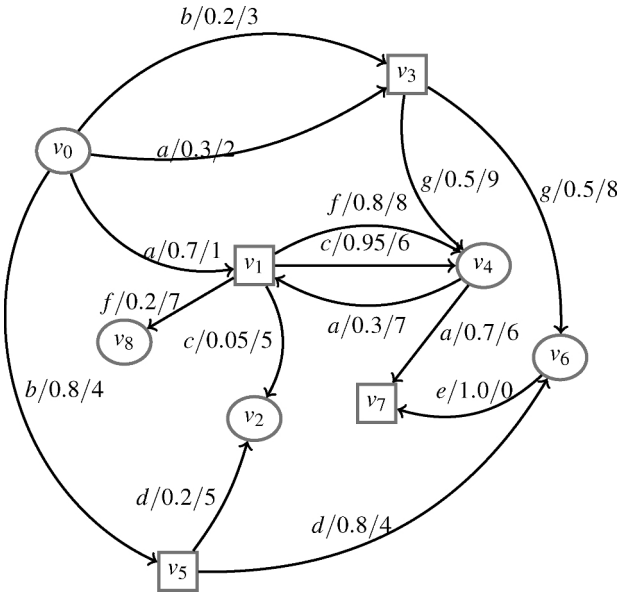

Fig. 1. This is a MpAPG $\mathscr{G}^{\mathrm{M}}$ between the system and the environment, where $AP = \{succ, acc\}$ is the set of atomic propositions.

Based on the results of autonomous safety driving, this paper adopts the mean error method to compare the convergence and volatility of the two methods. As shown in the following table (for simplicity of presentation, 5 decimal places are taken):

Robot patrol

This experiment considers an MpAPG graph abstracted from the robot patrol scenario. For an MpAPG $\mathscr{G}^{\mathrm{M}} = \langle AP, V, Act, \Theta_E, \Theta_S, L, W \rangle$, the robot (the system) and the system are two players. $\mathscr{G}^{\mathrm{M}}$ simulates the interaction between the two players, and the winning condition φ is the GR(1) formula. This paper uses $B \subseteq V$ to denote a set of states that satisfy the GR(1) formula φ. The set B is obtained by using the μ -calculus operators, and then the mean payoff $MP(v \models \mathrm{F}B)$ is calculated for the system in each state. Finally, the expected mean payoff $EM(v \models \mathrm{F}B)$ is calculated for the system with the probability of winning. As shown in Fig. 2, the circle states are controlled by the environment, and the square states are controlled by the system. The action set is $Act = \{a, b, c, d, e\}$, and the winning condition is the GR(1) formula $\varphi = ((\mathrm{GF}J_1^e \wedge \mathrm{GF}J_2^e) \Rightarrow (\mathrm{GF}J_1^s \wedge \mathrm{GF}J_2^s))$, where J_1^e, J_2^e, J_1^s, and J_1^s are four Boolean formulas. Let $T_1^e = \{v_8\}, T_2^e = \{v_{10}\}, T_1^s = \{v_7, v_{11}\}$ and $T_2^s = \{v_9, v_{13}\}$ be the sets satisfying J_1^e, J_2^e, J_1^s and J_1^s, respectively. The decimal labels represent the probabilities of the enabled actions, and the integers represent payoffs. After being calculated by the methods and algorithms[24], the set $B = \{v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{13}\}$. Again, this experiment assumes that there are plays π_i (including loops play) starting from v, where $i = 12, \ldots, n$ and $v \in V$.

This paper takes $n = 500$, $n = 1000$, $n = 5000$, and $n = 10{,}000$ as examples, and gives the mean payoffs and the expected mean payoffs for each state when $n = 10{,}000$. According to Algorithm 1, the mean payoff is calculated as $MP(v \models \mathrm{F}B) = [-1.85440, -2.17380, -2.95500, -1.59060, 0, 0, 0, 0, 0, 0, 0, 0, 0]$. This is the mean payoff for each state in which the system satisfies the winning condition. Then, the expected mean payoff is calculated for the system as a vector $[-1.30401, -1.38964, -0.18890, -1.14392, 0, 0, 0, 0, 0, 0, 0, 0, 0]$. This is the expected mean payoff for each state in which the system has a probability of winning. For example, in the state v_0, the expected mean payoff of the environment under the probability of system winning is -1.30401. Since -1.30401 is less than 0, the expected mean payoff of the system in the state v_0 is 1.30401.

