Logical Methods in Computer Science
Volume 21, Issue 2, 2025, pp. 19:1–19:41
https://lmcs.episciences.org/

Submitted May 14, 2024
Published Jun. 05, 2025

STOCHASTIC WINDOW MEAN-PAYOFF GAMES

LAURENT DOYEN [a], PRANSHU GABA [b], AND SHIBASHIS GUHA [b]

[a] CNRS & LMF, ENS Paris-Saclay, Gif-sur-Yvette, France
e-mail address: ldoyen@lmf.cnrs.fr

[b] Tata Institute of Fundamental Research, Mumbai, India
e-mail address: pranshu.gaba@tifr.res.in, shibashis@tifr.res.in

Abstract. Stochastic two-player games model systems with an environment that is both adversarial and stochastic. The adversarial part of the environment is modeled by a player (Player 2) who tries to prevent the system (Player 1) from achieving its objective. We consider finitary versions of the traditional mean-payoff objective, replacing the long-run average of the payoffs by payoff average computed over a finite sliding window. Two variants have been considered: in one variant, the maximum window length is fixed and given, while in the other, it is not fixed but is required to be bounded. For both variants, we present complexity bounds and algorithmic solutions for computing strategies for Player 1 to ensure that the objective is satisfied with positive probability, with probability 1, or with probability at least p, regardless of the strategy of Player 2. The solution crucially relies on a reduction to the special case of non-stochastic two-player games. We give a general characterization of prefix-independent objectives for which this reduction holds. The memory requirement for both players in stochastic games is also the same as in non-stochastic games by our reduction. Moreover, for non-stochastic games, we improve upon the upper bound for the memory requirement of Player 1 and upon the lower bound for the memory requirement of Player 2.

1. Introduction

We consider two-player turn-based stochastic games played on graphs. Games are a central model in computer science, in particular for the verification and synthesis of reactive systems [GTW02, CH12, FV96]. A stochastic game is played by two players on a graph with stochastic transitions, where the players represent the system and the adversarial environment, while the objective represents the functional requirement that the synthesized system aims to satisfy with a probability p as large as possible. The vertices of the graph are partitioned into system, environment, and probabilistic vertices. A stochastic game is played in infinitely many rounds, starting by initially placing a token on some vertex. In every round, if the token is on a system or an environment vertex, then the owner of the vertex chooses a successor vertex; if the token is on a probabilistic vertex, then the

Key words and phrases: stochastic games, finitary objectives, mean-payoff, reactive synthesis.

This work is partially supported by the Indian Science and Engineering Research Board (SERB) grant SRG/2021/000466 and by the Indo-French Centre for the Promotion of Advanced Research (IFCPAR).

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.46298/LMCS-21(2:19)2025

successor vertex is chosen according to a given probability distribution. The token moves to the successor vertex, from where the next round starts. The outcome is an infinite sequence of vertices, which is winning for the system if it satisfies the given objective. The associated *quantitative satisfaction problem* is to decide, given a threshold p, whether the system can win with probability at least p. The *almost-sure problem* is the special case where $p = 1$, and the *positive problem* is to decide whether the system can win with positive probability. The almost-sure and the positive problems are referred to as the *qualitative satisfaction problems*. When the answer to these decision problems is Yes, it is useful to construct a winning strategy for the system that can be used as a model for an implementation that ensures the objective be satisfied with the given probability.

Traditional objectives in stochastic games are ω-regular such as reachability, safety, and parity objectives [CH12], or quantitative such as mean-payoff objectives [EM79, ZP96]. For example, a parity objective may specify that every request of the environment is eventually granted by the system, and a mean-payoff objective may specify the long-run average power consumption of the system. A well-known drawback of parity and mean-payoff objectives is that only the long-run behaviour of the system may be specified [AH94, CHH09a, HTWZ15], allowing weird transient behaviour: for example, a system may grant all its requests but with an unbounded response time; or a system with long-run average power consumption below some threshold may exhibit arbitrarily long (but finite) sequences with average power consumption above the threshold. This limitation has led to considering finitary versions of those objectives [CHH09a, KPV09, CDRR15], where the sequences of undesired transient behaviours must be of fixed or bounded length. Window mean-payoff objectives [CDRR15] are quantitative finitary objectives that strengthen the traditional mean-payoff objective: the satisfaction of a window mean-payoff objective implies the satisfaction of the standard mean-payoff objective. Given a length $\ell \geq 1$, the fixed window mean-payoff objective ($\mathsf{FWMP}(\ell)$) is satisfied if except for a finite prefix, from every point in the play, there exists a window of length at most ℓ starting from that point such that the mean payoff of the window is at least a given threshold. In the bounded window mean-payoff objective (BWMP), it is sufficient that there exists some length ℓ for which the $\mathsf{FWMP}(\ell)$ objective is satisfied.

Contributions. We present algorithmic solutions for stochastic games with window mean-payoff objectives, and show that the positive and almost-sure problems are solvable in polynomial time for $\mathsf{FWMP}(\ell)$ (Theorem 6.5), and in $\mathsf{NP} \cap \mathsf{coNP}$ for BWMP (Theorem 6.9). The complexity result for the almost-sure problem entails that the quantitative satisfaction problem is in $\mathsf{NP} \cap \mathsf{coNP}$ (for both the fixed and bounded version), using standard techniques for solving stochastic games with prefix-independent objectives [CHH09b]. Note that the $\mathsf{NP} \cap \mathsf{coNP}$ bound for the quantitative satisfaction problem matches the special case of reachability objectives in simple stochastic games [Con92], and thus would require a major breakthrough to be improved.

As a consequence, using the $\mathsf{FWMP}(\ell)$ objective instead of the standard mean-payoff objective provides a stronger guarantee on the system, and even with a polynomial complexity for the positive and the almost-sure problems (which is not known for mean-payoff objectives), and at no additional computational cost for the quantitative satisfaction problem. The solution relies on a reduction to non-stochastic two-player games (stochastic games without probabilistic vertices). The key result is to show that in order to win positively from some vertex of the game graph, it is necessary to win from some vertex of the non-stochastic

game obtained by transforming all probabilistic vertices into adversarial vertices. This condition, which we call the sure-almost-sure (SAS) property (Definition 5.1), was used to solve finitary Streett objectives [CHH09b]. We follow a similar approach and generalize it to prefix-independent objectives satisfying the SAS property (Theorem 5.3). The bounds on the memory requirement of optimal strategies of Player 1 can also be derived from the key result, and are the same as optimal bounds for non-stochastic games. For FWMP(ℓ) and BWMP objectives in particular, we show that the memory requirement of Player 2 is also no more than the optimal memory required by winning strategies in non-stochastic games.

As solving a stochastic game with a prefix-independent objective φ reduces to solving non-stochastic games with objective φ and showing that φ satisfies the SAS property, we show that the FWMP(ℓ) and BWMP objectives satisfy the SAS property (Lemma 6.1, Lemma 6.8) and rely on the solution of non-stochastic games with these objectives [CDRR15] to complete the reduction.

We improve the memory bounds for optimal strategies of both players in non-stochastic games. It is stated in [CDRR15] that $|V| \cdot \ell$ memory suffices for both players, where $|V|$ and ℓ are the number of vertices and the window length respectively. In [BHR16b, Theorem 2], the bound is loosened to $\mathcal{O}(w_{\max} \cdot \ell^2)$ and $\mathcal{O}(w_{\max} \cdot \ell^2 \cdot |V|)$ for Player 1 and Player 2 respectively, where $w_{\max}$ is the maximum absolute payoff in the graph, as the original tighter bounds [CDRR15] were stated without proof. Since the payoffs are given in binary, this is exponential in the size of the input. In contrast, we tighten the bounds stated in [CDRR15]. We show that for Player 1, memory ℓ suffices (Theorem 4.4), and improve the bound on memory of Player 2 strategies as follows. We compute the set W of vertices from which Player 2 can ensure that the mean payoff remains negative for ℓ steps, as well as the vertices from which Player 2 can ensure that the game reaches W. These vertices are identified recursively on successive subgames of the original input game. If k is the number of recursive calls, then we show that $k \cdot \ell$ memory suffices for Player 2 to play optimally (Theorem 4.8). Note that $k \leq |V|$. We also provide a lower bound on the memory size for Player 2. Given $\ell \geq 2$, for every $k \geq 1$, we construct a graph with a set V of vertices such that Player 2 requires at least $k + 1 = \frac{1}{2}(|V| - \ell + 3)$ memory to play optimally (Theorem 4.13). This is an improvement over the result in [CDRR15] which showed that memoryless strategies do not suffice for Player 2. Our result is quite counterintuitive since given an open window (a window in which every prefix has a total weight less than 0) that needs to be kept open for another $j \leq \ell$ steps from a vertex v, one would conjecture that it is sufficient for a Player 2 winning strategy to choose an edge from v that leads to the minimum payoff over paths of length j. Thus for every j, Player 2 should choose a fixed edge and hence memory of size ℓ should suffice. However, we show that this is not the case.

To the best of our knowledge, this work leads to the first study of stochastic games with finitary quantitative objectives.

Related work. Window mean-payoff objectives were first introduced in [CDRR15] for non-stochastic games, where solving FWMP(ℓ) was shown to be in PTIME and BWMP in NP$\cap$coNP. These have also been studied for Markov decision processes (MDPs) in [BDOR20, BGR19]. In [BDOR20], a threshold probability problem has been studied, while in [BGR19], the authors studied the problem of maximising the expected value of the window mean-payoff. Positive, almost-sure, and quantitative satisfaction of BWMP in MDPs are in NP $\cap$ coNP [BDOR20], the same as in non-stochastic games.

Parity objectives can be viewed as a special case of mean-payoff [Jur98]. A bounded-window parity objective has been studied in [Hor07, CHH09a] where a play satisfies the objective if from some point on, there exists a bound ℓ such that from every state with an odd priority a smaller even priority occurs within at most ℓ steps. Non-stochastic games with bounded window parity objectives can be solved in PTIME [Hor07, CHH09a]. Stochastic games with bounded window parity objectives have been studied in [CHH09b] where the positive and almost-sure problems are in PTIME while the quantitative satisfaction problem is in $\mathsf{NP} \cap \mathsf{coNP}$. A fixed version of the window parity objective has been studied for two-player games and shown to be PSPACE-complete [WZ17]. Another window parity objective has been studied in [BHR16a] for which both the fixed and the bounded variants have been shown to be in PTIME for non-stochastic games. The threshold problem for this objective has also been studied in the context of MDPs, and both fixed and bounded variants are in PTIME [BDOR20]. Finally, synthesis for *bounded* eventuality properties in LTL is 2-$\mathsf{EXPTIME}$-complete [KPV09].

Outline. In Section 2, we provide the necessary technical preliminaries. In Section 3, we define window mean-payoff objectives and state relevant decision problems for stochastic games. In Section 4, we give improved bounds on the memory requirements of the players' strategies for non-stochastic games with window mean-payoff objectives. In Section 5, we define a property of prefix-independent objectives that allows one to solve stochastic games by reducing them to non-stochastic games. Finally, as an application, in Section 6, we provide solutions to problems in stochastic games with window mean-payoff objectives.

2. Preliminaries

Probability distributions. A *probability distribution* over a finite nonempty set A is a function $\mathsf{Pr}\colon A \to [0,1]$ such that $\sum_{a \in A} \mathsf{Pr}(a) = 1$. The *support* of a probability distribution Pr over A, denoted by $\mathsf{Supp}(\mathsf{Pr})$, is the set of all elements a in A such that $\mathsf{Pr}(a) > 0$. We denote by $\mathcal{D}(A)$ the set of all probability distributions over A. For the algorithmic and complexity results, we assume that probabilities are given as rational numbers.

Stochastic games. We consider two-player turn-based zero-sum stochastic games (or simply, stochastic games in the sequel). The two players are referred to as Player 1 and Player 2. A *stochastic game* is a weighted directed graph $\mathcal{G} = ((V,E),(V_1,V_2,V_\Diamond),\mathbb{P},w)$, where:

- (V,E) is a directed graph with a set V of vertices and a set $E \subseteq V \times V$ of directed edges such that for all vertices $v \in V$, the set $E(v) = \{v' \in V \mid (v,v') \in E\}$ of out-neighbours of v is nonempty, i.e., $E(v) \neq \varnothing$ (no deadlocks). A stochastic game is said to be finite if V is a finite set, and infinite otherwise. Unless mentioned otherwise, stochastic games considered in this paper are finite;
- $(V_1,V_2,V_\Diamond)$ is a partition of V. The vertices in V_1 belong to Player 1, the vertices in V_2 belong to Player 2, and the vertices in $V_\Diamond$ are called probabilistic vertices;

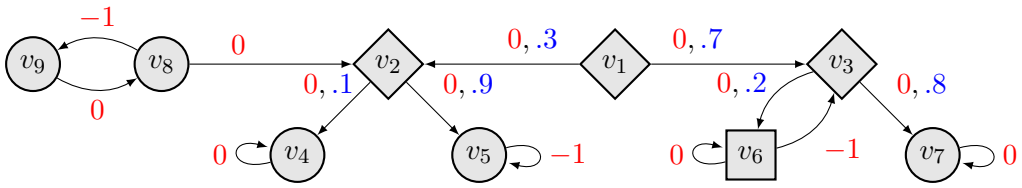


FIGURE 1. A stochastic game. Player 1 vertices are denoted by circles, Player 2 vertices are denoted by boxes, and probabilistic vertices are denoted by diamonds. The payoff for each edge is shown in red and probability distribution out of probabilistic vertices is shown in blue.

- $\mathbb{P}\colon V_\Diamond \to \mathcal{D}(V)$ is a *probability function* that describes the behaviour of probabilistic vertices in the game. It maps the probabilistic vertices $v \in V_\Diamond$ to a probability distribution $\mathbb{P}(v)$ over the set $E(v)$ of out-neighbours of v such that $\mathbb{P}(v)(v') > 0$ for all $v' \in E(v)$ (i.e., all neighbours have nonzero probability);
- $w\colon E \to \mathbb{Q}$ is a *payoff function* assigning a rational payoff to every edge in the game.

Stochastic games are played in rounds. The game starts by initially placing a token on some vertex. At the beginning of a round, if the token is on a vertex v, and $v \in V_i$ for $i \in \{1, 2\}$, then Player i chooses an out-neighbour $v' \in E(v)$; otherwise $v \in V_\Diamond$, and an out-neighbour $v' \in E(v)$ is chosen with probability $\mathbb{P}(v)(v')$. Player 1 receives from Player 2 the amount $w(v, v')$ given by the payoff function, and the token moves to v' for the next round. This continues ad infinitum resulting in an infinite sequence $\pi = v_0 v_1 v_2 \cdots \in V^\omega$ such that $(v_i, v_{i+1}) \in E$ for all $i \geq 0$.

A stochastic game with $V_\Diamond = \varnothing$ is called a non-stochastic two-player game, a stochastic game with $V_2 = \varnothing$ is called a Markov decision process (MDP), a stochastic game with $V_2 = V_\Diamond = \varnothing$ is called a one-player game, and a stochastic game with $V_1 = V_2 = \varnothing$ is called a Markov chain. We use pronouns "she/her" for Player 1 and "he/him" for Player 2. Figure 1 shows an example of a stochastic game. In figures, Player 1 vertices are shown as circles, Player 2 vertices as boxes, and probabilistic vertices as diamonds.

Plays and prefixes. A *play* in $\mathcal{G}$ is an infinite sequence $\pi = v_0 v_1 \cdots \in V^\omega$ of vertices such that $(v_i, v_{i+1}) \in E$ for all $i \geq 0$. We denote by $\mathrm{occ}(\pi)$ the set of vertices in V that occur at least once in π, and by $\inf(\pi)$ the set of vertices in V that occur infinitely often in π. For $i < j$, we denote by $\pi(i, j)$ the *infix* $v_i v_{i+1} \cdots v_j$ of π. Its length is $|\pi(i, j)| = j - i$, the number of edges. We denote by $\pi(0, j)$ the finite *prefix* $v_0 v_1 \cdots v_j$ of π, and by $\pi(i, \infty)$ the infinite *suffix* $v_i v_{i+1} \ldots$ of π. We denote by $\mathsf{Plays}_\mathcal{G}$ and $\mathsf{Prefs}_\mathcal{G}$ the set of all plays and the set of all prefixes in $\mathcal{G}$ respectively; the symbol $\mathcal{G}$ is omitted when it can easily be derived from the context. We denote by $\mathsf{Last}(\rho)$ the last vertex of a prefix $\rho \in \mathsf{Prefs}_\mathcal{G}$. We denote by $\mathsf{Prefs}^i_\mathcal{G}$ ($i \in \{1, 2\}$) the set of all prefixes ρ such that $\mathsf{Last}(\rho) \in V_i$. The *cone* at ρ is the set $\mathsf{Cone}(\rho) = \{\pi \in \mathsf{Plays}_\mathcal{G} \mid \rho \text{ is a prefix of } \pi\}$, the set of all plays having ρ as a prefix.

Objectives. An *objective* φ is a Borel-measurable subset of $\mathsf{Plays}_\mathcal{G}$ [BK08]. A play $\pi \in \mathsf{Plays}_\mathcal{G}$ *satisfies* an objective φ if $\pi \in \varphi$. In a (zero-sum) stochastic game $\mathcal{G}$ with objective φ, the objective of Player 1 is φ, and the objective of Player 2 is the complement set $\overline{\varphi} = \mathsf{Plays}_\mathcal{G} \setminus \varphi$. Given $T \subseteq V$, we define some qualitative objectives:

- the *reachability* objective $\mathsf{Reach}_{\mathcal{G}}(T) = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid T \cap \mathrm{occ}(\pi) \neq \varnothing\}$, the set of all plays that visit a vertex in T,
- the dual *safety* objective $\mathsf{Safe}_{\mathcal{G}}(T) = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid \mathrm{occ}(\pi) \subseteq T\}$, the set of all plays that never visit a vertex outside T,
- the *Büchi* objective $\mathsf{Büchi}_{\mathcal{G}}(T) = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid T \cap \inf(\pi) \neq \varnothing\}$, the set of all plays that visit a vertex in T infinitely often, and
- the dual *coBüchi* objective $\mathsf{coBüchi}_{\mathcal{G}}(T) = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid \inf(\pi) \subseteq T\}$, the set of all plays that eventually only visit vertices in T.

An objective φ is *closed under suffixes* if for all plays π satisfying φ, all suffixes of π also satisfy φ, that is, $\pi(j,\infty) \in \varphi$ for all $j \geq 0$. An objective φ is *closed under prefixes* if for all plays π satisfying φ, for all prefixes ρ such that the concatenation $\rho \cdot \pi$ is a play in $\mathcal{G}$, i.e., $\rho \cdot \pi \in \mathsf{Plays}_{\mathcal{G}}$, we have that $\rho \cdot \pi \in \varphi$. An objective φ is *prefix-independent* if it is closed under both prefixes and suffixes. An objective φ is closed under suffixes if and only if the complement objective $\overline{\varphi}$ is closed under prefixes. Thus, an objective φ is prefix-independent if and only if its complement $\overline{\varphi}$ is prefix-independent. The reachability objective is closed under prefixes, the safety objective is closed under suffixes, and the Büchi and coBüchi objectives are closed under both prefixes and suffixes.

Strategies. A *strategy* for Player $i \in \{1,2\}$ in a game $\mathcal{G}$ is a function $\sigma_i : \mathsf{Prefs}^i_{\mathcal{G}} \to \mathcal{D}(V)$ that maps prefixes ending in a vertex $v \in V_i$ to a probability distribution over the out-neighbours of v. That is, a strategy prescribes a randomized move for the player, taking into account the history seen so far. A strategy σ_i is *deterministic* (or *pure*) if for all prefixes $\rho \in \mathsf{Prefs}^i_{\mathcal{G}}$, the support $\mathsf{Supp}(\sigma_i(\rho))$ is a singleton, that is, $\sigma_i(\rho)$ is a single vertex with probability 1. A deterministic strategy σ_i can be viewed as a function $\sigma_i : \mathsf{Prefs}^i_{\mathcal{G}} \to V$. Unless mentioned otherwise, the strategies considered in this paper are deterministic.

The set of all strategies of Player $i \in \{1,2\}$ in the game $\mathcal{G}$ is denoted by $\Lambda_i(\mathcal{G})$, or Λ_i when $\mathcal{G}$ is clear from the context. Strategies can be realized as the output of a (possibly infinite-state) Mealy machine. A *Mealy machine* is a transition system with transitions labeled by a pair of symbols: one from the input alphabet and one from an output alphabet. For each state q of the Mealy machine and every letter a of the input alphabet, there is exactly one transition defined from state q on reading the letter a. Formally, a Mealy machine M is a tuple $(Q, q_0, \Sigma_i, \Sigma_o, \Delta, \delta)$ where Q is the set of states of M (the memory of the induced strategy), $q_0 \in Q$ is the initial state, Σ_i is the input alphabet, Σ_o is the output alphabet, $\Delta\colon Q \times \Sigma_i \to Q$ is a transition function that reads the current state of M and an input letter and returns the next state of M, and $\delta\colon Q \times \Sigma_i \to \Sigma_o$ is an output function that reads the current state of M and an input letter and returns an output letter.

The transition function Δ can be extended to a function $\hat{\Delta}\colon Q \times \Sigma_i^+ \to Q$ that reads words and can be defined inductively by $\hat{\Delta}(q,a) = \Delta(q,a)$ and $\hat{\Delta}(q, x \cdot a) = \Delta(\hat{\Delta}(q,x), a)$, for $q \in Q$, $x \in \Sigma_i^+$, and $a \in \Sigma_i$. The output function δ can be also be similarly extended to a function $\hat{\delta}\colon Q \times \Sigma_i^+ \to \Sigma_o$ on words and can be defined inductively by $\hat{\delta}(q,a) = \delta(q,a)$ and $\hat{\delta}(q, x \cdot a) = \delta(\hat{\Delta}(q,x), a)$, for $q \in Q$, $x \in \Sigma_i^+$, and $a \in \Sigma_i$.

A player's strategy can be defined by a Mealy machine whose input and output alphabets are V and $V \cup \{\epsilon\}$ respectively. For $i \in \{1,2\}$, a strategy σ_i of Player i can be defined by a Mealy machine $(Q, q_0, V, V \cup \{\epsilon\}, \Delta, \delta)$ as follows: Given a prefix $\rho \in \mathsf{Prefs}^i_{\mathcal{G}}$ ending in a Player i vertex, the strategy σ_i defined by a Mealy machine is $\sigma_i(\rho) = \hat{\delta}(q_0, \rho)$. Intuitively,

in each turn, if the token is on a vertex v that belongs to Player i for $i \in \{1,2\}$, then v is given as input to the Mealy machine, and the Mealy machine outputs the successor vertex of v that Player i must choose. Otherwise, the token is on a vertex v that either belongs to Player i's opponent or is a probabilistic vertex, in which case, the Mealy machine outputs the symbol ϵ to denote that Player i cannot decide the successor vertex of v. The *memory size* of a strategy σ_i is the smallest number of states a Mealy machine defining σ_i can have. A strategy σ_i is *memoryless* if $\sigma_i(\rho)$ only depends on the last element of the prefix ρ, that is for all prefixes $\rho, \rho' \in \mathsf{Prefs}^i_{\mathcal{G}}$ if $\mathsf{Last}(\rho) = \mathsf{Last}(\rho')$, then $\sigma_i(\rho) = \sigma_i(\rho')$. Memoryless strategies can be defined by Mealy machines with only one state.

A strategy *profile* $\overline{\sigma} = (\sigma_1, \sigma_2)$ is a pair of strategies $\sigma_1 \in \Lambda_1(\mathcal{G})$ and $\sigma_2 \in \Lambda_2(\mathcal{G})$. A play $\pi = v_0 v_1 \cdots$ is *consistent* with a strategy $\sigma_i \in \Lambda_i$ ($i \in \{1,2\}$) if for all $j \geq 0$, we have that if $v_j \in V_i$, then $v_{j+1} = \sigma_i(\pi(0,j))$. A play π is an *outcome* of a profile $\overline{\sigma} = (\sigma_1, \sigma_2)$ if it is consistent with both σ_1 and σ_2. We denote by $\mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\varphi)$ the probability that an outcome of the profile (σ_1, σ_2) in $\mathcal{G}$ with initial vertex v satisfies φ. First, we define this probability measure over cones inductively as follows. If $|\rho| = 0$, then ρ is just a vertex v_0, and $\mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\mathsf{Cone}(\rho))$ is 1 if $v = v_0$, and 0 otherwise. For the inductive case $|\rho| > 0$, there exist $\rho' \in \mathsf{Prefs}_{\mathcal{G}}$ and $v' \in V$ such that $\rho = \rho' \cdot v'$, and we have

$$\mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\mathsf{Cone}(\rho' \cdot v')) = \begin{cases} \mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\mathsf{Cone}(\rho')) \cdot \mathbb{P}(\mathsf{Last}(\rho'))(v') & \text{if } \mathsf{Last}(\rho') \in V_\Diamond, \\ \mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\mathsf{Cone}(\rho')) & \text{if } \mathsf{Last}(\rho') \in V_i \text{ and } \sigma_i(\rho') = v', \\ 0 & \text{otherwise.} \end{cases}$$

It is sufficient to define $\mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\varphi)$ on cones in $\mathcal{G}$ since a measure defined on cones extends to a unique measure on $\mathsf{Plays}_{\mathcal{G}}$ by Carathéodory's extension theorem [Bil86].

Non-stochastic two-player games. A stochastic game without probabilistic vertices (that is, with $V_\Diamond = \varnothing$) is called a *non-stochastic two-player game* (or simply, non-stochastic game in the sequel). In a non-stochastic game $\mathcal{G}$ with objective φ, a strategy σ_i is *winning* for Player i ($i \in \{1,2\}$) if every play in $\mathcal{G}$ consistent with σ_i satisfies the objective φ. A vertex $v \in V$ is *winning* for Player i in $\mathcal{G}$ if Player i has a winning strategy in $\mathcal{G}$ when the initial vertex is v. The set of vertices in V that are winning for Player i in $\mathcal{G}$ is the *winning region* of Player i in $\mathcal{G}$, denoted $\langle\langle i \rangle\rangle_{\mathcal{G}}(\varphi)$. If a vertex v belongs to the winning region of Player i ($i \in \{1,2\}$), then Player i is said to play *optimally* from v if they follow a winning strategy. By fixing a strategy σ_i of Player i in a non-stochastic game $\mathcal{G}$ we obtain a (possibly infinite) one-player game $\mathcal{G}^{\sigma_i}$ with vertices $V \times Q$, where Q is the set of states of a Mealy machine defining σ_i.

Subgames. Given a stochastic game $\mathcal{G} = ((V,E),(V_1,V_2,V_\Diamond),\mathbb{P},w)$, a subset $V' \subseteq V$ of vertices *induces* a subgame if (i) every vertex $v' \in V'$ has an outgoing edge in V', that is $E(v') \cap V' \neq \varnothing$, and (ii) every probabilistic vertex $v' \in V_\Diamond \cap V'$ has all outgoing edges in V', that is $E(v') \subseteq V'$. The induced *subgame* is $((V',E'),(V_1 \cap V', V_2 \cap V', V_\Diamond \cap V'),\mathbb{P}',w')$, where $E' = E \cap (V' \times V')$, and $\mathbb{P}'$ and w' are restrictions of $\mathbb{P}$ and w respectively to (V',E'). We denote this subgame by $\mathcal{G} \restriction V'$. Let φ be an objective in the stochastic game $\mathcal{G}$. We define the restriction of φ to a subgame $\mathcal{G}'$ of $\mathcal{G}$ to be the set of all plays in $\mathcal{G}'$ satisfying φ, that is, the set $\mathsf{Plays}_{\mathcal{G}'} \cap \varphi$.

Satisfaction probability. A strategy σ_1 of Player 1 is *winning* with probability p from an initial vertex v in $\mathcal{G}$ for objective φ if $\mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\varphi) \geq p$ for all strategies σ_2 of Player 2. A strategy σ_1 of Player 1 is *positive* winning (resp., *almost-sure* winning) from v for Player 1 in $\mathcal{G}$ with objective φ if $\mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\varphi) > 0$ (resp., $\mathsf{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G},v}(\varphi) = 1$) for all strategies σ_2 of Player 2. We refer to positive and almost-sure winning as *qualitative* satisfaction of φ, while for arbitrary $p \in [0,1]$, we call it *quantitative* satisfaction. We denote by $\langle\langle 1\rangle\rangle^{\mathsf{Pos}}_{\mathcal{G}}(\varphi)$ (resp., by $\langle\langle 1\rangle\rangle^{\mathsf{AS}}_{\mathcal{G}}(\varphi)$) the positive (resp., almost-sure) winning region of Player 1, i.e., the set of all vertices in $\mathcal{G}$ from which Player 1 has a positive (resp., almost-sure) winning strategy for $\mathcal{G}$ with objective φ. If a vertex v belongs to the positive (resp., almost-sure) winning region of Player 1, then Player 1 is said to play *optimally* from v if she follows a positive (resp., almost-sure) winning strategy from v. We omit analogous definitions for Player 2.

Positive attractors and traps. The Player i *positive attractor* ($i \in \{1,2\}$) to $T \subseteq V$, denoted $\mathsf{PosAttr}_i(T)$, is the set $\langle\langle i\rangle\rangle^{\mathsf{Pos}}_{\mathcal{G}}(\mathsf{Reach}(T))$ of vertices in V from which Player i can ensure that the token reaches a vertex in T with positive probability. It can be computed as the least fixed point of the operator $\lambda x.\mathsf{PosPre}_i(x) \cup T$ where $\mathsf{PosPre}_1(x) = \{v \in V_1 \cup V_\Diamond \mid E(v) \cap x \neq \varnothing\} \cup \{v \in V_2 \mid E(v) \subseteq x\}$ is the positive predecessor operator for Player 1, and PosPre_2 is defined analogously for Player 2. Intuitively $\mathsf{PosPre}_i(x)$ is the set of vertices from which Player i has a strategy to ensure with positive probability that the vertex in the next round is in x. It is possible to compute the positive attractor in $\mathcal{O}(|E|)$ time [CH08]. It is easy to derive from the computation of $\mathsf{PosAttr}_i(T)$ a memoryless strategy for Player i that ensures the positive satisfaction of $\mathsf{Reach}(T)$ from vertices in $\mathsf{PosAttr}_i(T)$. We call such a strategy a *positive-attractor strategy* of Player i. Given a set T, we denote the standard notion of an attractor to T from the literature by $\mathsf{Attr}_i(T)$. In non-stochastic games, a positive-attractor to the set T is the same as a standard attractor to T.

A *trap* for Player 1 is a set $T \subseteq V$ such that for every vertex $v \in T$, if $v \in V_1 \cup V_\Diamond$, then $E(v) \subseteq T$, and if $v \in V_2$, then $E(v) \cap T \neq \varnothing$, that is $\mathsf{PosPre}_1(V \setminus T) = \varnothing$. In other words, from every vertex $v \in T$, Player 2 can ensure (with probability 1) that the game never leaves T, moreover using a memoryless strategy. A trap for Player 2 can be defined analogously.

Remark 2.1. Let $\mathcal{G}$ be a non-stochastic game with objective φ for Player 1. If φ is closed under suffixes, then the winning region of Player 1 is a trap for Player 2. As a corollary, if φ is prefix-independent, then the winning region of Player 1 is a trap for Player 2 and the winning region of Player 2 is a trap for Player 1.

3. Window mean payoff

We consider two types of window mean-payoff objectives, introduced in [CDRR15]: (*i*) *fixed window mean-payoff* objective ($\mathsf{FWMP}(\ell)$) in which a window length $\ell \geq 1$ is given, and (*ii*) *bounded window mean-payoff* objective (BWMP) in which for every play, we need a bound on window lengths. We define these objectives below.

For a play π in a stochastic game $\mathcal{G}$, the *total payoff* of an infix $\pi(i, i+n) = v_i v_{i+1} \cdots v_{i+n}$ is the sum of the payoffs of the edges in the infix and is defined as $\mathsf{TP}(\pi(i, i+n)) = \sum_{k=i}^{i+n-1} w(v_k, v_{k+1})$. The *mean payoff* of an infix $\pi(i, i+n)$ is the average of the payoffs of the

edges in the infix and is defined as $\mathsf{MP}(\pi(i,i+n)) = \frac{1}{n}\mathsf{TP}(\pi(i,i+n)) = \sum_{k=i}^{i+n-1} \frac{1}{n} w(v_k, v_{k+1})$. The mean payoff of a play π is defined as $\mathsf{MP}(\pi) = \liminf_{n\to\infty} \mathsf{MP}(\pi(0,n))$.

Given a window length $\ell \geq 1$, a play $\pi = v_0 v_1 \cdots$ in $\mathcal{G}$ satisfies the *fixed window mean-payoff objective* $\mathsf{FWMP}_{\mathcal{G}}(\ell)$ if from every position after some point, it is possible to start an infix of length at most ℓ with a nonnegative mean payoff. Formally,

$$\mathsf{FWMP}_{\mathcal{G}}(\ell) = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid \exists k \geq 0 \cdot \forall i \geq k \cdot \exists j \in \{1,\ldots,\ell\} : \mathsf{MP}(\pi(i,i+j)) \geq 0\}.$$

We omit the subscript $\mathcal{G}$ when it is clear from the context. In this definition, there is no loss of generality in considering mean-payoff threshold 0 rather than some $\lambda \in \mathbb{Q}$: consider the game $\mathcal{G}'$ obtained by subtracting λ from every edge payoff in $\mathcal{G}$, and the mean payoff of any infix of a play in $\mathcal{G}$ is at least λ if and only if its mean payoff in $\mathcal{G}'$ is nonnegative. Moreover, observe that the mean payoff of an infix is nonnegative if and only if the total-payoff of the infix is nonnegative.

Note that when $\ell = 1$, the $\mathsf{FWMP}(1)$ and $\overline{\mathsf{FWMP}(1)}$ (i.e., the complement of $\mathsf{FWMP}(1)$) objectives reduce to coBüchi and Büchi objectives respectively. To see this, let T be the set of all vertices $v \in V$ such that either $v \in V_1$ and all out-edges of v have a negative payoff, or $v \in V_2$ and at least one out-edge of v has a negative payoff. Then, a play satisfies the $\overline{\mathsf{FWMP}(1)}$ objective if and only if it satisfies the $\mathsf{Büchi}(T)$ objective. The following properties of $\mathsf{FWMP}(\ell)$ have been observed in [CDRR15]. The fixed window mean-payoff objective provides a robust and conservative approximation of the traditional mean-payoff objective, defined as the set of plays with nonnegative mean payoff: for all window lengths ℓ, if a play π satisfies $\mathsf{FWMP}_{\mathcal{G}}(\ell)$, then it has a nonnegative mean payoff. Since $\ell \leq \ell'$ implies $\mathsf{FWMP}_{\mathcal{G}}(\ell) \subseteq \mathsf{FWMP}_{\mathcal{G}}(\ell')$, more precise approximations of mean payoff can be obtained by increasing the window length. In all plays satisfying $\mathsf{FWMP}(\ell)$, there exists a suffix that can be decomposed into infixes of length at most ℓ, each with a nonnegative mean payoff. Such a desirable robust property is not guaranteed by the classical mean-payoff objective, where infixes of unbounded lengths may have negative mean payoff.

As defined in [CDRR15], given a play $\pi = v_0 v_1 \cdots$ and $0 \leq i < j$, we say that the window $\pi(i,j)$ is *open* if the total-payoff of $\pi(i,k)$ is negative for all $i < k \leq j$. Otherwise, the window is *closed*. Given $j > 0$, we say a window is open at j if there exists an open window $\pi(i,j)$ for some $i < j$. The window starting at position i *closes* at position j if j is the first position after i such that the total payoff of $\pi(i,j)$ is nonnegative. If the window starting at i closes at j, then for all $i \leq k < j$, the windows $\pi(k,j)$ are closed. This property is called the *inductive property of windows*. A play π satisfies $\mathsf{FWMP}(\ell)$ if and only if, from some point on, every window in π closes within at most ℓ steps.

We also consider the bounded window mean payoff objective $\mathsf{BWMP}_{\mathcal{G}}$. We omit the subscript $\mathcal{G}$ when it is clear from the context. A play π satisfies the BWMP objective if there exists a window length $\ell \geq 1$ for which π satisfies $\mathsf{FWMP}(\ell)$. Formally,

$$\mathsf{BWMP}_{\mathcal{G}} = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid \exists \ell \geq 1 : \pi \in \mathsf{FWMP}(\ell)\}.$$

Equivalently, a play π does not satisfy BWMP if and only if for every suffix of π, for all $\ell \geq 1$, the suffix contains an open window of length ℓ. Note that both $\mathsf{FWMP}(\ell)$ for all $\ell \geq 1$ and BWMP are prefix-independent objectives.

Decision problems. Given a game $\mathcal{G}$, an initial vertex $v \in V$, a rational threshold $p \in [0,1]$, and an objective φ (that is either $\mathsf{FWMP}(\ell)$ for a given window length $\ell \geq 1$, or BWMP), consider the problem of deciding:

Algorithm 1 $\mathsf{NonStocFWMP}(\mathcal{G}, \ell)$ [CDRR15, Algorithm 1]

Input: $\mathcal{G} = ((V, E), (V_1, V_2, \varnothing), w)$, the non-stochastic game, and $\ell \geq 1$, the window length
Output: The set of vertices in V from which Player 1 wins $\mathsf{FWMP}(\ell)$

```
W_d ← NonStocDirFWMP(G, ℓ)
if W_d = ∅ then
    return ∅
else
    A ← Attr_1(W_d)
    return A ∪ NonStocFWMP(G ↾ (V \ A), ℓ)
```

- *Positive satisfaction of* φ: if Player 1 positively wins φ from v, i.e., if $v \in \langle\langle 1 \rangle\rangle_{\mathcal{G}}^{\mathsf{Pos}}(\varphi)$.
- *Almost-sure satisfaction of* φ: if Player 1 almost-surely wins φ from v, i.e., if $v \in \langle\langle 1 \rangle\rangle_{\mathcal{G}}^{\mathsf{AS}}(\varphi)$.
- *Quantitative satisfaction of* φ (also known as *quantitative value problem* [CHH09b]): if Player 1 wins φ from v with probability at least p, i.e., if $\sup_{\sigma_1 \in \Lambda_1} \inf_{\sigma_2 \in \Lambda_2} \mathsf{Pr}_{\mathcal{G},v}^{\sigma_1,\sigma_2}(\varphi) \geq p$.

Note that these three problems coincide for non-stochastic games. As considered in previous works [CDRR15, BGR19, BDOR20], the window length ℓ is usually small (typically $\ell \leq |V|$), and therefore we assume that ℓ is given in unary (while the payoff on the edges is given in binary).

Determinacy. From determinacy of Blackwell games [Mar98], stochastic games with window mean-payoff objectives as defined above are determined, i.e., the largest probability with which Player 1 is winning and the largest probability with which Player 2 is winning add up to 1.

Algorithms for non-stochastic window mean-payoff games. To compute the positive and almost-sure winning regions for Player 1 for $\mathsf{FWMP}(\ell)$, we recall intermediate objectives defined in [CDRR15]. The *good window* objective $\mathsf{GW}_{\mathcal{G}}(\ell)$ consists of all plays π in $\mathcal{G}$ such that the window opened at the first position in the play closes in at most ℓ steps:

$$\mathsf{GW}_{\mathcal{G}}(\ell) = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid \exists j \in \{1, \ldots, \ell\} : \mathsf{MP}(\pi(0, j)) \geq 0\}.$$

The *direct fixed window mean-payoff* objective $\mathsf{DirFWMP}_{\mathcal{G}}(\ell)$ consists of all plays π in $\mathcal{G}$ such that from every position in π, the window closes in at most ℓ steps:

$$\mathsf{DirFWMP}_{\mathcal{G}}(\ell) = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid \forall i \geq 0 : \pi(i, \infty) \in \mathsf{GW}_{\mathcal{G}}(\ell)\}.$$

The $\mathsf{FWMP}_{\mathcal{G}}(\ell)$ objective can be expressed in terms of $\mathsf{DirFWMP}_{\mathcal{G}}(\ell)$:

$$\mathsf{FWMP}_{\mathcal{G}}(\ell) = \{\pi \in \mathsf{Plays}_{\mathcal{G}} \mid \exists k \geq 0 : \pi(k, \infty) \in \mathsf{DirFWMP}_{\mathcal{G}}(\ell)\}.$$

We refer to Algorithm 1, 2, and 3 from [CDRR15] shown below with the same numbering. They compute the winning regions for Player 1 for the $\mathsf{FWMP}(\ell)$, $\mathsf{DirFWMP}(\ell)$, and $\mathsf{GW}(\ell)$ objectives in non-stochastic games respectively. [CDRR15, Algorithm 2 and Algorithm 3] contain subtle errors for which the fixes are known [BHR16b, Hau18]. In fact, a related objective that is a combination of the good window and reachability objectives was studied in [Hau18] and [BHR16b] from which the correct algorithms can be derived. For completeness, we include below counterexamples for the versions in [CDRR15], along with the correct algorithms and brief explanations of correctness.