According to Algorithm 2, the value of the mean payoff for each state is calculated as a vector $[-2.72562, -1.15064, 8.32250, -2.76553, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, and the expected mean payoff for each state is also a vector $[-1.91664, -0.73557, 0.53203, -1.98891, 0, 0, 0, 0, 0, 0, 0, 0, 0]$.

Similarly, this paper adopts the mean error method to compare the convergence and volatility of the two methods based on the results of robot patrol. As shown in the following table (for the simplicity of presentation, 5 decimal places are taken):

Results

The results in Table 1 indicate that as the number of plays increases, the convergence rate becomes higher, and the volatility becomes smaller. In the same number of plays, the State-MP algorithm converges faster than the Path-MP algorithm, and the volatility of the State-MP algorithm is smaller than that of the Path-MP algorithm.

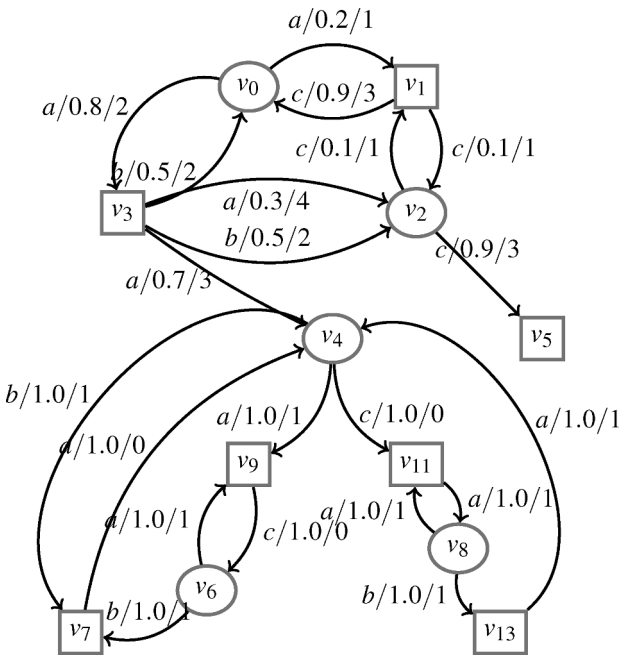

Fig. 2. An MpAPG $\mathscr{G}^{\mathrm{M}}$ about robot patrol.

Plays	State-MP		Path-MP	
	Convergence	Volatility	Convergence	Volatility
500	2.49128	2.89005	11.70143	19.09528
1000	1.66440	1.14389	3.51079	6.25358
5000	0.41922	0.27873	1.68187	1.21162
10,000	0.23342	0.11194	0.71252	0.62127

Table 1. Experiment result comparison on algorithms.

Plays	State-MP		Path-MP	
	Convergence	Volatility	Convergence	Volatility
500	1.54447	1.50095	1.68177	4.31239
1000	1.00454	0.70382	1.15800	1.45370
5000	0.19077	0.11516	0.66301	0.19486
10,000	0.06170	0.07388	0.13485	0.18761

Table 2. Experiment result comparison on algorithms.

In Table 2, whether by vertical or horizontal comparison, it can be observed that the State-MP algorithm converges faster and is more stable than the Path-MP algorithm. Meanwhile, it is indicated that the probability of the system winning is the value of the expected mean payoff of the system winning when the mean payoffs are 1 for each state.

Conclusion

In this paper, the expected mean payoff of the system with a winning probability is investigated, and symbolic algorithms are proposed for system synthesis with a quantitative objective. Specifically, a probabilistic model MpAPG is considered, which combines the asynchronous probability game (APG) with the GR(1) winning condition and the mean-payoff game. The model also has the property of a zero-sum game, i.e., in the game, the sum of the two players' payoffs is zero in each state. Meanwhile, two algorithms are designed to solve the mean payoff for each state, namely by the State-MP algorithm based on the state properties and the Path-MP algorithm based on the path properties. In the State-MP algorithm, this paper directly considers the mean-payoff problem; in the Path-MP algorithm, this paper analyzes whether the play can reach the state in the set that satisfies the winning condition. The experimental evaluation results indicate that the two proposed algorithms are effective, and the State-MP algorithm is more stable and converges faster than the Path-MP algorithm.

In future work, we will extend our theoretical approach to a wider range of practical applications. Also, we will consider the time constraint and focus on probabilistic synthesis with the time constraint and its applications.

Data availability

The authors declare that the data supporting the findings of this study are available within the paper and its Supplementary Information files. Should any raw data files be needed in another format, they are available from the corresponding author upon reasonable request.

Code availability

The code that supports the findings of this study is available from the corresponding authors upon reasonable request.

Received: 31 August 2024; Accepted: 3 January 2025
Published online: 17 January 2025

References

1. Henzinger, T. A. Quantitative reactive models. In *Model Driven Engineering Languages and Systems—15th International Conference, MODELS 2012, Innsbruck, Austria, September 30–October 5, 2012. Proceedings. Lecture Notes in Computer Science* Vol. 7590 (eds France, R. B. et al.) 1–2 (Springer, 2012). https://doi.org/10.1007/978-3-642-33666-9_1.
2. Hunter, P., Pauly, A., Pérez, G. A. & Raskin, J. Mean-payoff games with partial observation. *Theor. Comput. Sci.* **735**, 82–110. https://doi.org/10.1016/j.tcs.2017.03.038 (2018).
3. Chatterjee, K., Doyen, L., Gimbert, H. & Oualhadj, Y. Perfect-information stochastic mean-payoff parity games. In *Foundations of Software Science and Computation Structures—17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5–13, 2014, Proceedings. Lecture Notes in Computer Science* Vol. 8412 (ed. Muscholl, A.) 210–225 (Springer, 2014). https://doi.org/10.1007/978-3-642-54830-7_14.
4. Velner, Y. et al. The complexity of multi-mean-payoff and multi-energy games. *Inf. Comput.* **241**, 177–196. https://doi.org/10.1016/j.ic.2015.03.001 (2015).
5. Brim, L., Chaloupka, J., Doyen, L., Gentilini, R. & Raskin, J. Faster algorithms for mean-payoff games. *Formal Methods Syst. Des.* **38**, 97–118. https://doi.org/10.1007/s10703-010-0105-x (2011).