Algorithm 2 NonStocDirFWMP$(\mathcal{G}, \ell)$

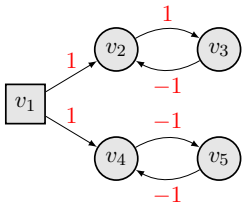
Input: $\mathcal{G} = ((V, E), (V_1, V_2, \varnothing), w)$ the non-stochastic game, and $\ell \geq 1$, the window length
Output: The set of vertices in V from which Player 1 wins DirFWMP(ℓ)

1: $W_{gw} \leftarrow \mathsf{GoodWin}(\mathcal{G}, \ell)$
2: **if** $W_{gw} = V$ or $W_{gw} = \varnothing$ **then**
3: **return** W_{gw}
4: **else**
5: $A \leftarrow \mathsf{Attr}_2(V \setminus W_{gw})$
6: **return** NonStocDirFWMP$(\mathcal{G} \restriction (W_{gw} \setminus A), \ell)$

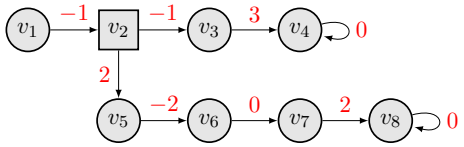
Algorithm 3 GoodWin$(\mathcal{G}, \ell)$

Input: $\mathcal{G} = ((V, E), (V_1, V_2, \varnothing), w)$ the non-stochastic game, and $\ell \geq 1$, the window length
Output: The set of vertices in V from which Player 1 wins GW(ℓ)

1: **for all** $v \in V$ **do**
2: $C_0(v) \leftarrow 0$
3: **for all** $i \in \{1, \ldots, \ell\}$ **do**
4: $C_i(v) \leftarrow -\infty$
5: **for all** $i \in \{1, \ldots, \ell\}$ **do**
6: **for all** $v \in V_1$ **do**
7: $C_i(v) \leftarrow \max_{(v,v') \in E}\{\max\{w(v, v'), w(v, v') + C_{i-1}(v')\}\}$ ▷ *In* [CDRR15], $w(v, v') + C_{i-1}(v')$ *was used instead of* $\max\{w(v, v'), w(v, v') + C_{i-1}(v')\}$.
8: **for all** $v \in V_2$ **do**
9: $C_i(v) \leftarrow \min_{(v,v') \in E}\{\max\{w(v, v'), w(v, v') + C_{i-1}(v')\}\}$ ▷ *In* [CDRR15], $w(v, v') + C_{i-1}(v')$ *was used instead of* $\max\{w(v, v'), w(v, v') + C_{i-1}(v')\}$.
10: $W_{gw} \leftarrow \{v \in V \mid C_\ell(v) \geq 0\}$ ▷ *In* [CDRR15], W_{gw} *was defined as* $\{v \in V \mid \exists i \in \{1, 2, \ldots \ell\},\ C_i(v) \geq 0\}$ *instead because* $C_i(v)$ *had a different definition in* [CDRR15].
11: **return** W_{gw}



(A) A counterexample for computing winning region for DirFWMP(ℓ) [CDRR15, Algorithm 2] with $\ell = 2$.



(B) A counterexample for computing winning region for GW(ℓ) [CDRR15, Algorithm 3] with $\ell = 3$.

FIGURE 2. Counterexamples for algorithms in [CDRR15]

Description of algorithms from [CDRR15]**.** Algorithm 1 [CDRR15, Algorithm 1] computes the winning region of Player 1 for the FWMP(ℓ) objective. First (Line 1) it computes the winning region W_d for Player 1 for the DirFWMP(ℓ) objective (using Algorithm 2). If W_d is empty, then it is easy to show that the winning region for Player 1 for objective FWMP(ℓ)

is also empty, and the algorithm terminates (Line 3). Otherwise, all vertices in the Player 1 attractor of W_d (Line 5) are also winning (as $\mathsf{FWMP}(\ell)$ is closed under prefixes), and the remaining states (i.e., the complement of the attractor) induce a smaller subgame, which can be solved (recursively) by the same algorithm.

Algorithm 2 computes the winning region of Player 1 for the $\mathsf{DirFWMP}(\ell)$ objective. It does so by first computing the region $V \setminus W_{gw}$ from which Player 1 cannot win the good window objective $\mathsf{GW}(\ell)$ (Line 3). If Player 1 does not win the $\mathsf{GW}(\ell)$ objective, then she does not win the $\mathsf{DirFWMP}(\ell)$ objective either, and thus, all vertices in $V \setminus W_{gw}$ are losing for Player 1. If $V \setminus W_{gw}$ is empty, that is, if $W_{gw} = V$, then Player 1 wins the $\mathsf{GW}(\ell)$ objective from every vertex, and it is easy to see that Player 1 also wins the $\mathsf{DirFWMP}(\ell)$ objective from every vertex. Otherwise, the Player 2 attractor to $V \setminus W_{gw}$ is also losing for Player 1. The remaining states (i.e., the complement of A) induce a smaller subgame, which can be solved (recursively) by the same algorithm.

Algorithm 3 computes the winning region of Player 1 for the good window objective $\mathsf{GW}(\ell)$, that is, the set of vertices from which Player 1 can close the window within at most ℓ steps. The algorithm uses dynamic programming to compute, for all $v \in V$ and all lengths $i \in \{1, \ldots, \ell\}$, the largest payoff $C_i(v)$ that Player 1 can ensure from v within at most i steps. The winning region for $\mathsf{GW}(\ell)$ for Player 1 consists of all vertices v such that $C_\ell(v) \geq 0$.

Correctness of Algorithm 2. We show the correctness of Algorithm 2, that is, we show that this algorithm correctly computes $\langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{DirFWMP}(\ell))$, the winning region for Player 1 for the $\mathsf{DirFWMP}(\ell)$ objective. The proof makes use of the fact that $\mathsf{DirFWMP}(\ell) \subseteq \mathsf{GW}(\ell)$, that is, if Player 1 does not win $\mathsf{GW}(\ell)$ from a vertex $v \in V$, then she also does not win $\mathsf{DirFWMP}(\ell)$ from v.

The algorithm successively finds vertices that are losing for Player 1 for the $\mathsf{GW}(\ell)$ objective, removes them, and recurses on the rest of the game graph. In Line 1, we have $W_{gw} = \langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{GW}(\ell))$, the winning region for Player 1 for the $\mathsf{GW}(\ell)$ objective.

- If $W_{gw} = \varnothing$, then Player 2 wins $\overline{\mathsf{GW}(\ell)}$ from all vertices in V, and therefore, Player 2 also wins $\overline{\mathsf{DirFWMP}(\ell)}$ from all vertices in V. Hence, $\langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{DirFWMP}(\ell)) = \varnothing$.
- Otherwise, if $W_{gw} = V$, then Player 1 wins $\mathsf{GW}(\ell)$ from all vertices in G. For all vertices $v \in V$, starting from v, Player 1 can ensure that the window starting at v closes in at most ℓ steps. When the window starting at v closes, suppose the token is on a vertex v'. By the inductive property of windows, all windows that opened after v are also closed by the time the token reaches v'. Now, since $v' \in \langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{GW}(\ell))$, Player 1 can ensure that the window starting at v' also closes in at most ℓ steps. In this manner, Player 1 closes every window in at most ℓ steps, resulting in an outcome that is winning for the $\mathsf{DirFWMP}(\ell)$ objective. We get that $\langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{DirFWMP}(\ell)) = V$.
- Finally, suppose $\varnothing \subsetneq W_{gw} \subsetneq V$. Starting from $V \setminus W_{gw}$, Player 2 wins the $\overline{\mathsf{GW}(\ell)}$ objective, and hence, also the $\overline{\mathsf{DirFWMP}(\ell)}$ objective. Therefore, no vertex in $V \setminus W_{gw}$ belongs to $\langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{DirFWMP}(\ell))$. Moreover, consider the Player 2 attractor A to $V \setminus W_{gw}$ (Line 5). Starting from a vertex in this attractor, Player 2 can follow a memoryless strategy to eventually reach $V \setminus W_{gw}$. Once the token reaches $V \setminus W_{gw}$, Player 2 can ensure that a window remains open for ℓ steps, resulting in Player 1 losing. Hence, no vertex in the attractor belongs to $\langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{DirFWMP}(\ell))$ either. The complement of this Player 2 attractor is a trap for Player 2 and induces a subgame. For all vertices v in this subgame, if Player 1 wins from v in the subgame, then she also wins from v in the original game as she can

mimic a winning strategy from the subgame while also ensuring that the token never leaves the subgame. Conversely, for all vertices v in the subgame, if Player 1 does not win from v in the subgame, then she also does not win from v in the original game. This is because if the token remains in the subgame forever, then Player 2 wins, and if the token ever leaves the subgame, then as discussed above, the token enters the Player 2 attractor, and Player 2 wins. Thus, for all vertices v in the subgame, Player 1 wins from v in the subgame if and only if she wins from v in the original game. Hence, the winning region for Player 1 in the subgame is equal to the winning region for Player 1 in the original game, and hence, the algorithm recurses on this subgame.

Correctness of Algorithm 3. The following characterization of $C_i(v)$ holds:

- There exists a strategy σ_1 of Player 1 such that for all strategies σ_2 of Player 2, there exists $1 \leq j \leq i$ such that in the outcome of the strategy profile (σ_1, σ_2) with initial vertex v, the total payoff in the first j steps of the outcome is at least $C_i(v)$;
- For all strategies σ_1 of Player 1, there exists a strategy σ_2 of Player 2 such that for all $1 \leq j \leq i$, the total payoff of the first j steps in the outcome of (σ_1, σ_2) with initial vertex v is at most $C_i(v)$.

A monotonicity property follows from the above characterization, namely that for all $v \in V$, if $1 \leq i \leq j \leq \ell$, then $C_i(v) \leq C_j(v)$, which can also easily be established from Algorithm 3. Note that monotonicity does not hold for $i = 0$ as $C_0(v) = 0$ for all $v \in V$, but we may have $C_1(v) < 0$ (e.g., if all outgoing edges from v have negative weight). It also follows from this characterization that $C_i(v) \geq 0$ if Player 1 wins from v for objective $\mathsf{GW}(i)$.

We now show the correctness of Algorithm 3 to compute $C_i(\cdot)$, by induction on $i \in \{1, \ldots, \ell\}$. The base case $i = 1$ holds since $C_0(v) = 0$ for all $v \in V$ and the maximum possible payoff from v in one step is $C_1(v) = \max_{(v,v') \in E}\{w(v, v')\}$ if $v \in V_1$ is a vertex of Player 1, and $C_1(v) = \min_{(v,v') \in E}\{w(v, v')\}$ if $v \in V_2$ is a vertex of Player 2. For the induction step $i \geq 2$, assume that $C_{i-1}(v)$ is correctly computed by the algorithm for all $v \in V$, as the maximum payoff that Player 1 can ensure from v in at least 1 and at most $i - 1$ steps. Then, from a vertex v and if the edge (v, v') is chosen (either by Player 1 or by Player 2), the maximum payoff that Player 1 can ensure in at least 1 and at most i steps is either $w(v, v')$ (in 1 step) or $w(v, v') + C_{i-1}(v')$ (in at least $1 + 1 = 2$ steps and at most $1 + i - 1 = i$ steps), whichever is greater. Hence if $v \in V_1$ is a vertex of Player 1, then $C_i(v)$ is the maximum such value across the out-neighbours v' of v, and if $v \in V_2$ is a vertex of Player 2, then $C_i(v)$ is the minimum, as in Line 7 and Line 9 of the algorithm. Finally by the characterization of $C_i(v)$, Player 1 wins from v for the $\mathsf{GW}(\ell)$ objective if $C_\ell(v) \geq 0$ (Line 10), which by the monotonicity property, is equivalent to the condition $\exists 1 \leq i \leq \ell : C_i(v) \geq 0$ used in [CDRR15].

Counterexample for [CDRR15, Algorithm 2]**.** The version of Algorithm 2 in [CDRR15] does not compute (and does not remove) the Player 2 attractor $\mathsf{Attr}_2(V \setminus W_{gw})$ to the winning region of Player 2 for the good-window objective (Line 5). However, it is easy to see that Player 2 can spoil the good-window objective from A, not only from $V \setminus W_{gw}$. This may lead to incorrectly classifying some losing states as being winning (for Player 1). Consider the non-stochastic game shown in Figure 2a with $\ell = 2$. The vertices v_4 and v_5 are losing for Player 1 for $\mathsf{DirFWMP}(\ell)$, and since $v_1 \in V_2$, the vertex v_1 is also losing for Player 1. The

remaining vertices $\{v_2, v_3\}$ are winning for Player 1. After computing the winning region of Player 2 for the good-window objective, which is $\{v_4, v_5\}$, the winning region in the subgame induced by $V \setminus \{v_4, v_5\} = \{v_1, v_2, v_3\}$ is $\{v_1, v_2, v_3\}$, which is returned as the winning region for $\mathsf{DirFWMP}(\ell)$, instead of $\{v_2, v_3\}$.

Counterexample for [CDRR15, Algorithm 3]**.** The version of Algorithm 3 in [CDRR15] differs at Line 7 and Line 9, and we show that it does not compute the winning region for the good-window objective.

Consider the non-stochastic game $\mathcal{G}$ shown in Figure 2b with $\ell = 3$. The vertex v_1 is winning for $\mathsf{GW}(3)$ since the window closes in three steps irrespective of the successor chosen by Player 2 from v_2. If Player 2 chooses v_3 from v_2, the window closes in three steps, whereas if Player 2 chooses v_5 from v_2, the window closes in two steps. However, Algorithm 3 in [CDRR15] does not include v_1 in the winning region for $\mathsf{GW}(3)$. This is because for every vertex v in the game, it computes for all $1 \leq i \leq \ell$, the value $C_i(v)$ as the best payoff that Player 1 can ensure from v in *exactly* i steps, instead of at most i steps. The algorithm includes a vertex v in the winning region for $\mathsf{GW}(\ell)$ if at least one of the $C_i(v)$ is nonnegative. In our example, the best payoff that Player 1 can ensure from v_1 in exactly one step is -1, in exactly two steps is -2 (corresponding to the prefix $v_1v_2v_3$), and in exactly three steps is -1 (corresponding to the prefix $v_1v_2v_5v_6$). Thus, for all $1 \leq i \leq 3$, the value $C_i(v_1)$ is negative. This example shows that it is possible for Player 1 to ensure a nonnegative payoff in at most ℓ steps despite the worst payoff in exactly i steps being negative for all $i \in \{1, \ldots, \ell\}$.

4. Memory requirement for non-stochastic window mean-payoff games

The memory requirement for winning strategies of both Player 1 and Player 2 in non-stochastic games with objective $\mathsf{FWMP}(\ell)$ is claimed to be $\mathcal{O}(|V| \cdot \ell)$ without proof [CDRR15, Lemma 7]. Further, the bounds are "correctly stated" as $\mathcal{O}(w_{\max} \cdot \ell^2)$ and $\mathcal{O}(w_{\max} \cdot \ell^2 \cdot |V|)$ for Player 1 and Player 2 respectively, where $w_{\max}$ is the maximum absolute payoff in the graph [BHR16b, Theorem 2]. We improve upon these bounds and show that memory of size ℓ suffices for a winning strategy of Player 1. Furthermore, a formal argument for memory requirement for Player 2 strategies is missing in [CDRR15] which we provide here.

We show constructions of Mealy machines M_1^{NS} and M_2^{NS} (with at most ℓ and $|V| \cdot \ell$ states respectively) that define winning strategies σ_1^{NS} and σ_2^{NS} of Player 1 and Player 2 respectively, showing upper bounds on the memory requirements for both players. We also present a family of games with arbitrarily large state space where Player 2 is winning and all his winning strategies require at least $\frac{1}{2}(|V| - \ell) + 3$ memory, while it was only known that memoryless strategies are not sufficient for Player 2 [CDRR15].

The paper [CDRR15] also has results on the analysis of the BWMP objective. It has been shown that solving BWMP for non-stochastic games is in $\mathsf{NP} \cap \mathsf{coNP}$, and memoryless strategies suffice for Player 1, whereas Player 2 may need infinite memory strategies to play optimally.

4.1. **Memory requirement for Player 1 for FWMP objective.**

Upper bound on memory requirement for Player 1. We show that memory of size ℓ suffices for winning strategies of Player 1 for the $\mathsf{DirFWMP}(\ell)$ objective (Lemma 4.1), which is in turn used to show that the same memory also works for the $\mathsf{FWMP}(\ell)$ objective (Theorem 4.4).

Lemma 4.1. *If Player* 1 *wins in a non-stochastic game with objective* $\mathsf{DirFWMP}(\ell)$*, then Player* 1 *has a winning strategy with memory of size* ℓ*.*

Proof. Given a game $\mathcal{G}$, let W_d be the winning region of Player 1 in $\mathcal{G}$ for objective $\mathsf{DirFWMP}(\ell)$. Note that the region W_d is a trap for Player 2 in $\mathcal{G}$, as the objective $\mathsf{DirFWMP}(\ell)$ is closed under suffixes. Every vertex in W_d is moreover winning for Player 1 with objective $\mathsf{GW}(\ell)$, by definition.

A winning strategy of Player 1 is to play for the objective $\mathsf{GW}(\ell)$ until the window closes (which Player 1 can ensure within at most ℓ steps), and then to restart with the same strategy, playing for $\mathsf{GW}(\ell)$ and so on. Using memory space $Q = \{1, \ldots, \ell\}$, we may store the number of steps remaining before the window must close to satisfy $\mathsf{GW}(\ell)$, and reset the memory to $q_0 = \ell$ whenever the window closes. However, the window may close any time within ℓ steps, and the difficulty is to detect when this happens: how to update the memory $q = i$, given the next visited vertex v, but independently of the history? Intuitively, the memory should be updated to $q = i - 1$ if the window did not close yet upon reaching v, and to $q = \ell$ if it did, but that depends on which path was followed to reach v (not just on v), which is not stored in the memory space.

The crux is to show that it is not always necessary for Player 1 to be able to infer when the window closes. Given the current memory state $q = i$, and the next visited vertex v, the memory update is as follows: if $C_i(v) \geq 0$ (that is, Player 1 can ensure the window from v will close within i steps), then we update to $q = i - 1$ (decrement) although the window may or may not have closed upon reaching v; otherwise $C_i(v) < 0$ and we update to $q = \ell - 1$ (reset to ℓ and decrement) and we show that in this case the window did close. Intuitively, updating to $q = i - 1$ is safe even if the window did close, because the strategy of Player 1 will anyway ensure the (upcoming) window is closed within $i - 1 < \ell$ steps. For the case $C_i(v) < 0$, we want the Mealy machine to be in state ℓ when v is being read. However, there is an additional difficulty to this. Assume that vertex v' is read by the Mealy machine before reading v. The Mealy machine is thus in state $i + 1$ and $C_{i+1}(v') \geq 0$. Now $C_i(v) < 0$ denotes that an open window is closed on the edge (v', v), and the state of the Mealy machine should be reset to ℓ. However, if v' is a Player 2 vertex, since the vertex chosen by a Player 2 strategy from v' is not known to Player 1 (the output on the transition of the Mealy machine is thus ϵ), the state of the Mealy machine is updated from $i + 1$ to i while reading v'. The state is then updated to $\ell - 1$ after reading v to simulate that the window was already closed upon reaching v.