6. Björklund, H., Sandberg, S. & Vorobyov, S. G. Memoryless determinacy of parity and mean payoff games: a simple proof. *Theor. Comput. Sci.* **310**, 365–378. https://doi.org/10.1016/S0304-3975(03)00427-4 (2004).
7. Zwick, U. & Paterson, M. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.* **158**, 343–359. https://doi.org/10.1016/0304-3975(95)00188-3 (1996).
8. Bruyère, V., Filiot, E., Randour, M. & Raskin, J. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Inf. Comput.* **254**, 259–295. https://doi.org/10.1016/j.ic.2016.10.011 (2017).
9. Church, A. Logic, arithmetic, and automata. *J. Symb. Logic* **29**, 210 (1964).
10. Pnueli, A. & Rosner, R. On the synthesis of an asynchronous reactive module. In *Automata, Languages and Programming, 16th International Colloquium, ICALP89, Stresa, Italy, July 11–15, 1989, Proceedings. Lecture Notes in Computer Science* Vol. 372 (eds Ausiello, G. et al.) 652–671 (Springer, 1989). https://doi.org/10.1007/BFb0035790.
11. Wallmeier, N., Hütten, P. & Thomas, W. Symbolic synthesis of finite-state controllers for request-response specifications. In *Implementation and Application of Automata, 8th International Conference, CIAA 2003, Santa Barbara, California, USA, July 16–18, 2003, Proceedings. Lecture Notes in Computer Science* Vol. 2759 (eds Ibarra, O. H. & Dang, Z.) 11–22 (Springer, 2003). https://doi.org/10.1007/3-540-45089-0_3.
12. Alur, R. & Torre, S. L. Deterministic generators and games for ltl fragments. *ACM Trans. Comput. Log.* **5**, 1–25. https://doi.org/10.1145/963927.963928 (2004).
13. Harding, A., Ryan, M. & Schobbens, P. A new algorithm for strategy synthesis in LTL games. In *Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, TACAS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4–8, 2005, Proceedings. Lecture Notes in Computer Science* Vol. 3440 (eds Halbwachs, N. & Zuck, L. D.) 477–492 (Springer, 2005). https://doi.org/10.1007/978-3-540-31980-1_31.
14. Jobstmann, B., Griesmayer, A. & Bloem, R. Program repair as a game. In *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6–10, 2005, Proceedings. Lecture Notes in Computer Science* Vol. 3576 (eds Etessami, K. & Rajamani, S. K.) 226–238 (Springer, 2005). https://doi.org/10.1007/11513988_23.
15. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A. & Sa'ar, Y. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.* **78**, 911–938. https://doi.org/10.1016/j.jcss.2011.08.007 (2012).
16. Ehrenfeucht, A. & Mycielski, J. Positional strategies for mean payoff games. *Int. J. Game Theory* **8**, 109–113 (1979).
17. Ummels, M. & Wojtczak, D. The complexity of Nash equilibria in limit-average games. In *CONCUR 2011—Concurrency Theory—22nd International Conference, CONCUR 2011, Aachen, Germany, September 6–9, 2011. Proceedings. Lecture Notes in Computer Science* Vol. 6901 (eds Katoen, J. & König, B.) 482–496 (Springer, 2011). https://doi.org/10.1007/978-3-642-23217-6_32.
18. Bouyer, P., Brenguier, R. & Markey, N. Nash equilibria for reachability objectives in multi-player timed games. In *CONCUR 2010—Concurrency Theory, 21st International Conference, CONCUR 2010, Paris, France, August 31–September 3, 2010. Proceedings. Lecture Notes in Computer Science* Vol. 6269 (eds Gastin, P. & Laroussinie, F.) 192–206 (Springer, 2010). https://doi.org/10.1007/978-3-642-15375-4_14.
19. Chatterjee, K., Henzinger, T. A. & Jurdzinski, M. Mean-payoff parity games. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings* 178–187 (IEEE Computer Society, 2005). https://doi.org/10.1109/LICS.2005.26
20. Chatterjee, K., Komárková, Z. & Kretínský, J. Unifying two views on multiple mean-payoff objectives in Markov decision processes. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6–10, 2015* 244–256 (IEEE Computer Society, 2015). https://doi.org/10.1109/LICS.2015.32
21. Almagor, S., Kupferman, O. & Velner, Y. Minimizing expected cost under hard Boolean constraints, with applications to quantitative synthesis. In *27th International Conference on Concurrency Theory, CONCUR 2016, August 23–26, 2016, Québec City, Canada. LIPIcs* Vol. 59 (eds Desharnais, J. & Jagadeesan, R.) 9:1-9:15 (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016). https://doi.org/10.4230/LIPIcs.CONCUR.2016.9.
22. Jiménez-Lizárraga, M., Escobedo-Trujillo, B. A. & López-Barrientos, J. D. Mixed deterministic and stochastic disturbances in a discrete-time Nash game. *Int. J. Syst. Sci.*[SPACE]https://doi.org/10.1007/978-3-642-22993-0_21 (2024).
23. Gutierrez, J., Steeples, T. & Wooldridge, M. Mean-payoff games with ω-regular specifications. *Games* **13**, 19 (2022).
24. Clemente, L. & Raskin, J. Multidimensional beyond worst-case and almost-sure problems for mean-payoff objectives. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6–10, 2015* 257–268 (IEEE Computer Society, 2015). https://doi.org/10.1109/LICS.2015.33
25. Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (Wiley, 2014).
26. Chatterjee, K. & Doyen, L. Energy and mean-payoff parity Markov decision processes. In *Mathematical Foundations of Computer Science 2011—36th International Symposium, MFCS 2011, Warsaw, Poland, August 22–26, 2011. Proceedings. Lecture Notes in Computer Science* Vol. 3907 (eds Murlak, F. & Sankowski, P.) 206–218 (Springer, 2011). https://doi.org/10.1007/978-3-642-22993-0_21.
27. López-Barrientos, J. D., Jiménez-Lizárraga, M. & Escobedo-Trujillo, B. A. On the discrete-time minimum principle in multiple-mode systems. *Cybern. Syst.*[SPACE]https://doi.org/10.1080/01969722.2023.2175492 (2023).
28. Zhao, W., Li, R., Liu, W., Dong, W. & Liu, Z. Probabilistic synthesis against GR(1) winning condition. *Front. Comput. Sci.* **16**, 162203. https://doi.org/10.1007/s11704-020-0076-z (2022).

Author contributions

W.Z. conceived the project. W.W.L. performed the numerical simulations. Z.M.L. and T.X.W analyzed the data and interpreted the results, and proposed improvements. W.Z. and Z.M.L. both contributed to the writing and editing of the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to W.Z. or Z.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.