For $v \in V_1$, we define the next vertex chosen by the strategy as

$$D_i(v) = \arg \max_{(v,v') \in E} \{\max\{w(v, v'), w(v, v') + C_{i-1}(v')\}\},$$

the out-neighbour from v that maximizes the expression of Algorithm 3 for the $\mathsf{GW}(\ell)$ objective, Line 7. If there is more than one such out-neighbour, we choose one arbitrarily. Example 4.2 illustrates how computing $C_i(v)$ and checking if it is nonnegative is useful in constructing a winning strategy for Player 1. We see in Construction 4.3 a formal description

of a Mealy machine with ℓ states defining a winning strategy of Player 1 for the $\mathsf{DirFWMP}(\ell)$ objective. This concludes the proof of Lemma 4.1. □

Example 4.2. In this example, we show why checking if $C_i(v)$ is negative or not is sometimes necessary. Figure 3 shows a fragment of a game where if Player 1 does not know if the window has closed, then she may choose a vertex that causes her to lose the $\mathsf{DirFWMP}(\ell)$ objective for $\ell = 4$. Suppose it is the case that all windows in the play have closed when the

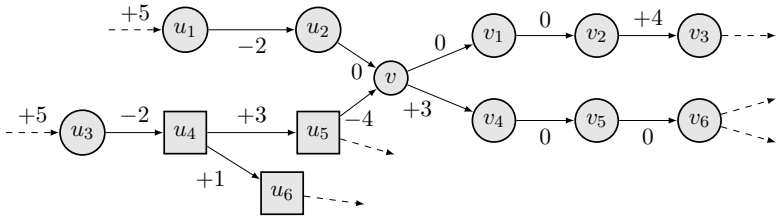


FIGURE 3. The successor of v that Player 1 should choose depends on how many more steps she has to close the window. If Player 1 does not detect that the window is closed on (u_4, u_5), then she chooses v_4 from v. Otherwise, she chooses v_1 from v. Computing $C_2(u_5)$ shows that it is negative and this implies that the window starting at u_3 must have closed at u_5.

token reaches u_1 and u_3. If the token reaches v along u_1u_2, then Player 1 must move the token from v to $D_2(v) = v_4$ as this closes the window starting at u_1. If Player 1 moves the token from v to v_1 instead, then this results in an open window $u_1u_2vv_1v_2$ of length 4 which is not desirable for Player 1.

On the other hand, if the token reaches v along $u_3u_4u_5$, then since the window starting at u_3 closes at u_5, we have that Player 1 must choose a successor of v such that the window starting at u_5 closes in at most three steps from v. Hence, if Player 1 moves the token from v to $D_3(v)$, that is, v_1, then the window starting at u_5 closes in at most 4 steps. However, suppose Player 1 does not detect that the window starting at u_3 closes at u_5. Although the total payoff along $u_3u_4u_5vv_4$ is nonnegative (which implies that the window starting at u_3 is closed at v_4), one cannot use the inductive property of windows to claim that all subsequent windows are closed at v_4. The inductive property of windows does not hold since the window starting at u_3 closes at u_5 and this gives no information about when the window starting at u_5 closes. If Player 1 plays from v as if she has only one more step to close the window, then she moves the token to $D_1(v) = v_4$ and this results in an open window $u_5vv_4v_5v_6$ of length 4 which is undesirable for Player 1.

Thus, if Player 1 never detects window closings, then this may result in open windows of length ℓ in the outcome. Since $C_2(u_5)$ is negative and u_5 belongs to the winning region for $\mathsf{DirFWMP}(\ell)$, this implies that the window starting at u_3 must have closed at u_5 and Player 1 cannot continue playing as if the window did not close at u_5. Computing $C_i(v)$ in general helps detect those window closings where Player 1 cannot continue on as if the window did not close. If $C_i(v) \geq 0$, then even if a window closes along a path when v is reached, it is safe not to detect it, and we can still construct a winning strategy if one exists. □

Construction 4.3. We construct a Mealy machine $M_{\mathsf{d}} = (Q_{\mathsf{d}}, q_0, V, V \cup \{\epsilon\}, \Delta_{\mathsf{d}}, \delta_{\mathsf{d}})$ with ℓ states that defines a winning strategy σ_{d} of Player 1 for the $\mathsf{DirFWMP}(\ell)$ objective, where:

- the memory $Q_{\mathsf{d}} = \{1, \dots, \ell\}$ stores a counter (modulo ℓ), and we assume arithmetic modulo ℓ (that is, $\ell + 1 = 1$, $1 - 1 = \ell$, etc.);
- the initial state is $q_0 = \ell$ (although we show that an arbitrary initial state also induces a winning strategy);
- the input alphabet is V, as the Mealy machine reads vertices of the game,
- the output alphabet is $V \cup \{\epsilon\}$, as the Mealy machine either outputs a vertex (upon reading a vertex of Player 1) or ϵ (upon reading a vertex of Player 2);
- The transition function $\Delta_{\mathsf{d}} \colon Q_{\mathsf{d}} \times V \to Q_{\mathsf{d}}$ is defined as follows:

$$\Delta_{\mathsf{d}}(i, v) = \begin{cases} i - 1 \pmod{\ell} & \text{if } C_i(v) \geq 0 \quad (\textit{decrement}) \\ \ell - 1 & \text{if } C_i(v) < 0 \quad (\textit{reset and decrement}) \end{cases}$$

- The output function $\delta_{\mathsf{d}} \colon \{1, \dots, \ell\} \times V \to V \cup \{\epsilon\}$ is defined as follows:

$$\delta_{\mathsf{d}}(i, v) = \begin{cases} \epsilon & \text{if } v \in V_2 \\ D_i(v) & \text{if } v \in V_1 \text{ and } C_i(v) \geq 0 \\ D_\ell(v) & \text{if } v \in V_1 \text{ and } C_i(v) < 0 \end{cases}$$

Note that if $C_i(v) < 0$, then $\delta_{\mathsf{d}}(i, v) = \delta_{\mathsf{d}}(\ell, v)$, that is, on a reset the strategy plays as if the counter was equal to ℓ.

We establish the correctness of the construction as follows. We show that the strategy σ_{d} of Player 1 defined by M_{d} is winning, that is for all strategies σ_2 of Player 2, the outcome π of the strategy profile $(\sigma_{\mathsf{d}}, \sigma_2)$ satisfies the objective $\mathsf{DirFWMP}(\ell)$.

We show that every window in π closes within at most ℓ steps. We split the outcome π into segments (where the last vertex of each segment is the same as the first vertex of the next segment) such that for each segment, the state of the Mealy machine is updated to $\ell - 1$ upon reading the last vertex of the segment (thus also upon reading the first vertex of each segment), but the Mealy machine is never in state $\ell - 1$ in between. Note that the initial memory state of the Mealy machine is ℓ, thus the first segment starts at the beginning of the outcome, and the segments cover the whole outcome. Note also that the length of each segment (i.e., the number of transitions) is at most ℓ since either the memory state is either updated to $\ell - 1$, or decreased by 1 (modulo ℓ).

For all segments in the outcome π, we show that all windows that open in the segment are closed by (or before) the end of the segment, from which we can conclude that the objective $\mathsf{DirFWMP}(\ell)$ is satisfied.

Consider a segment $v_\ell v_{\ell-1} v_{\ell-2} \cdots v_{p+1} v_p$ (where $p \geq 0$ since each segment has at most ℓ transitions), and the sequence of memory states along the segment:

$$q_\ell \xrightarrow{v_\ell} q_{\ell-1} \xrightarrow{v_{\ell-1}} q_{\ell-2} \ \cdots \ q_{p+1} \xrightarrow{v_{p+1}} q_p \xrightarrow{v_p} q_{p-1}$$

which can be written as:

$$x \xrightarrow{v_\ell} \ell - 1 \xrightarrow{v_{\ell-1}} \ell - 2 \ \cdots \ p + 1 \xrightarrow{v_{p+1}} y \xrightarrow{v_p} \ell - 1$$

where $q_{\ell-1} = \ell - 1$ and $q_{p-1} = \ell - 1$ by the definition of segments, and $q_i = i$ for $i = p + 1, \dots, \ell - 2$ since the counter is decremented whenever it is not reset to $\ell - 1$ (and thus we also have $C_i(v_i) \geq 0$ for $i = p + 1, \dots, \ell - 1$). We discuss the possible values of $q_p = y$ (and $q_\ell = x$ for which the situation is similar). There are two possibilities: either $y = \ell$ and the counter is decremented upon reading v_p (and thus $p = 0$ and $C_p(v_p) = 0$), or $y = p < \ell$ and the counter is reset upon reading v_p (and thus $C_p(v_p) < 0$). It follows that in both cases $C_p(v_p) \leq 0$.

Moreover at the beginning of the segment (considering $q_\ell = x$), the strategy chooses $D_\ell(v_\ell)$ upon reading v_ℓ (if $v_\ell \in V_1$ is a player-1 vertex), as either $x = \ell$ and the output on a decrement is $D_x(v_\ell) = D_\ell(v_\ell)$, or $x < \ell$ and $C_x(v_\ell) < 0$ (and the output on a reset is $D_\ell(v_\ell)$ by definition). Hence we have $v_{i-1} = D_i(v_i)$ whenever $v_i \in V_1$ is a player-1 vertex, for all $i = p+1, \dots, \ell$.

We now show by induction on i that $C_i(v_i) \leq \mathsf{TP}(v_i \cdots v_{p+1} v_p)$ for all $i \in \{p+1, \dots, \ell-1\}$, which implies, since $C_i(v_i) \geq 0$, that $\mathsf{TP}(v_i \cdots v_{p+1} v_p) \geq 0$, and thus all windows in the segment close within ℓ steps.

For the base case $i = p+1$, since $C_p(v_p) \leq 0$, the max subexpression at Line 7 and Line 9 of Algorithm 3 simplifies to $w(v_{p+1}, v_p)$, and accordingly we get either $C_{p+1}(v_{p+1}) = w(v_{p+1}, v_p)$ if $v_{p+1} \in V_1$ is a player-1 vertex, or $C_{p+1}(v_{p+1}) \leq w(v_{p+1}, v_p)$ if $v_{p+1} \in V_2$ is a player-2 vertex, which establishes the base case $C_{p+1}(v_{p+1}) \leq w(v_{p+1}, v_p) = \mathsf{TP}(v_{p+1} v_p)$.

For the induction step, let $i \in \{p+2, \dots, \ell-1\}$. Since $C_{i-1}(v_{i-1}) \geq 0$, we have $C_i(v_i) \leq w(v_i, v_{i-1}) + C_{i-1}(v_{i-1})$ by a similar argument as in the base case. By the induction hypothesis, we get $C_i(v_i) \leq w(v_i, v_{i-1}) + \mathsf{TP}(v_{i-1} \cdots v_p) = \mathsf{TP}(v_i \cdots v_p)$.

We show that the result holds no matter which is the initial memory state of the Mealy machine. It suffices to remark that with initial state $q_0 = i$ instead of $q_0 = \ell$, the prefix of the outcome until the first reset occurs (and the first segment starts) can be considered as a truncated segment (thus of length at most $i \leq \ell$) where the same argument can be used to show that all windows close within the length of the segment. □

Theorem 4.4. *If Player* 1 *wins in a non-stochastic game* $\mathcal{G}$ *with objective* $\mathsf{FWMP}(\ell)$*, then Player* 1 *has a winning strategy with memory of size* ℓ*.*

Proof. Since $\mathsf{FWMP}(\ell)$ is a prefix-independent objective, we have that the winning region $\langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{FWMP}(\ell))$ of Player 1 is a trap for Player 2 (Remark 2.1), and induces a subgame, say $\mathcal{G}_0$. We construct a winning strategy σ_1^{NS} for Player 1 in $\mathcal{G}_0$, with memory of size ℓ. Let there be $k+1$ calls to the subroutine $\mathsf{NonStocDirFWMP}$ from Algorithm 1. We denote by $(W_i)_{i \in \{1,\dots,k\}}$ the nonempty W_d returned by the i^{th} call to the subroutine, and let $A_i = \mathsf{Attr}_1(W_i)$. The A_i's are pairwise disjoint, and their union is $\bigcup_{i=1}^{k} A_i = \langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{FWMP}(\ell))$. For $i \in \{1, \dots, k\}$, inductively define $\mathcal{G}_i$ to be the subgame induced by the complement of A_i in $\mathcal{G}_{i-1}$. Since $\mathsf{DirFWMP}(\ell)$ is closed under suffixes, for all $i \in \{1, \dots, k\}$, we have that W_i is a trap for Player 2 in $\mathcal{G}_i$ (Remark 2.1).

Let $W = \bigcup_{i=1}^{k} W_i$ be the union of the regions W_i over all subgames $\mathcal{G}_i$, and let $A = \bigcup_{i=1}^{k} (A_i \setminus W_i)$ be the union of the regions $A_i \setminus W_i$ over all subgames $\mathcal{G}_i$, for $i \in \{1, \dots, k\}$. Note that $W \cap A = \varnothing$ and $W \cup A = \langle\langle 1 \rangle\rangle_{\mathcal{G}}(\mathsf{FWMP}(\ell))$.

We construct a strategy σ_1^{NS} that plays according to the (memoryless) attractor strategy in A, and according to the winning strategy σ_{d} for $\mathsf{DirFWMP}(\ell)$ objective (defined in Construction 4.3) in W. Formally, define the Mealy machine M_1^{NS} with ℓ states that defines σ_1^{NS}. The Mealy machine M_1^{NS} is given by the tuple $(Q_1^{\mathsf{NS}}, \ell, V, V \cup \{\epsilon\}, \Delta_1^{\mathsf{NS}}, \delta_1^{\mathsf{NS}})$, where

- the memory $Q_1^{\mathsf{NS}} = \{1, \dots, \ell\}$ of the Mealy machine stores a counter (modulo ℓ);
- the initial state is $q_0 = \ell$;
- the input alphabet is V, as the Mealy machine reads vertices of the game;
- the output alphabet is $V \cup \{\epsilon\}$, as the Mealy machine either outputs a vertex (upon reading a vertex of Player 1) or ϵ (upon reading a vertex of Player 2);

- The transition function $\Delta_1^{\mathsf{NS}}\colon Q_1^{\mathsf{NS}} \times V \to Q_1^{\mathsf{NS}}$ is defined as:

$$\Delta_1^{\mathsf{NS}}(i,v) = \begin{cases} \ell & \text{if } v \in A \quad \text{(follow attractor strategy)} \\ \Delta_{\mathsf{d}}(i,v) & \text{if } v \in W \quad \text{(follow } \sigma_{\mathsf{d}} \text{ for objective } \mathsf{DirFWMP}(\ell)) \end{cases}$$

- The output function $\delta_1^{\mathsf{NS}}\colon \{1,\dots,\ell\} \times V \to V \cup \{\epsilon\}$ is defined as follows. Here, $A(v)$ is the output of a (memoryless) attractor strategy to reach the set W.

$$\delta_1^{\mathsf{NS}}(i,v) = \begin{cases} \epsilon & \text{if } v \in V_2 \\ A(v) & \text{if } v \in A \cap V_1 \\ \delta_{\mathsf{d}}(i,v) & \text{if } v \in W \cap V_1 \end{cases}$$

We establish the correctness of the construction as follows. We show that the strategy σ_1^{NS} of Player 1 defined by M_1^{NS} is winning, that is for all strategies σ_2 of Player 2, the outcome π of the strategy profile $(\sigma_1^{\mathsf{NS}}, \sigma_2)$ satisfies the objective $\mathsf{FWMP}(\ell)$.

The crux is to show that one of the sets W_i for some $i \in \{1,\dots,k\}$ is never left from some point on. Intuitively, given the token is in A_i for some $i \in \{1,\dots,k\}$ (thus in $\mathcal{G}_i$), following σ_1^{NS} the token will either remain in A_i, or leave the subgame $\mathcal{G}_i$, thus entering A_j for a smaller index $j < i$. Repeating this argument (at most k times, as the index is decreasing) shows that the token eventually remains in some W_i ($i \in \{1,\dots,k\}$). From that point on, the strategy plays like σ_{d} (with some initial memory state $i \in \{1,\dots,\ell\}$) which ensures objective $\mathsf{DirFWMP}(\ell)$ (proof of Lemma 4.1), and thus from the initial state the objective $\mathsf{FWMP}(\ell)$ is satisfied. □

Remark 4.5. In every play π consistent with σ_1^{NS}, eventually, all windows close in at most ℓ steps. If Player 1 follows the strategy σ_1^{NS}, then irrespective of how Player 2's choices are made (whether they are deterministic or randomized), the outcome always satisfies the $\mathsf{FWMP}(\ell)$ objective. The proof of Theorem 4.4 thus continues to hold even if the strategy σ_2 of Player 2 is not deterministic. Since the constructed strategy σ_1^{NS} is a deterministic strategy, we have that deterministic strategies suffice for the $\mathsf{FWMP}(\ell)$ objective for Player 1, and memory of size ℓ suffices.

Lower bound on memory requirement for Player 1. In [CDRR15], the authors show an example of game with $\ell = 4$ where Player 1 requires memory at least 3. While it is not difficult to generalize this to arbitrary ℓ, we state it here for completeness.

Theorem 4.6. *There exists a family of games $\{\mathcal{G}_\ell\}_{\ell \geq 2}$ with objective $\mathsf{FWMP}(\ell)$ for Player 1 such that every winning strategy of Player 1 in $\mathcal{G}_\ell$ requires at least $\ell - 1$ memory.*

Proof. We describe the game $\mathcal{G}_\ell$ with objective $\mathsf{FWMP}(\ell)$ in Figure 4. The vertex u_0 belongs to Player 2. All other vertices in the game belong to Player 1. For each $i \in \{1,\dots,\ell-1\}$, there is an edge from (u_0, u_i) with payoff $-i$. For all $i \in \{1,\dots,\ell-2\}$, there is an edge (u_i, u_{i+1}) with payoff 0. There are $\ell - 1$ disjoint paths from $u_{\ell-1}$ to v, each of length $\ell - 1$. The i^{th} edge on the i^{th} path has payoff $+i$. All other edges in all paths from $u_{\ell-1}$ to v have payoff 0. Finally, there is an edge (v, u_0) with payoff 0.

If Player 2 moves the token from u_0 to u_i for $i \in \{1,\dots,\ell-1\}$, then Player 1 needs to ensure a payoff of at least $+i$ from $u_{\ell-1}$ in at most i steps to ensure that the window starting at u_0 closes in at most ℓ steps. When Player 2 moves the token from u_0 to u_i, we have that Player 1 must take the i^{th} path from $u_{\ell-1}$ so the window starting at u_0 closes in

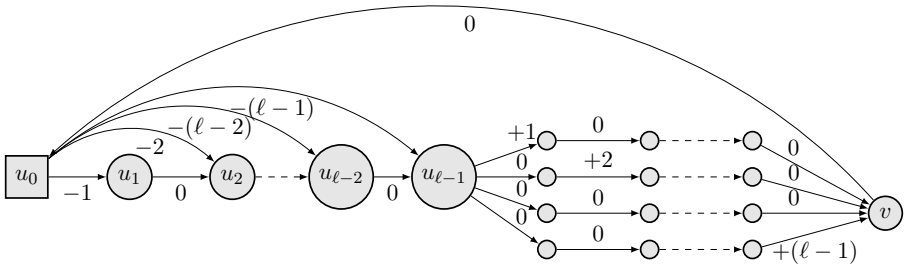


FIGURE 4. Memory $\ell - 1$ is necessary for Player 1.

at most ℓ steps. If Player 1 chooses any other successor of $u_{\ell-1}$, then the window starting at u_0 remains open for more than ℓ steps. Since there are $\ell - 1$ different choices for Player 1 from $u_{\ell-1}$, a Mealy machine defining a winning strategy of Player 1 requires at least $\ell - 1$ distinct states. Thus, a winning strategy of Player 1 in the game $\mathcal{G}_\ell$ requires at least $\ell - 1$ memory. □

Remark 4.7. The game $\mathcal{G}_\ell$ in Figure 4 shows that the memory requirement of Player 1 is $\ell - 1$ even if she uses randomized strategies. This is because after a certain point in the play, each time the token reaches $u_{\ell-1}$, Player 1 must choose the correct successor with probability 1 in order to win the $\mathsf{FWMP}(\ell)$ objective. Thus, randomization does not improve the lower bound for the size of the memory required.

4.2. **Memory requirement for Player 2 for FWMP objective.**

Upper bound on memory requirement for Player 2. Now we show that for $\overline{\mathsf{FWMP}(\ell)}$ objective, Player 2 has a winning strategy that requires at most $|V| \cdot \ell$ memory. This has been loosely stated in [CDRR15] without a formal proof. We use this result to show in Section 6 that the same memory bound for Player 2 also suffices in stochastic games. In fact, we show a stronger result that the memory required in the stochastic window mean-payoff games is no more than the optimal memory bounds for non-stochastic games.

Theorem 4.8. *Let $\mathcal{G}$ be a non-stochastic game with objective $\overline{\mathsf{FWMP}(\ell)}$ for Player 2. Then, Player 2 has a winning strategy with memory size at most $|V| \cdot \ell$.*

Proof. Since $\mathsf{FWMP}(\ell)$ is a prefix-independent objective, so is $\overline{\mathsf{FWMP}(\ell)}$. We have that $\langle\langle 2\rangle\rangle_{\mathcal{G}}(\overline{\mathsf{FWMP}(\ell)})$ is a trap for Player 1 (Remark 2.1) and induces a subgame, say $\mathcal{H}_0$, of $\mathcal{G}$. Let there be $k+1$ calls to the subroutine $\mathsf{GoodWin}$ from Algorithm 2, and let $\mathcal{H}_i$ be the subgame corresponding to the i^{th} call of the subroutine. We denote by $(W_i)_{i=1}^k$ the complement of W_{gw} in $\mathcal{H}_i$, where W_{gw} is returned by the i^{th} call to the subroutine, and let $A_i = \mathsf{Attr}_2(W_i)$. The A_i's are pairwise disjoint, and their union is $\bigcup_{i=1}^k A_i = \langle\langle 2\rangle\rangle_{\mathcal{G}}(\overline{\mathsf{FWMP}(\ell)})$.

We describe a winning strategy for the $\overline{\mathsf{FWMP}(\ell)}$ objective with memory $k \cdot \ell$, which is at most $|V| \cdot \ell$. The strategy is always in either *attractor mode* or *window-open mode*. When the game begins, it is in attractor mode. If the strategy is in attractor mode and the token is on a vertex $v \in A_i \setminus W_i$ for some $i \in \{1, \ldots, k\}$, then the attractor strategy is to eventually

reach W_i. If the token reaches W_i, then the strategy switches to window-open mode. Since all vertices in W_i are winning for Player 2 for the $\overline{\mathsf{GW}(\ell)}$ objective, he can keep the window open for ℓ more steps, provided that Player 1 does not move the token out of the subgame $\mathcal{H}_i$. If, at some point, Player 1 moves the token out of the subgame $\mathcal{H}_i$ to A_j for a smaller index $j < i$, then the strategy switches back to attractor mode, this time trying to reach W_j in the bigger subgame $\mathcal{H}_j$. Otherwise, if Player 2 keeps the window open for ℓ steps, then the strategy switches back to attractor mode until the token reaches a vertex in $\bigcup_{i=1}^{k} W_i$. This strategy can be defined by a Mealy machine M_2^{NS} with states $\{1,\dots,k\} \times \{1,\dots,\ell\}$, where the first component tracks the smallest subgame $\mathcal{H}_i$ in which the window started to remain open, and the second component indicates how many more steps the window needs to be kept open for. A formal description of M_2^{NS} is given in Construction 4.9. □

Construction 4.9. Let $W = \bigcup_{i=1}^{k} W_i$, and $A = \bigcup_{i=1}^{k}(A_i \setminus W_i)$. For $1 \le i \le k$, let H_i denote the set of vertices in the subgame $\mathcal{H}_i$. For all $v \in V$, let $\Gamma(v)$ denote the largest $j \ge 1$ such that $v \in H_j$. That is, $\mathcal{H}_{\Gamma(v)}$ is the smallest subgame that v belongs to. Given $j \ge 1$ and $v \in A \cap V_2 \cap (H_j \setminus H_{j+1})$, let $A^j(v)$ denote a successor vertex that Player 2 can choose to eventually reach the W_j region in $W \cap (H_j \setminus H_{j+1})$. This is given by a memoryless attractor strategy. Given $i \in \{1,\dots,\ell\}$, $j \ge 1$ and $v \in V_2 \cap H_j$, let $D_i^j(v)$ denote the best successor that Player 2 should choose from v to ensure that the window remains open for i more steps. These values can be computed for all $i \in \{1,\dots,\ell\}$ and for all $v \in H_j$ by running the $\mathsf{GoodWin}$ algorithm on the subgame $\mathcal{H}_j$. Recall from Line 9 in Algorithm 3 that $C_i(v) = \min_{(v,v')\in E}\{\max\{w(v,v'), w(v,v') + C_{i-1}(v')\}\}$. We let $D_i^j(v)$ be the successor vertex $v' \in E(v)$ of v such that the value of $\{\max\{w(v,v'), w(v,v') + C_{i-1}(v')\}\}$ is minimized. If there is more than one such v', we choose one arbitrarily.

We construct a Mealy machine M_2^{NS} that defines σ_2^{NS}, a winning strategy of Player 2. The Mealy machine M_2^{NS} is a tuple $(Q_2^{\mathsf{NS}}, (1,\ell), V, V \cup \{\epsilon\}, \Delta_2^{\mathsf{NS}}, \delta_2^{\mathsf{NS}})$ where

- the set of states Q_2^{NS} is the set $\{1,\dots,k\} \times \{1,\dots,\ell\}$,
- the initial state of the Mealy machine is $(1,\ell)$ irrespective of where the game begins in $\mathcal{H}_1$. We could also instead have set the initial state of the Mealy machine as $(\Gamma(v_{\mathsf{init}}),\ell)$, where v_{init} is the initial vertex of the game. However, to keep the initial state of the Mealy machine independent of the initial vertex of the game, we have the initial state of the Mealy machine as $(1,\ell)$. The transitions are defined such that the state of the Mealy machine is changed to $\Gamma(v)$ in the very next step.
- the input alphabet is V, same as in M_1^{NS},
- the output alphabet is $V \cup \{\epsilon\}$, same as in M_1^{NS}.

The transition function $\Delta_2^{\mathsf{NS}} \colon Q_2^{\mathsf{NS}} \times V \to Q_2^{\mathsf{NS}}$ is defined as follows:

$$\Delta_2^{\mathsf{NS}}(q,v) = \begin{cases} \Delta_2^{\mathsf{NS}}((1,\ell),v) & q = (j,i), v \in H_1 \setminus H_j, \text{for all } i \in \{1,\dots,\ell\}, j \in \{2,\dots,k\} \\ (\Gamma(v),\ell) & q = (j,\ell), v \in A \cap H_j, \text{for all } j \in \{1,\dots,k\} \\ (\Gamma(v),\ell-1) & q = (j,\ell), v \in W \cap H_j, \text{for all } j \in \{1,\dots,k\} \\ (j,i-1) & q = (j,i), v \in H_j, \text{for all } i \in \{2,\dots,\ell-1\}, j \in \{1,\dots,k\} \\ (1,\ell) & q = (j,1), v \in H_j, \text{for all } j \in \{1,\dots,k\} \end{cases}$$

Suppose the Mealy machine is in state $q \in Q_2^{\mathsf{NS}}$ and the token is in vertex $v \in V$. We describe the definition of $\Delta_2^{\mathsf{NS}}(q,v)$.

- If $q = (j, i)$ for $i \in \{1, \ldots, \ell\}$ and $j \in \{2, \ldots, k\}$, but $v \in H_1 \setminus H_j$, then this means that Player 1 must have moved the token out of the subgame $\mathcal{H}_j$. Player 2 may have been in the process of keeping the window open for ℓ steps in $\mathcal{H}_j$, but the move by Player 1 may have closed the window. The strategy switches to attractor mode, and the state of the Mealy machine changes to $\Delta_2^{\mathsf{NS}}((1, \ell), v)$. Note that Player 1 can only move the token out of a subgame finitely many times, and once the token is in $H_1 \setminus H_2$, then Player 1 can no longer move the token out of $\mathcal{H}_1$, and Player 2 will be able to keep the window open for ℓ steps without resetting the strategy. Now, for the remaining cases, suppose that if $q = (j, i)$, then $v \in H_j$.
- If $q = (j, \ell)$ for $j \in \{1, \ldots, k\}$ and $v \in A \cap H_j$, then the strategy is in attractor mode. Since $v \in H_j$, we have that $\Gamma(v) \geq j$, i.e. $\mathcal{H}_{\Gamma(v)}$ is a subgame of $\mathcal{H}_j$. When the game begins, or when Player 2 manages to keep the window open for ℓ steps, the first component resets to 1 according to the fifth type of transition in the definition of $\Delta_2^{\mathsf{NS}}(q, v)$. In such cases, it may happen that $\Gamma(v) > j$. Another possibility is that, at some point, the Mealy machine is in state (j', i) for $j' \in \{2, \ldots, k\}$ and $i \in \{1, \ldots, \ell\}$, and the token is in a vertex $u \in H_{j'}$. If u belongs to Player 1 and she moves the token from u to a vertex u' that is outside $H_{j'}$, then in the next turn, the first kind of transition of Δ_2^{NS} occurs, that is, $\Delta_2^{\mathsf{NS}}((1, \ell), u')$. In this case, it is possible that $\Gamma(u') > 1$. For such scenarios, the first component of the state of the Mealy machine updates to $\Gamma(v)$ to be an indicator of the smallest subgame that v belongs to. The second component remains equal to ℓ.
- If $q = (j, \ell)$ for $j \in \{1, \ldots, k\}$ and $v \in W \cap H_j$, then the strategy was in attractor mode, but switches to window-open mode in this step. Again, if the game begins, or if Player 2 manages to keep the window open for ℓ steps, or if Player 1 moves the token out of a subgame to a W-vertex, it may be the case that $\Gamma(v) > j$. Thus, the first component of the state of the Mealy machine updates to $\Gamma(v)$ to correctly reflect the smallest subgame that Player 2 begins to keep the window open in. The second component decreases by one, since Player 2 has begun to keep the window open from v and only needs to keep the window open for $\ell - 1$ steps after this.
- If $q = (j, i)$ for $j \in \{1, \ldots, k\}$ and $i \in \{2, \ldots, \ell - 1\}$, and $v \in H_j$, then the strategy is in window-open mode. The state of the Mealy machine changes to $(j, i - 1)$. The first component of the state of the Mealy machine keeps track of the smallest subgame in which Player 2 began to keep the window open in order to decide the optimal successor vertex $D_i^j(v)$, so it remains unchanged. The second component decreases to $i - 1$, and we now describe why. If v belongs to Player 2, and since H_j is a trap for Player 2, the successor v' of v chosen by Player 2 also belongs to H_j, and Player 2 chooses the successor that lets him keep the window open for $i - 1$ more steps after v'. On the other hand, if v belongs to Player 1, then the successor v' of v chosen by Player 1 may or may not belong to H_j. If v' belongs to H_j, then Player 2 can still keep the window open for $i - 1$ more steps after v', since Player 2 has been playing optimally for the objective $\overline{\mathsf{GW}(\ell)}$ restricted to the subgame $\mathcal{H}_j$. However, if Player 1 moves the token out of $\mathcal{H}_j$, then the window may have closed, but we let the state of the Mealy machine change to $(j, i - 1)$ regardless. In the next turn, the Mealy machine will be in state $(j, i - 1)$, but v' will not belong to H_j. Thus, by the first kind of transition of Δ_2^{NS}, after the next step, the state of the Mealy machine will change to $\Delta_2^{\mathsf{NS}}((1, \ell), v')$. The Mealy machine will behave as if its state had changed to $(1, \ell)$ (instead of $(j, i - 1)$) after reading v.

- If $q = (j, 1)$ for $j \in \{1, \dots, k\}$ and $v \in H_j$, then the strategy is in window-open mode. After this step, Player 2 has successfully kept the window open for ℓ steps, and the strategy switches to attractor mode. The second component resets to ℓ to indicate that the strategy switches to attractor mode. The first component of the state of the Mealy machine resets to 1, the way it was at the beginning of the game. Since $\Gamma(v) \geq 1$, the first component will correctly change to $\Gamma(v)$ in the next step by the second or third type of transition in the definition of $\Delta_2^{\mathsf{NS}}(q, v)$.

The output function $\delta_2^{\mathsf{NS}} \colon Q_2^{\mathsf{NS}} \times V \to V \cup \{\epsilon\}$ is defined as follows:

$$\delta_2^{\mathsf{NS}}(q, v) = \begin{cases} \epsilon & q \in Q_2^{\mathsf{NS}}, v \in V_1 \\ \delta_2^{\mathsf{NS}}((1, \ell), v) & q = (j, i), v \in V_2 \cap H_1 \setminus H_j, \text{for all } i \in \{1, \dots, \ell\}, j \in \{2, \dots, k\} \\ A^{\Gamma(v)}(v) & q = (j, \ell), v \in A \cap V_2 \cap H_j, \text{for all } j \in \{1, \dots, k\}, \\ D_\ell^{\Gamma(v)}(v) & q = (j, \ell), v \in W \cap V_2 \cap H_j, \text{for all } j \in \{1, \dots, k\}, \\ D_i^j(v) & q = (j, i), v \in V_2 \cap H_j, \text{for all } i \in \{1, \dots, \ell - 1\}, j \in \{1, \dots, k\} \end{cases}$$

Suppose the Mealy machine is in state $q \in Q_2^{\mathsf{NS}}$ and the token is in vertex $v \in V$. We describe the definition of $\delta_2^{\mathsf{NS}}(q, v)$.

- If $v \in V_1$, then the Mealy machine outputs ϵ since a strategy of Player 2 is not defined for prefixes ending with a Player 1 vertex.
- If $q = (j, i)$ for some $i \in \{1, \dots, \ell\}$ and $j \in \{2, \dots, k\}$, but $v \in H_1 \setminus H_j$, then as described in the definition of the transition function $\Delta_2^{\mathsf{NS}}(q, v)$, this means that Player 1 must have moved the token out of the subgame $\mathcal{H}_j$ when the last vertex before v was read. The Mealy machine behaves as if it was in state $(1, \ell)$, and outputs $\delta_2^{\mathsf{NS}}((1, \ell), v)$.
- If $q = (j, \ell)$ and $v \in A \cap V_2 \cap H_j$ for $j \in \{1, \dots, k\}$, then the strategy is in attractor mode. Player 2 must move the token by following the attractor strategy to reach $W_{\Gamma(v)}$. The vertex given by the attractor strategy is $A^{\Gamma(v)}(v)$.
- If $q = (j, \ell)$ and $v \in W \cap V_2 \cap H_j$ for $j \in \{1, \dots, k\}$, then the strategy switches to window-open mode. Player 2 has not started to keep to window open, but can now begin to keep the window open for ℓ steps. The best vertex to choose for this is given by the successor $D_\ell^{\Gamma(v)}(v)$.
- Finally, if $q = (j, i)$ and $v \in V_2 \cap H_j$ for $i \in \{1, \dots, \ell - 1\}$ and $j \in \{1, \dots, k\}$, then the strategy is in window-open mode. Player 2 has kept the window open for $\ell - i$ steps already, with the first vertex in the window from $H_j \setminus H_{j+1}$. He must keep the window open for i more steps, and hence chooses $D_i^j(v)$ as the successor vertex.

This concludes the construction of a Mealy machine defining a winning strategy of Player 2. □

Remark 4.10. The definitions $\Delta_2^{\mathsf{NS}}(q, v)$ and $\delta_2^{\mathsf{NS}}(q, v)$ are recursive definitions when $q = (j, i)$ and $v \in H_1 \setminus H_j$. Recall that when this is the case, we have $\Delta_2^{\mathsf{NS}}(q, v) = \Delta_2^{\mathsf{NS}}((1, \ell), v)$ and $\delta_2^{\mathsf{NS}}(q, v) = \delta_2^{\mathsf{NS}}((1, \ell), v)$. The output and the transition of the Mealy machine M_2^{NS} from the state (j, i) are as if the Mealy machine is actually in state $(1, \ell)$. If the Mealy machine is in state $(1, \ell)$, then since $j = 1$, it is not possible for v to belong to $H_1 \setminus H_1$ because it is an empty set. Thus, when $j = 1$, the transition and output functions do not make recursive calls to themselves. Hence, the depth of recursion in both Δ_2^{NS} and δ_2^{NS} is never greater than 1.

Remark 4.11. Note that the memory required for Player 2 to play optimally is $k \cdot \ell$, where k is the number of recursive calls to the GoodWin algorithm. This gives a tighter bound than that claimed in [CDRR15] since $k \leq |V|$.

Remark 4.12. Every play π that is consistent with the strategy constructed in Construction 4.9 has infinitely many open windows of length ℓ, and therefore satisfies the $\overline{\mathsf{FWMP}(\ell)}$ objective. If Player 2 follows this strategy, then the outcome is always winning for him, irrespective of how Player 1's choices are made (whether deterministic or randomized). The proof of Theorem 4.8 thus continues to hold even if the strategy of Player 1 is not deterministic. Since the strategy constructed in Construction 4.9 is a deterministic strategy, we have that deterministic strategies suffice for the $\overline{\mathsf{FWMP}(\ell)}$ objective for Player 2, and memory of size $|V| \cdot \ell$ suffices.

Lower bound on memory requirement for Player 2. In [CDRR15], it was shown that memoryless strategies do not suffice for Player 2. We improve upon this lower bound. Given a window length $\ell \geq 2$, for every $k \geq 1$, we construct a graph $\{\mathcal{G}_{k,\ell}\}$ with $2k + \ell - 1$ vertices such that every winning strategy of Player 2 in $\{\mathcal{G}_{k,\ell}\}$ requires at least $k + 1$ memory.

Theorem 4.13. *There exists a family of non-stochastic games $\{\mathcal{G}_{k,\ell}\}_{k\geq 1,\ell\geq 2}$ with objective $\mathsf{FWMP}(\ell)$ for Player 1 and edge weights $-1, 0, +1$ such that every winning strategy of Player 2 requires at least $\frac{1}{2}(|V| - \ell + 1) + 1$ memory, where $|V| = 2k + \ell - 1$.*

Proof. Let $A = \{a_1, \ldots, a_k\}$, $B = \{b_1, \ldots, b_k\}$, and $C = \{c_1, \ldots, c_{\ell-1}\}$ be pairwise disjoint sets. The vertices of $\mathcal{G}_{k,\ell}$ are $A \cup B \cup C$ with $V_1 = A \cup C$ and $V_2 = B$. Now we list the edges in $\mathcal{G}_{k,\ell}$:

(1) For all $p \in \{1, \ldots, k\}$ and $r \in \{1, \ldots, k\}$ such that $p \leq r$, we have an edge (a_p, b_r) with payoff -1.
(2) For all $p \in \{2, \ldots, k\}$, we have an edge (a_p, a_{p-1}) with payoff $+1$.
(3) For all $p \in \{2, \ldots, k\}$, we have an edge (a_p, b_{p-1}) with payoff $+1$.
(4) For all $p \in \{1, \ldots, k\}$, we have an edge $(b_p, c_{\ell-1})$ with payoff 0.
(5) For all $p \in \{1, \ldots, k\}$, we have an edge (b_p, a_p) with payoff $+1$.
(6) For all $p \in \{2, \ldots, \ell - 1\}$, we have an edge (c_p, c_{p-1}) with payoff 0.
(7) We have edges (c_1, a_k) and (c_1, b_k) with payoff $+1$ each.

Figure 5 shows the game $\mathcal{G}_{4,3}$.

Observe that the only open windows of length ℓ in the game $\mathcal{G}_{k,\ell}$ are sequences of the form $a_p b_r c_{\ell-1} \cdots c_1$ for all $p \leq r$. Also note that Player 2 has a winning strategy that wins starting from every vertex in the game:

- If the token is in C, then it eventually reaches c_1. From c_1, the token can move to either a_k or b_k. In the latter case, Player 2 moves the token from b_k to a_k. In both cases, the token reaches A.
- If the token is in A, then it cannot remain in A forever; it must eventually move to B. Moreover, Player 2 can ensure that the token eventually moves from A to B along an edge with negative payoff. Suppose whenever the token moves from A to B, it moves an edge with positive payoff, i.e., from $a_p \in A$ to $b_{p-1} \in B$ for some $p \in \{2, \ldots, k\}$. Then, Player 2 moves the token from b_{p-1} back to $a_{p-1} \in A$. The token then eventually reaches a_1 from which all out-edges have negative payoff, and must necessarily move to B along an edge with negative payoff.

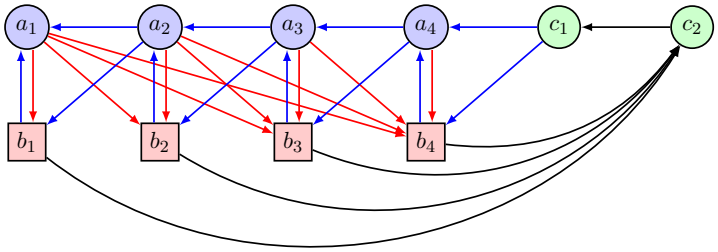


FIGURE 5. The game $\mathcal{G}_{4,3}$ with parameter $k = 4$ and window length $\ell = 3$. Red edges have payoff -1, black edges have payoff 0, and blue edges have payoff $+1$. Memory of size at least $k = 4$ is needed to define a winning strategy for the $\overline{\mathsf{FWMP}(\ell)}$ objective for Player 2 in this game.

$a_1b_1 \to c_2$	$a_2b_1 \to a_1$			
$a_1b_2 \to c_2$	$a_2b_2 \to c_2$	$a_3b_2 \to a_2$		
$a_1b_3 \to c_2$	$a_2b_3 \to c_2$	$a_3b_3 \to c_2$	$a_4b_3 \to a_3$	
$a_1b_4 \to c_2$	$a_2b_4 \to c_2$	$a_3b_4 \to c_2$	$a_4b_4 \to c_2$	$c_1b_4 \to a_4$

TABLE 1. Good choices $\chi(u, b_r)$ for all $u \in A \cup \{c_1\}$ and $b_r \in B$ in the game $\mathcal{G}_{4,3}$.

- Eventually, the token moves from a_p to b_r for some $p \leq r$. In this case, Player 2 moves the token from b_r to $c_{\ell-1}$ and eventually to c_1. The edge (a_p, b_r) has a negative payoff and the $\ell - 1$ edges on the path from b_r to c_1 have payoff 0 each. Hence, the window starting at a_p remains open for ℓ steps.
- Now, the token is on c_1 again and Player 2 can eventually keep the window open for ℓ steps again.

In this manner, Player 2 can ensure that open windows of length ℓ occur infinitely often in the play.

Good choices. When the token reaches a vertex $b_r \in B$, Player 2 can either move the token to $a_r \in A$ or to $c_{\ell-1} \in C$. Depending on which vertex the token was on before reaching b_r, one of the two choices is *good* for Player 2. If the token reaches b_r from the left or above, i.e., from a_p for $p \leq r$, then the edge (a_p, b_r) has negative payoff. In this case, it is good for Player 2 to move the token to $c_{\ell-1} \in C$ so that the window starting at a_p remains open for ℓ steps. Otherwise, if the token reaches b_r from the right, i.e., from a_{r+1}, then it is good for Player 2 to move the token to a_r so that an edge with negative payoff may eventually be taken.

For all $u \in A \cup \{c_1\}$, for all $b_r \in B$ such that (u, b_r) is an edge in $\mathcal{G}_{k,\ell}$, we denote by $\chi(u, b_r)$ the vertex a_r or $c_{\ell-1}$ that is good for Player 2. We list the good choices in the game $\mathcal{G}_{4,3}$ in Table 1. The columns are indexed by $u \in A \cup \{c_1\}$ and the rows are indexed by $b_r \in B$. If the edge (u, b_r) does not exist in the game, then the cell corresponding to this edge is left empty in the table.

In Lemma 4.14, we show that for each column in the table, there exists a distinct memory state in every Mealy machine defining a winning strategy of Player 2. This gives a lower bound of $k + 1$ on the number of states of such a Mealy machine. Since $\mathcal{G}_{k,\ell}$ has

$2k + \ell - 1$ vertices, the memory requirement of a winning strategy of Player 2 is at least $\frac{1}{2}(|V| - \ell + 1) + 1$. This concludes the proof of Theorem 4.13. □

Constructing Mealy machines. We show that every winning strategy of Player 2 in $\mathcal{G}_{k,\ell}$ requires at least $k + 1$ memory. Let σ_2^{NS} be a winning strategy of Player 2 in $\mathcal{G}_{k,\ell}$, and let $M_2^{\mathsf{NS}} = (Q_2^{\mathsf{NS}}, q_0, V, V \cup \{\epsilon\}, \Delta_2^{\mathsf{NS}}, \delta_2^{\mathsf{NS}})$ be a Mealy machine defining σ_2^{NS}.

For all $u \in A \cup \{c_1\}$, let Q_u denote the set of all states that M_2^{NS} could be in after reading a prefix ending in u, i.e., $Q_u = \{q \in Q_2^{\mathsf{NS}} \mid \exists \rho \in \mathsf{Prefs} : \hat{\Delta}_2^{\mathsf{NS}}(q_0, \rho \cdot u) = q\}$. Lemma 4.14 gives a lower bound on the number of states in M_2^{NS}.

Lemma 4.14. *Let σ_2^{NS} be a winning strategy for Player 2 for the $\overline{\mathsf{FWMP}(\ell)}$ objective, and let M_2^{NS} be a Mealy machine defining σ_2^{NS}. Then, for all vertices $u \in A \cup \{c_1\}$, there exists a state $q_u \in Q_u$ such that for all $b_r \in B \cap E(u)$, we have that $\delta_2^{\mathsf{NS}}(q_u, b_r) = \chi(u, b_r)$. Moreover, the Mealy machine M_2^{NS} has $k + 1$ distinct states.*

Proof. We prove the contrapositive. Suppose there exists a vertex $u \in A \cup \{c_1\}$ such that for all $q_u \in Q_u$, there exists a vertex $b_r \in B \cap E(u)$ such that $\delta_2^{\mathsf{NS}}(q_u, b_r) \neq \chi(u, b_r)$. Then, we show that σ_2^{NS} is not winning for Player 2. We show this by constructing a strategy σ_1 of Player 1 such that the outcome $(\sigma_1, \sigma_2^{\mathsf{NS}})$ satisfies $\mathsf{FWMP}(\ell)$ and is thus losing for Player 2. We have that either $u \in A$ or $u = c_1$.

- Suppose $u = c_1$.
  Irrespective of which vertex the game begins from, the strategy σ_1 tries to eventually move the token to c_1. If the token never reaches c_1, then this implies that the token never reaches $c_{\ell-1}$, and therefore, this implies that every time the token reaches B, Player 1 moves it to A and not C. Thus, no windows remain open for ℓ steps. Otherwise, the token eventually reaches c_1 after having seen at most one open window of length ℓ. After the token reaches c_1 for the first time, we show that under the assumption that for all $q_{c_1} \in Q_{c_1}$, there exists a vertex $b_r \in B \cap E(c_1)$ such that $\delta_2^{\mathsf{NS}}(q_{c_1}, b_r) \neq \chi(c_1, b_r)$, that subsequently there are no more open windows of size ℓ in the outcome.

  To see this, observe that $B \cap E(c_1) = \{b_k\}$ and $\chi(c_1, b_k) = a_k$. Since for all $q_{c_1} \in Q_{c_1}$ we have that $\delta_2^{\mathsf{NS}}(q_{c_1}, b_k) \neq \chi(c_1, b_k)$, we have that $\delta_2^{\mathsf{NS}}(q_{c_1}, b_k) = c_{\ell-1}$. That is, each time the token reaches b_k from c_1, the strategy σ_2^{NS} moves the token to $c_{\ell-1}$. Hence, the token is stuck in the cycle $(b_k c_{\ell-1} \cdots c_1)$ where every edge has nonnegative payoff, and no windows open.
- Otherwise, suppose $u = a_j \in A$ for some $j \in \{1, \ldots, k\}$.
  Irrespective of which vertex the game begins from, the strategy σ_1 eventually moves the token to a_j encountering at most one open window of length ℓ. Under the assumption in the contrapositive statement, we have that each time the token reaches a_j, there exists a successor b_r of a_j such that σ_2^{NS} does not play according to $\chi(u, b_r)$ from b_r. Specifically, at least one of the following holds:
  - If σ_1 moves the token from a_j to b_{j-1}, then σ_2^{NS} moves the token from b_{j-1} to $c_{\ell-1}$.
  - There exists $r \geq j$ such that if σ_1 moves the token from a_j to b_r, then σ_2^{NS} moves the token from b_r to a_r.

  In particular, if $j = 1$, then the first statement does not hold since b_0 is not defined. Hence, in the case of $j = 1$, the second statement holds.

To see this, suppose that when the token reaches a_j, the state of the Mealy machine M_2^{NS} becomes q'. When the token moves from a_j to a successor of a_j, suppose that the state of M_2^{NS} becomes $q \in Q_{a_j}$, i.e., we have that $\delta_2^{\mathsf{NS}}(q', a_j) = q$. Now, there exists $b_r \in B \cap E(a_j)$ such that $\delta_2^{\mathsf{NS}}(q, b_r) \neq \chi(a_j, b_r)$. Recall that $B \cap E(a_j) = \{b_{j-1}, b_j, \ldots, b_k\}$, and that $\chi(a_j, b_{j-1}) = a_{j-1}$, and $\chi(a_j, b_r) = c_{\ell-1}$ for all $r \geq j$. Therefore, we have that at least one of the following holds: $\delta_2^{\mathsf{NS}}(q, b_{j-1}) \neq a_{j-1}$ or there exists $r \geq j$ such that $\delta_2^{\mathsf{NS}}(q, b_r) \neq c_{\ell-1}$. Equivalently, at least one of the following holds: $\delta_2^{\mathsf{NS}}(q, b_{j-1}) = c_{\ell-1}$ or there exists $r \geq j$ such that $\delta_2^{\mathsf{NS}}(q, b_r) = a_r$.

In the first case, the window does not open at a_j. When the token reaches $c_{\ell-1}$, the strategy σ_1 eventually moves the token back to a_j without opening any new windows. In the second case, since the edge (a_j, b_r) has payoff -1, a window opens at a_j. However, since the edge (b_r, a_r) has payoff $+1$, this window closes in the next step and does not remain open for ℓ steps. The strategy σ_1 eventually moves the token back to a_j along vertices in A.

Therefore, each time the token reaches a_j, the strategy σ_1 moves the token to a successor b_r of a_j from which σ_2^{NS} does not play according to $\chi(a_j, b_r)$. This way there are subsequently no open windows of length ℓ in the outcome.

This completes the proof of the contrapositive. We now show that such a Mealy machine M_2^{NS} has at least $k+1$ distinct states. For all $u \in A \cup \{c_1\}$, let q_u denote a state in Q_u that plays in accordance with the good choices, i.e., for all $b_r \in B \cap E(u)$, we have that $\delta_2^{\mathsf{NS}}(q_u, b_r) = \chi(u, b_r)$. Then, for all i, j such that $0 \leq i < j < k$, we have that q_{a_i} and q_{a_j} are distinct states since $\delta_2^{\mathsf{NS}}(q_{a_i}, b_{j-1}) = c_{\ell-1}$ but $\delta_2^{\mathsf{NS}}(q_{a_j}, b_{j-1}) = a_{j-1}$. This gives k distinct states in M_2^{NS}. In addition to this, since $\delta_2^{\mathsf{NS}}(q_{c_1}, b_k) = a_k$ but $\delta_2^{\mathsf{NS}}(q_{a_j}, b_k) = c_{\ell-1}$ for all $j \in \{1, \ldots, k\}$, we have that q_{c_1} is distinct from each of the k distinct states found before. Thus M_2^{NS} has at least $k+1$ distinct states. □

If we allow Player 2 to use randomized strategies, then the upper bound on the memory size required for Player 2 improves to memoryless strategies.

Proposition 4.15. *A memoryless randomized winning strategy exists for Player* 2 *for the* $\overline{\mathsf{FWMP}(\ell)}$ *objective.*

A memoryless randomized winning strategy for Player 2 for the $\overline{\mathsf{FWMP}(\ell)}$ objective is the following: Recall that the winning region of Player 2 is a trap for Player 1. In each turn, Player 2 picks an out-edge uniformly at random out of all out-edges that keep the token in the trap. It is always the case that with probability 1, an open window of length ℓ will eventually occur in the play. Thus, following this strategy, with probability 1, infinitely many open windows of length ℓ occur in the outcome, resulting in Player 2 winning the $\overline{\mathsf{FWMP}(\ell)}$ objective.

Given a (deterministic) winning strategy σ_2^{NS} of Player 2 for the $\overline{\mathsf{FWMP}(\ell)}$ objective, the following lemma gives an upper bound on the number of steps between consecutive open windows of length ℓ in any play consistent with σ_2^{NS}. This lemma is used in Section 6, where we construct an almost-sure winning strategy of Player 2 for the $\overline{\mathsf{FWMP}(\ell)}$ objective.

Lemma 4.16. *Let* $\mathcal{G}$ *be a non-stochastic game such that all vertices in* $\mathcal{G}$ *are winning for Player* 2*, that is,* $\langle\langle 2 \rangle\rangle_{\mathcal{G}}(\overline{\mathsf{FWMP}(\ell)}) = V$*. Let* σ_2^{NS} *be a finite-memory strategy of Player* 2 *of memory size* M *that is winning for* $\overline{\mathsf{FWMP}(\ell)}$ *from all vertices in* $\mathcal{G}$*. Then, for every play* π

of $\mathcal{G}$ consistent with σ_2^{NS}, every infix of π of length $\mathsf{M} \cdot |V| \cdot \ell$ contains an open window of length ℓ.

Proof. Since σ_2^{NS} has memory of size M, fixing this strategy in $\mathcal{G}$ gives a one-player game $\mathcal{G}^{\sigma_2^{\mathsf{NS}}}$ with $\mathsf{M} \cdot |V|$ vertices. Then, the claim is that every path of length $\mathsf{M} \cdot |V| \cdot \ell$ in $\mathcal{G}^{\sigma_2^{\mathsf{NS}}}$ contains an open window of length ℓ. Suppose towards a contradiction that there exists a path of length $\mathsf{M} \cdot |V| \cdot \ell$ in the one-player game that does not contain an open window of length ℓ. Since every window is closed in no more than ℓ steps, and the window is closed at the initial vertex to begin with, there are at least $\mathsf{M} \cdot |V| + 1$ vertices in this path where a window closes. Since there are only $\mathsf{M} \cdot |V|$ vertices in $\mathcal{G}^{\sigma_2^{\mathsf{NS}}}$, by the pigeonhole principle, there exists a vertex u that is visited twice in this path, both times with the window closed. Thus, the path contains a cycle without open windows of length ℓ. Since Player 1 can reach this cycle and loop in it forever, it gives an outcome that is winning for Player 1 in $\mathcal{G}$, which is a contradiction. □

5. Reducing stochastic games to non-stochastic games

In this section, we recall a sufficient condition that allows us to solve stochastic games by solving, as a subroutine, a non-stochastic game with the same objective. The sufficient condition was presented in [CHH09b] for solving finitary Streett objectives and can be generalized to arbitrary prefix-independent objectives. Under this condition, the qualitative problems for stochastic games can be solved as efficiently (up to a factor of $|V|^2$) as non-stochastic games with the same objective. Also, it follows that the memory requirement for Player 1 to play optimally in stochastic games is the same as in non-stochastic games with the same objective.

We now describe the sufficient condition that we call the sure-almost-sure property. Given a stochastic game $\mathcal{G}$, let $\mathcal{G}_{\mathsf{NS}} = ((V, E), (V_1, V_2 \cup V_\Diamond, \varnothing), w)$ be the *(adversarial) non-stochastic game corresponding to* $\mathcal{G}$, obtained by changing all probabilistic vertices to Player 2 vertices. We omit the probability function in the tuple since there are no more probabilistic vertices.

Definition 5.1 (Sure-almost-sure (SAS) property)**.** A prefix-independent objective φ in a game $\mathcal{G}$ satisfies the SAS property if $\langle\langle 2 \rangle\rangle_{\mathcal{G}_{\mathsf{NS}}}(\overline{\varphi}) = V$ implies $\langle\langle 2 \rangle\rangle^{\mathsf{AS}}_{\mathcal{G}}(\overline{\varphi}) = V$, that is, if Player 2 wins the objective $\overline{\varphi}$ from every vertex in $\mathcal{G}_{\mathsf{NS}}$, then Player 2 almost-surely wins the same objective $\overline{\varphi}$ from every vertex in $\mathcal{G}$.

The definition of the SAS property implies that if there exists a vertex from which Player 1 wins the objective φ positively in the stochastic game $\mathcal{G}$, then there exists a vertex from which Player 1 wins the same objective φ in the non-stochastic game $\mathcal{G}_{\mathsf{NS}}$. Note that every prefix-independent objective satisfies the converse of the SAS property since if Player 2 wins almost-surely from all vertices in $\mathcal{G}$, then since he controls all probabilistic vertices in $\mathcal{G}_{\mathsf{NS}}$, he wins from all vertices in $\mathcal{G}_{\mathsf{NS}}$ by choosing optimal successors of probabilistic vertices.

Remark 5.2. We show in Section 6 that for all stochastic games $\mathcal{G}$, the objectives $\mathsf{FWMP}_{\mathcal{G}}(\ell)$ and $\mathsf{BWMP}_{\mathcal{G}}$ satisfy the SAS property. As noted earlier in Section 3, the $\mathsf{FWMP}_{\mathcal{G}}(1)$ objective is equivalent to a coBüchi objective, and thus, coBüchi satisfies the SAS property as well. One can show that the generalized coBüchi objective, that is, an objective that is a union of several coBüchi objectives also satisfies the SAS property. In particular, objectives such as

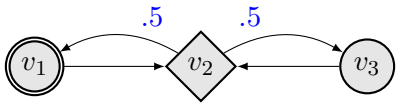


FIGURE 6. Büchi objective does not satisfy the SAS property in this game.

$\mathsf{BWMP}_{\mathcal{G}}$, and finitary parity and finitary Streett objectives (as defined in [CHH09b]) can be seen as countable unions of coBüchi objectives, and these objectives satisfy the SAS property.

Now, we present an example of objective that *does not* satisfy the SAS property. Consider the example in Figure 6. The objective φ in this game is a Büchi objective: a play π satisfies the Büchi objective if π visits vertex v_1 infinitely often. Although from every vertex, with positive probability (in fact, with probability 1), a play visits v_1 infinitely often, from none of the vertices, Player 1 can ensure the Büchi objective in the non-stochastic game $\mathcal{G}_{\mathsf{NS}}$.

The following theorem states that if an objective φ satisfies the SAS property, then solving the positive (resp., almost-sure) satisfaction problem can be done within a linear (resp., quadratic) factor of the time needed to solve non-stochastic games with the same objective.

Theorem 5.3. *Given $\mathcal{G}$ and φ, suppose in every subgame $\mathcal{G}'$ of $\mathcal{G}$, the objective φ restricted to $\mathcal{G}'$ satisfies the* SAS *property. Let $\mathsf{NonStocWin}_{\varphi}(\mathcal{G}_{\mathsf{NS}})$ be an algorithm computing $\langle\langle 1\rangle\rangle_{\mathcal{G}_{\mathsf{NS}}}(\varphi)$ in $\mathcal{G}_{\mathsf{NS}}$ in time $\mathbb{C}$. Then, the positive and almost-sure satisfaction of φ can be decided in time $\mathcal{O}(|V| \cdot (\mathbb{C} + |E|))$ and $\mathcal{O}(|V|^2 \cdot (\mathbb{C} + |E|))$ respectively.*

Moreover, for positive and almost-sure satisfaction of φ, the memory requirement for Player 1 to play optimally in stochastic games is no more than that for non-stochastic games.

Theorem 5.3 does not give bounds on memory requirement for winning Player 2 strategies for objective φ in the stochastic game, but we provide such bounds specifically for $\mathsf{FWMP}(\ell)$ and BWMP in Section 6. The proof of the theorem appears shortly after Corollary 5.4.

Finally, we look at the quantitative decision problem. The quantitative satisfaction for φ can be decided in NP^B ([CHH09b, Theorem 6]), where B is an oracle deciding positive and almost-sure satisfaction problems for φ. It is not difficult to see that the quantitative satisfaction for φ can be decided in $\mathsf{NP}^B \cap \mathsf{coNP}^B$. Moreover, as stated in [CHH09b, Definition 2], the vertices of a stochastic game can be partitioned into classes from which Player 1 wins φ with the same maximal probability. From [CHH09b, Lemma 7], a strategy of Player 1 that is almost-sure winning in every class for the objective $\varphi \cup \mathsf{Reach}(Z)$ for some suitable subset Z of the class is a winning strategy of Player 1 for the quantitative satisfaction of φ. Analogously, a strategy of Player 2 that is positive winning in every class for objective $\overline{\varphi} \cap \mathsf{Safe}(\overline{Z})$, where $\overline{Z}$ is the complement of Z in the class, is a winning strategy for Player 2. Thus, the memory requirement of winning strategies for both players for the quantitative decision problem is no greater than that for the qualitative decision problem.

Corollary 5.4. *Given $\mathcal{G}$ and φ as described in Theorem 5.3, let B be an oracle deciding the qualitative satisfaction of φ. Then, the quantitative satisfaction of φ is in $\mathsf{NP}^B \cap \mathsf{coNP}^B$. Moreover, the memory requirement of optimal strategies for both players is no greater than that for the positive and almost-sure satisfaction of φ.*

Now, we describe an algorithm $\mathsf{PosWin}_{\varphi}$ to compute the positive winning region of Player 1 in $\mathcal{G}$ with objective φ. The algorithm uses $\mathsf{NonStocWin}_{\varphi}$ as a subroutine. Then, we describe an algorithm ASWin_{φ} that uses $\mathsf{PosWin}_{\varphi}$ as a subroutine to compute the almost-sure winning

Algorithm 4 $\mathsf{PosWin}_\varphi(\mathcal{G})$

Input: $\mathcal{G} = ((V, E), (V_1, V_2, V_\Diamond), \mathbb{P}, w)$, the stochastic game
Output: The set of vertices from which Player 1 positively wins objective φ in $\mathcal{G}$

1: $W_1 \leftarrow \mathsf{NonStocWin}_\varphi(\mathcal{G}_{\mathsf{NS}})$
2: **if** $W_1 = \varnothing$ **then**
3: **return** $\varnothing$
4: **else**
5: $A_1 \leftarrow \mathsf{PosAttr}_1(W_1)$
6: **return** $A_1 \cup \mathsf{PosWin}_\varphi(\mathcal{G} \restriction (V \setminus A_1))$

Algorithm 5 $\mathsf{ASWin}_\varphi(\mathcal{G})$

Input: $\mathcal{G} = ((V, E), (V_1, V_2, V_\Diamond), \mathbb{P}, w)$, the stochastic game
Output: The set of vertices in V from which Player 1 almost-surely wins φ in $\mathcal{G}$

1: $W_2 \leftarrow V \setminus \mathsf{PosWin}_\varphi(\mathcal{G})$
2: **if** $W_2 = \varnothing$ **then**
3: **return** V
4: **else**
5: $A_2 \leftarrow \mathsf{PosAttr}_2(W_2)$
6: **return** $\mathsf{ASWin}_\varphi(\mathcal{G} \restriction (V \setminus A_2))$

region for Player 1 for the objective φ. The algorithms and their correctness proof are the same as in the case of finitary Streett objectives described in [CHH09b].

Proof of Theorem 5.3. We recall the recursive procedures (in Algorithm 4 and Algorithm 5) to compute the positive and the almost-sure winning regions for Player 1 in stochastic games with an objective that satisfies the SAS property. The algorithms are similar to the case of finitary Streett objectives [CHH09b], which satisfy the SAS property. Note that, because of determinacy, the positive winning region $\langle\langle 1 \rangle\rangle^{\mathsf{Pos}}_{\mathcal{G}}(\varphi)$ for Player 1 is the complement of the almost-sure winning region $\langle\langle 2 \rangle\rangle^{\mathsf{AS}}_{\mathcal{G}}(\overline{\varphi})$ for Player 2.

The depth of recursive calls in Algorithm 4 is bounded by $|V|$, the number of vertices in $\mathcal{G}$, as the argument in the recursive call (Line 6) has strictly fewer vertices than $|V|$, since $A_1 \neq \varnothing$. The Player 1 positive attractor is computed in time $\mathcal{O}(|E|)$, and suppose $\mathsf{NonStocWin}_\varphi$ runs in time $\mathbb{C}$. These subroutines are executed at most $|V|$ times, once in every depth of the recursive call. Thus, the total running time of PosWin_φ is $\mathcal{O}(|V| \cdot (\mathbb{C} + |E|))$.

Let W_1^i and A_1^i denote the sets W_1 and A_1 computed in the recursive call of depth i respectively. Recall that the sets A_1^i form a partition of the positive winning region $\langle\langle 1 \rangle\rangle^{\mathsf{Pos}}_{\mathcal{G}}(\varphi)$ for Player 1, and that for all i, we have that $W_1^i \subseteq A_1^i$. Let σ_1^{NS} be a winning strategy of Player 1 in the non-stochastic game $\mathcal{G}_{\mathsf{NS}}$. We construct a positive-winning strategy σ_1^{Pos} for Player 1 in the stochastic game as follows. Given a prefix $\rho \in \mathsf{Prefs}^1_{\mathcal{G}}$, we determine the value of i for which $\mathsf{Last}(\rho) \in A_1^i$. Then, if $\mathsf{Last}(\rho) \in W_1^i$, then σ_1^{Pos} plays like σ_1^{NS}, that is $\sigma_1^{\mathsf{Pos}}(\rho) = \sigma_1^{\mathsf{NS}}(\rho)$; otherwise, $\mathsf{Last}(\rho) \in A_1^i \setminus W_1^i$, and let $\sigma_1^{\mathsf{Pos}}(\rho) = \sigma_1^{\mathrm{Attr}}(\rho)$ where σ_1^{Attr} is a positive-attractor strategy to W_1^i (which is memoryless, i.e., $\sigma_1^{\mathrm{Attr}}(\rho) = \sigma_1^{\mathrm{Attr}}(\mathsf{Last}(\rho))$). Then, σ_1^{Pos} is a positive-winning strategy for Player 1 from all vertices in $\mathsf{PosWin}_\varphi(\mathcal{G})$.

The depth of recursive calls in Algorithm 5 is also bounded by $|V|$. The set W_2 from which Player 2 wins almost-surely for objective $\overline{\varphi}$ is computed in time $\mathcal{O}(|V| \cdot (\mathbb{C} + |E|))$, and

the Player 2 positive attractor is computed in time $\mathcal{O}(|E|)$. This leads to a total running time $\mathcal{O}(|V|^2 \cdot (\mathbb{C} + |E|))$. □

The following lemma, which is a special case of Theorem 1 in [Cha07] where it has been proved for concurrent stochastic games, allows us to use results from the computation of the positive winning region for Player 1 in $\mathcal{G}$ to obtain the almost-sure winning region for Player 1 in $\mathcal{G}$. In Theorem 1 in [Cha07], it has been shown that for a prefix-independent objective, if there exists a vertex from which Player 1 wins positively, then there exists a vertex from which Player 1 wins almost-surely. Since prefix-independent objectives are closed under complementation, the theorem also holds for Player 2. Considering the theorem for Player 2, and taking the contrapositive, we have the following lemma for the special case of turn-based zero-sum stochastic games. In particular, in Algorithm 5, since $W_2 = \varnothing$ denotes that all vertices are positively winning for Player 1, this gives that all vertices are almost-surely winning for Player 1 by Lemma 5.5.

Lemma 5.5. [Cha07, Theorem 1] *If Player* 1 *positively wins a stochastic game* $\mathcal{G}$ *with a prefix-independent objective* φ *from every vertex in* V*, then Player* 1 *almost surely wins* $\mathcal{G}$ *with objective* φ *from every vertex in* V*, that is, if* $\langle\langle 1\rangle\rangle_{\mathcal{G}}^{\mathsf{Pos}}(\varphi) = V$*, then* $\langle\langle 1\rangle\rangle_{\mathcal{G}}^{\mathsf{AS}}(\varphi) = V$*.*

Let σ_1^{Pos} be the strategy of Player 1 as described in the PosWin_φ algorithm. An almost-sure winning strategy σ_1^{AS} of Player 1 for objective φ is the same as σ_1^{Pos}; if Player 1 follows the strategy σ_1^{Pos}, then she almost-surely satisfies φ from all vertices in $\mathsf{ASWin}_\varphi(\mathcal{G})$. For both positive and almost-sure winning, Player 1 does not require any additional memory in the stochastic game compared to the non-stochastic game.[1]

6. Reducing stochastic window mean-payoff games: A special case

In this section, we show that for all stochastic games $\mathcal{G}$ and for all $\ell \geq 1$, the fixed window mean-payoff objective $\mathsf{FWMP}_{\mathcal{G}}(\ell)$ and the bounded window mean-payoff objective $\mathsf{BWMP}_{\mathcal{G}}$, which are prefix independent objectives, satisfy the SAS property of Definition 5.1. Thus, by Theorem 5.3, we obtain bounds on the complexity and memory requirements of Player 1 for positive satisfaction and almost-sure satisfaction of these objectives. The algorithms to compute the positive and the almost-sure winning regions of Player 1 for $\mathsf{FWMP}(\ell)$ (resp., BWMP) objective can be obtained by instantiating Algorithms 4 and 5 respectively with φ equal to $\mathsf{FWMP}(\ell)$ (resp., BWMP). We also show that for both these objectives, the memory requirements of Player 2 to play optimally for positive and almost-sure winning in stochastic games is no more than that of the non-stochastic games.

6.1. Fixed window mean-payoff objective.

We show that the SAS property holds for the objective $\mathsf{FWMP}(\ell)$ for all stochastic games $\mathcal{G}$ and for all $\ell \geq 1$.

Lemma 6.1. *For all stochastic games* $\mathcal{G}$ *and for all* $\ell \geq 1$*, the objective* $\mathsf{FWMP}(\ell)$ *satisfies the* SAS *property.*

[1] If deterministic strategies suffice for Player 1 in non-stochastic games to win an objective φ satisfying the SAS property, then deterministic strategies also suffice for Player 1 for the positive and almost-sure winning strategies of the same objective φ in stochastic games.

Proof. We need to show that if $\langle\langle 2\rangle\rangle_{\mathcal{G}_{\mathsf{NS}}}(\overline{\mathsf{FWMP}(\ell)}) = V$, then $\langle\langle 2\rangle\rangle^{\mathsf{AS}}_{\mathcal{G}}(\overline{\mathsf{FWMP}(\ell)}) = V$.

If $\langle\langle 2\rangle\rangle_{\mathcal{G}_{\mathsf{NS}}}(\overline{\mathsf{FWMP}(\ell)}) = V$, then from Theorem 4.8, there exists a finite-memory strategy σ_2^{NS} (say, with memory M) of Player 2 that is winning for objective $\overline{\mathsf{FWMP}(\ell)}$ from every vertex in $\mathcal{G}_{\mathsf{NS}}$. Given such a strategy, we construct below a strategy σ_2^{AS} of Player 2 in the stochastic game $\mathcal{G}$ that is almost-sure winning for $\overline{\mathsf{FWMP}(\ell)}$ from every vertex in $\mathcal{G}$.

In $\mathcal{G}_{\mathsf{NS}}$, Player 2 chooses the successor of vertices in $V_2 \cup V_\Diamond$ according to the strategy σ_2^{NS}. Since σ_2^{NS} is a winning strategy, Player 2 can satisfy the $\overline{\mathsf{FWMP}(\ell)}$ objective irrespective of Player 1's strategy. In the stochastic game $\mathcal{G}$, however, Player 2 has less control. He can only choose successors for vertices in V_2, while the successors for vertices in $V_\Diamond$ are chosen according to the probability function $\mathbb{P}$ that is specified in the game. It is possible that for a probabilistic vertex, the successor chosen by the distribution is not what Player 2 would have chosen, resulting in a potentially worse outcome for him. We use the fact that $\overline{\mathsf{FWMP}(\ell)}$ is a Büchi-like objective (Player 2 would like to *always eventually* see an open window of length ℓ), and that σ_2^{NS} is winning from every vertex to show that despite having control over fewer vertices, Player 2 has a strategy σ_2^{AS} that is almost-sure winning for the $\overline{\mathsf{FWMP}(\ell)}$ objective from every vertex in the stochastic game $\mathcal{G}$.

Let $\pi = v_0 v_1 \cdots$ be an outcome in the stochastic game $\mathcal{G}$ when Player 2 follows the strategy σ_2^{NS}, i.e., for all v_i in π such that $v_i \in V_2$, we have that $v_{i+1} = \sigma_2^{\mathsf{NS}}(v_0 v_1 \cdots v_i)$. For all probabilistic vertices $v_j \in V_\Diamond$ in the play π, if the successor vertex of v_j chosen by the probability distribution is not equal to the successor vertex $\sigma_2^{\mathsf{NS}}(v_0 v_1 \cdots v_j)$ of v_j given by the strategy σ_2^{NS}, i.e., if $v_{j+1} \neq \sigma_2^{\mathsf{NS}}(v_0 v_1 \cdots v_j)$, then we say that a *deviation* from the strategy σ_2^{NS} occurs in π at v_j. Note that the prefix $v_0 v_1 \cdots v_j v_{j+1}$ with the deviation does never appears in any play in $\mathcal{G}_{\mathsf{NS}}$ that is consistent with σ_2^{NS}.

Some deviations may cause the outcome to be losing for Player 2. Therefore, starting with the strategy σ_2^{NS}, we construct a strategy σ_2^{AS} that mimics σ_2^{NS} as long as no such deviations occur, and *resets* otherwise, i.e., the strategy forgets the prefix of the play before the deviation. We call the strategy σ_2^{AS} a *reset strategy*. We see in Construction 6.3, given a Mealy machine M_2^{NS} that defines σ_2^{NS}, how to construct a Mealy machine M_2^{AS} that defines the reset strategy σ_2^{AS}. In the construction, we show that the memory size of σ_2^{AS} is no more than that of σ_2^{NS}. Therefore, all games with objective $\mathsf{FWMP}(\ell)$ satisfy the SAS property. □

In Example 6.2, we see an example of a stochastic game $\mathcal{G}$ along with a strategy σ_2^{NS} that is winning for Player 2 from all vertices in the adversarial game $\mathcal{G}_{\mathsf{NS}}$. We show that this strategy σ_2^{NS} need not be almost-sure winning for Player 2 in the stochastic game $\mathcal{G}$ and then give an intuition on how to use σ_2^{NS} to construct a reset strategy σ_2^{AS} that is almost-sure winning from all vertices in $\mathcal{G}$. In Construction 6.3, we formally show how to obtain a Mealy machine that defines σ_2^{AS} from a Mealy machine that defines σ_2^{NS} without adding any new states.

Example 6.2. Figure 7 shows a stochastic game $\mathcal{G}$ with objective $\overline{\mathsf{FWMP}(3)}$ for Player 2. The edges (v_2, v_4) and (v_3, v_5) have negative payoffs and all other edges have zero payoff. For each probabilistic vertex $v \in V_\Diamond$, we have that the probability function $\mathbb{P}(v)$ is a uniform distribution, i.e., we have: $\mathbb{P}(v_2)(v_4) = \mathbb{P}(v_2)(v_5) = \frac{1}{2}$, $\mathbb{P}(v_3)(v_4) = \mathbb{P}(v_3)(v_5) = \frac{1}{2}$, and $\mathbb{P}(v_6)(v_7) = \mathbb{P}(v_6)(v_8) = \frac{1}{2}$.

Figure 8 shows a Mealy machine M_2^{NS} defining a strategy σ_2^{NS} that is winning for $\overline{\mathsf{FWMP}(3)}$ from all vertices in the adversarial game $\mathcal{G}_{\mathsf{NS}}$. In figures, for states q_i, q_j of the

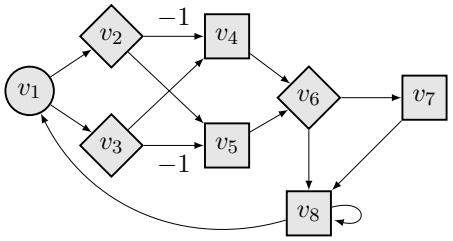


FIGURE 7. The game $\mathcal{G}$ with objective $\overline{\mathsf{FWMP}(3)}$ for Player 2 from Example 6.2. All edges except (v_2, v_4) and (v_3, v_5) have payoff 0. For all probabilistic vertices $v \in V_\Diamond$, the probability function $\mathbb{P}(v)$ is a uniform distribution over the out-neighbours $E(v)$ of v.

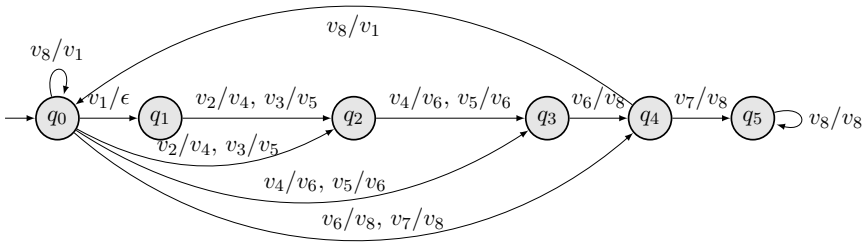


FIGURE 8. Mealy machine M_2^{NS} defining a strategy σ_2^{NS} that is winning from all vertices in $\mathcal{G}_{\mathsf{NS}}$ for $\overline{\mathsf{FWMP}(3)}$.

Mealy machine and for vertices v, v' of the game, an edge from state q_i to q_j with label v/v' denotes that the next state of the Mealy machine is $\Delta(q_i, v) = q_j$ and the next vertex is $\delta(q_i, v) = v'$. To see that σ_2^{NS} is a winning strategy, note that each time the token reaches v_1, we have that Player 1 may move the token from v_1 to either v_2 or v_3. The strategy σ_2^{NS} moves the token from v_2 and v_3 to v_4 and v_5 respectively, ensuring that a window opens. Then, when the token reaches v_6, the strategy always moves the token to v_8 and then back to v_1. The Mealy machine never moves the token to v_7. If the game begins in v_7, then the token is moved to v_8 and then to v_1 and the token never goes to v_7 after that. Each time the token reaches v_1, the token must move to v_2 or v_3. Since there are no edges with positive payoff, the window starting at v_2 or v_3 never closes, and in particular, remains open for 3 steps. Since the token reaches v_1 infinitely often, by following this strategy, Player 2 ensures for all strategies of Player 1, the outcome contains infinitely many open windows of length 3 and thus, the strategy σ_2^{NS} is winning for Player 2 for $\overline{\mathsf{FWMP}(3)}$ in $\mathcal{G}_{\mathsf{NS}}$.

Observe that the strategy σ_2^{NS} is not almost-sure winning for Player 2 from any vertex in the stochastic game $\mathcal{G}$. If Player 2 follows the strategy σ_2^{NS} in $\mathcal{G}$, then the probability that he wins $\overline{\mathsf{FWMP}(3)}$ is less than 1. This is because when the token reaches v_6, then with probability $\frac{1}{2}$, it moves to v_7. Once that happens, the state of the Mealy machine changes to q_5 and the strategy moves the token to v_8 and keeps it there forever. No new windows open, and the outcome is losing for $\overline{\mathsf{FWMP}(3)}$. Hence, if Player 2 follows the strategy σ_2^{NS} in $\mathcal{G}$, then with positive probability, he does not win $\overline{\mathsf{FWMP}(3)}$.

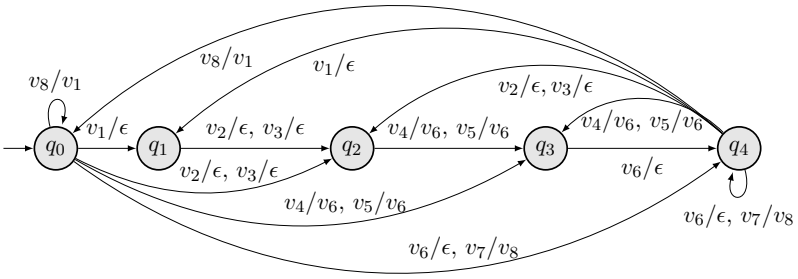


FIGURE 9. Part of the Mealy machine M_2^{AS} defining a reset strategy that is almost-sure winning from all vertices in $\mathcal{G}$. Reset transitions out of q_1, q_2, and q_3 have been omitted from the figure.

This is not an issue in the non-stochastic game $\mathcal{G}_{\mathsf{NS}}$ since the $q_4 \xrightarrow{v_7} q_5$ transition is never taken in $\mathcal{G}_{\mathsf{NS}}$ as long as Player 2 plays according to σ_2^{NS}. If Player 2 plays according to σ_2^{NS}, then in any transition that changes the state of the Mealy machine to q_4, the token moves to v_8 by that transition. There does not exist any prefix ρ in $\mathcal{G}_{\mathsf{NS}}$ that is consistent with σ_2^{NS} that causes the Mealy machine to take the $q_4 \xrightarrow{v_7} q_5$ transition. We call such transitions that cannot be taken in M_2^{NS} in $\mathcal{G}_{\mathsf{NS}}$ when Player 2 plays according to σ_2^{NS} *unreachable*. We see that there are no reachable transitions that change the state of M_2^{NS} to q_5, and thus, the outgoing transition $q_5 \xrightarrow{v_8} q_5$ from q_5 is also unreachable. One can verify that all transitions in M_2^{NS} other than the two mentioned above are reachable in $\mathcal{G}_{\mathsf{NS}}$.

If Player 2 follows the strategy σ_2^{NS} in the stochastic game $\mathcal{G}$, then a deviation may occur at a vertex in $V_\Diamond$ and the Mealy machine M_2^{NS} may follow a transition that is otherwise unreachable in $\mathcal{G}_{\mathsf{NS}}$. If an unreachable transition is taken, then we cannot guarantee that the output of the Mealy machine will result in a play that is winning for Player 2. For example, when the token moves from v_6 to v_7, the Mealy machine takes the unreachable $q_4 \xrightarrow{v_7} q_5$ transition, which as we saw above results in an outcome that is not winning for the $\overline{\mathsf{FWMP}(3)}$ objective.

Since we do not know how the Mealy machine M_2^{NS} behaves on taking unreachable transitions, we design the Mealy machine M_2^{AS} that defines the almost-sure winning strategy in $\mathcal{G}$ to reset instead of taking an unreachable transition. For instance, we define the transition in M_2^{AS} from state q_4 on reading vertex v_7 to be as if the game started from v_7. The Mealy machine M_2^{NS} on reading v_7 from the initial state, outputs v_8 and changes its state to q_4. Therefore, we want the same behaviour in M_2^{AS} from q_4, i.e., on reading vertex v_7 from v_4, the Mealy machine outputs v_8 and update its state to q_4. We add the necessary reset transitions in this manner for all states. For every state q, for every vertex v, if there is a reachable transition $q \xrightarrow{v} q'$ from q on reading v, then we retain the same transition in M_2^{AS}, i.e., with the same output and same next state. If there is no reachable transition from q on reading v, then we add a reset transition. We go to the same state and output the same vertex that would be output from q_0 on reading v. (For all vertices in $V_\Diamond$, we change the output of the Mealy machine to ϵ since Player 2 does not control these vertices in the stochastic game $\mathcal{G}$.) This gives us a complete Mealy machine.

Finally, after defining the new transitions, we see that there are no transitions that lead to q_5. It is an unreachable state and we delete it. Figure 9 shows some of the transitions in the Mealy machine M_2^{AS} obtained after the resetting. The figure excludes all unreachable transitions, and shows reset transitions out of q_4. There exist reset transitions out of states q_1, q_2 and q_3 as well, but we omit them in the figure for the sake of clarity. □

Now, we formally state a procedure to construct an almost-sure winning strategy in $\mathcal{G}$ from a given winning strategy in $\mathcal{G}_{\mathsf{NS}}$ and show the correctness of this procedure.

Construction 6.3 (Reset strategy)**.** Let σ_2^{NS} be a strategy of Player 2 that is winning for $\overline{\mathsf{FWMP}(\ell)}$ from every vertex in $\mathcal{G}_{\mathsf{NS}}$, and let $M_2^{\mathsf{NS}} = (Q_2^{\mathsf{NS}}, q_0, V, V \cup \{\epsilon\}, \Delta_2^{\mathsf{NS}}, \delta_2^{\mathsf{NS}})$ be a Mealy machine that defines the strategy σ_2^{NS}, where the set of states is Q_2^{NS}, the initial state is q_0, the input alphabet is V, the output alphabet is $V \cup \{\epsilon\}$, the transition function is $\Delta_2^{\mathsf{NS}} \colon Q_2^{\mathsf{NS}} \times V \to Q_2^{\mathsf{NS}}$, and the output function is $\delta_2^{\mathsf{NS}} \colon Q_2^{\mathsf{NS}} \times V \to V \cup \{\epsilon\}$. Since the strategy σ_2^{NS} is winning for $\overline{\mathsf{FWMP}(\ell)}$ in $\mathcal{G}_{\mathsf{NS}}$ irrespective of the initial vertex of the game, the transition and output functions are defined from the initial state of the Mealy machine M_2^{NS} for all vertices in the game, that is, $\Delta_2^{\mathsf{NS}}(q_0, v)$ and $\delta_2^{\mathsf{NS}}(q_0, v)$ are defined for all $v \in V$.

From this, we give a construction of a Mealy machine $M_2^{\mathsf{AS}} = (Q_2, q_0, V, V \cup \{\epsilon\}, \Delta_2, \delta_2)$ with $Q_2 \subseteq Q_2^{\mathsf{NS}}$ that defines the reset strategy σ_2^{AS} for Player 2 in the stochastic game $\mathcal{G}$. We begin with all the states of M_2^{NS} and then over the course of the construction, delete some states that are not needed. The initial state of M_2^{AS} is q_0, the same as the initial state of M_2^{NS}. The input and output alphabets of M_2^{AS} are the same as that of M_2^{NS}. It remains to define the transition function Δ_2^{NS} and the update function δ_2^{NS}.

We begin by computing all the transitions in M_2^{NS} that are reachable from the initial state q_0. For all states $q_1, q_2 \in Q_2^{\mathsf{NS}}$ and vertices $v \in V$, the transition $q_1 \xrightarrow{v} q_2$ is *reachable* in M_2^{NS} from q_0 if there exists a prefix $\rho \cdot v$ in $\mathcal{G}_{\mathsf{NS}}$ consistent with σ_2^{NS} such that $\hat{\Delta}_2^{\mathsf{NS}}(q_0, \rho) = q_1$ and $\Delta_2^{\mathsf{NS}}(q_1, v) = q_2$.

We have that for all $v \in V$, the transition $q_0 \xrightarrow{v} \Delta_2^{\mathsf{NS}}(q_0, v)$ is reachable. Moreover, for all transitions $q \xrightarrow{v} q'$ that are reachable, we have the following:

- if $v \in V_2$, then the transition from q' on input $\delta_2^{\mathsf{NS}}(q, v)$ is also reachable;
- if $v \in V_1$, then for all vertices $v' \in E(v)$, the transition from q on input v' is also reachable.

Since we do not know how M_2^{NS} behaves along unreachable transitions, we exclude unreachable transitions in M_2^{AS} and add reset transitions which we define now.

For all $q \in Q_2^{\mathsf{NS}}$ and all $v \in V$, we define the transition function Δ_2:

$$\Delta_2(q, v) = \begin{cases} \Delta_2^{\mathsf{NS}}(q, v) & \text{if there exists } q' \in Q_2^{\mathsf{NS}} \text{ such that } q \xrightarrow{v} q' \text{ is reachable,} \\ \Delta_2^{\mathsf{NS}}(q_0, v) & \text{otherwise.} \end{cases}$$

The Mealy machine on a reset transition behaves in the way it would on reading v if it were in the initial state q_0, that is, if the game began from v. This effectively resets the state of the Mealy machine. For all $q \in Q_2^{\mathsf{NS}}$ and all $v \in V$, we define the output function δ_2:

$$\delta_2(q, v) = \begin{cases} \epsilon & \text{if } v \in V_\Diamond \cup V_1, \\ \delta_2^{\mathsf{NS}}(q, v) & \text{if } v \in V_2 \text{ and there exists } q' \in Q_2^{\mathsf{NS}} \text{ such that } q \xrightarrow{v} q' \text{ is reachable,} \\ \delta_2^{\mathsf{NS}}(q_0, v) & \text{otherwise.} \end{cases}$$

If $v \in V_\Diamond \cup V_1$, then we have $\delta_2(q, v)$ equal to ϵ since σ_2^{AS} is a Player 2 strategy in the stochastic game $\mathcal{G}$ and is not defined for prefixes ending in vertices from $V_\Diamond \cup V_1$. Otherwise, if $v \in V_2$, then $\delta_2(q, v)$ is defined in a similar manner as $\Delta_2(q, v)$.

A state q' is unreachable if there does not exist $q \in Q_2^{\mathsf{NS}}$ and $v \in V$ such that the transition $q \xrightarrow{v} q'$ is reachable. We delete the unreachable states so we have $Q_2 \subseteq Q_2^{\mathsf{NS}}$. Since the unreachable states do not have incoming reachable transitions, this does not delete any transition that is reachable. Since the set of states in M_2^{AS} is a subset of the set of states in M_2^{NS}, the memory size of σ_2^{AS} is no greater than the memory size of σ_2^{NS}. This completes the construction of the Mealy machine M_2^{AS} defining the reset strategy σ_2^{AS}.

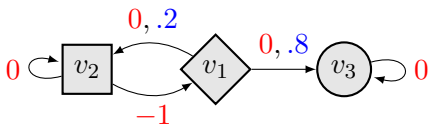
We now show that this strategy is almost-sure winning for Player 2 from all vertices. For all $q \in Q_2$ and $v \in V$, the output function $\Delta_2(q, v)$ and the transition function $\delta_2(q, v)$ are defined, and hence, M_2^{AS} is a complete Mealy machine. Moreover, from the way M_2^{AS} is constructed, we see that every transition in M_2^{AS} is either reachable in M_2^{NS} or is a reset transition.

Suppose Player 2 plays in the stochastic game $\mathcal{G}$ according to M_2^{AS} and this results in the prefix $\rho \cdot v$ for some $v \in V$. Let $\rho' \cdot v$ be the infix obtained from $\rho \cdot v$ by removing all vertices until the last occurrence of a reset transition in M_2^{AS}. In particular, if no reset transition occurs in M_2^{AS} on reading $\rho \cdot v$, then $\rho' \cdot v$ is equal to $\rho \cdot v$. From the definition of resetting, we have that starting from the initial state q_0 of M_2^{AS}, both $\rho' \cdot v$ and $\rho \cdot v$ take M_2^{AS} to the same state, i.e., $\hat{\Delta}_2(q_0, \rho' \cdot v) = \hat{\Delta}_2(q_0, \rho \cdot v)$.

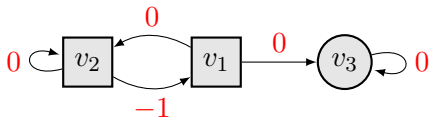
Since no reset transitions occur in M_2^{AS} on reading $\rho' \cdot v$, all transitions that occur are reachable if the Mealy machine M_2^{NS} is used for the stochastic game $\mathcal{G}$. Therefore, on reading $\rho' \cdot v$, the sequence of states visited in M_2^{NS} is the same as the sequence of states visited in M_2^{AS}. In particular, the state of M_2^{NS} on reading $\rho' \cdot v$ is the same as the state of M_2^{AS} on reading $\rho' \cdot v$, which is also the same as the state of M_2^{AS} on reading $\rho \cdot v$. Thus, we have that $\hat{\Delta}_2^{\mathsf{NS}}(q_0, \rho' \cdot v) = \hat{\Delta}_2(q_0, \rho \cdot v)$. Note that $\rho' \cdot v$ may contain deviations that M_2^{AS} does not reset on. For instance, in Example 6.2, if the token is on v_2, then with positive probability, it moves to v_5. This is a deviation as M_2^{NS} never moves the token from v_2 to v_5. However, in doing so, the Mealy machine M_2^{AS} does not follow any unreachable transition and does not reset. Note that both M_2^{AS} and M_2^{NS}, on reading the prefix $v_1 v_2 v_5$ with a deviation, reach the same state q_3.

Given $\rho' \cdot v$, there exists a finite path $\rho'' \cdot v$ of vertices without any deviations such that $\hat{\Delta}_2^{\mathsf{NS}}(q_0, \rho' \cdot v) = \hat{\Delta}_2^{\mathsf{NS}}(q_0, \rho'' \cdot v)$. This is because the transition from state $\hat{\Delta}_2^{\mathsf{NS}}(q_0, \rho')$ on input v is reachable in M_2^{NS}. Corresponding to the prefix $v_1 v_2 v_5$ in Example 6.2, we have that $v_1 v_3 v_5$ is a finite path without deviations that takes M_2^{AS} to the same state as $v_1 v_2 v_5$.

Thus, for every prefix $\rho \cdot v$ of an outcome of $\mathcal{G}$, there exists a finite path $\rho'' \cdot v$ without deviations such that $\hat{\Delta}_2(q_0, \rho \cdot v) = \hat{\Delta}_2^{\mathsf{NS}}(q_0, \rho'' \cdot v)$. As long as no deviations occur, the sequence of vertices seen after $\rho \cdot v$ is the same irrespective of whether Player 2 uses the strategy σ_2^{AS} or σ_2^{NS}. If a play in $\mathcal{G}$ continues for $\mathsf{M} \cdot |V| \cdot \ell$ steps without deviating, then by Lemma 4.16, it contains an open window of length ℓ. From any point in the play, the probability that σ_2^{AS} successfully copies σ_2^{NS} for i steps (that is, no deviations occur) is at least p^i, where p is the minimum probability over all the edges in $\mathcal{G}$. It follows that from every point in the play, the probability that an open window of length ℓ occurs in the next $\mathsf{M} \cdot |V| \cdot \ell$ steps is at least $p^{\mathsf{M} \cdot |V| \cdot \ell}$. Therefore, from every position in the play, the probability that an open window of length ℓ occurs eventually is at least $\sum_{i \geq 0} (1 - p^{\mathsf{M} \cdot |V| \cdot \ell})^i \cdot p^{\mathsf{M} \cdot |V| \cdot \ell} = 1$. Thus, with probability 1, infinitely many open windows of length ℓ occur in the outcome, and



(A) A stochastic game $\mathcal{G}$ with three vertices.



(B) The non-stochastic game corresponding to $\mathcal{G}$.

FIGURE 10. For all $\ell \geq 1$, Player 1 can positively satisfy $\mathsf{FWMP}(\ell)$ from every vertex in $\mathcal{G}$.

the outcome satisfies $\overline{\mathsf{FWMP}(\ell)}$. Thus, all vertices in $\mathcal{G}$ are almost-sure winning for Player 2 for $\overline{\mathsf{FWMP}(\ell)}$. This concludes the construction of a reset strategy that is almost-sure winning for Player 2 from all vertices in the stochastic game. □

We now construct a strategy σ_2^{Pos} of Player 2 that is positive winning from all vertices in $\langle\langle 2\rangle\rangle_{\mathcal{G}}^{\mathsf{Pos}}(\overline{\mathsf{FWMP}(\ell)})$. Let W_2^i and A_2^i denote the sets W_2 and A_2 computed in the i^{th} recursive call of $\mathsf{ASWin}_{\mathsf{FWMP}(\ell)}$ algorithm respectively. Here, $\mathsf{ASWin}_{\mathsf{FWMP}(\ell)}$ is the algorithm obtained by instantiating φ to $\mathsf{FWMP}(\ell)$ in Algorithm 5. If the token is in $\bigcup_i W_2^i$, then σ_2^{Pos} mimics σ_2^{AS}; if the token is in $\bigcup_i A_2^i \setminus W_2^i$, then σ_2^{Pos} is a positive-attractor strategy to W_2^i which is memoryless. Then, σ_2^{Pos} is a positive winning strategy for Player 2 from all vertices in $\langle\langle 2\rangle\rangle_{\mathcal{G}}^{\mathsf{Pos}}(\mathsf{FWMP}(\ell))$.

We have shown that for two-player stochastic games with $\mathsf{FWMP}(\ell)$ objective, the memory requirements of optimal strategies of both players is no greater than that for non-stochastic games with the same objective.

Remark 6.4. All plays consistent with the reset strategy of Player 2 described in Construction 6.3 are winning for Player 2. Thus, the reset strategy continues to be almost-sure winning even when Player 1 uses randomized strategies. Since the reset strategy is a deterministic strategy, we have that deterministic strategies suffice for Player 2 for the positive and almost-sure winning of the $\overline{\mathsf{FWMP}(\ell)}$ objective.

From [CDRR15], we have that the satisfaction problem for the $\mathsf{FWMP}(\ell)$ objective in non-stochastic games is in PTIME. Thus, from Theorem 5.3, Corollary 5.4, and Lemma 6.1, we have the following.

Theorem 6.5. *Given a stochastic game $\mathcal{G}$, a window length $\ell \geq 1$, and a threshold $p \in [0,1]$, for $\mathsf{FWMP}_{\mathcal{G}}(\ell)$, the positive and almost-sure satisfaction problems for Player 1 are in PTIME, and the quantitative satisfaction problem is in $\mathsf{NP} \cap \mathsf{coNP}$. Moreover for optimal strategies, memory of size ℓ is sufficient for Player 1 and memory of size $|V| \cdot \ell$ is sufficient for Player 2.*

Example 6.6. Consider the stochastic game $\mathcal{G}$ shown in Figure 10a, and objective $\mathsf{FWMP}(\ell)$ with window length $\ell = 2$. It is easy to see that all vertices are positively (even almost-surely) winning for Player 1 in $\mathcal{G}$. We compute the positive winning region as follows. First, consider the non-stochastic game $\mathcal{G}_{\mathsf{NS}}$ (Figure 10b). The winning region for Player 1 in $\mathcal{G}_{\mathsf{NS}}$ is $\{v_3\}$, and we thus have that the Player 1 positive attractor of $\{v_3\}$, which is $\{v_1, v_3\}$ is positively winning. The complement of the positive attractor induces the subgame with a single vertex v_2, that can be solved recursively to get that Player 1 positively (even almost-surely) wins from v_2. Therefore, we conclude that Player 1 is positive winning from every vertex. □

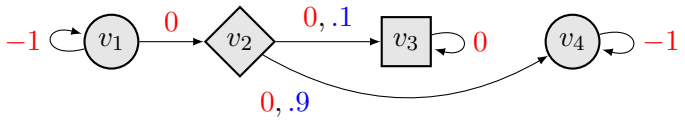


FIGURE 11. Player 1 almost surely wins in $\mathcal{G}$ for the objective $\mathsf{FWMP}(2)$ from $\{v_3\}$, while Player 2 positively wins from $\{v_1, v_2, v_4\}$.

Example 6.7. Consider the stochastic game $\mathcal{G}$ shown in Figure 11, and objective $\mathsf{FWMP}(\ell)$ with window length $\ell = 2$. We compute the almost-sure winning region for Player 1 by first computing the positive winning region for Player 2, which we do as follows. Using $\mathsf{PosWin}_{\mathsf{FWMP}(\ell)}$, the positive winning region for Player 1 in $\mathcal{G}$ is $\{v_1, v_2, v_3\}$. The complement of this set, $\{v_4\}$, is the almost-sure winning region for Player 2 in $\mathcal{G}$. The Player 2 positive attractor of $\{v_4\}$ is $\{v_2, v_4\}$, and we can conclude that this set is positively winning for Player 2. The complement of the positive attractor induces the subgame with vertices $\{v_1, v_3\}$, which can be solved recursively to get that Player 2 positively (even almost-surely) wins from $\{v_1\}$ but does not win even positively from $\{v_3\}$. Therefore, we conclude that Player 2 positively wins in the original stochastic game from $\{v_1, v_2, v_4\}$, and Player 1 almost surely wins from the complement $\{v_3\}$. □

6.2. **Bounded window mean-payoff objective.** We show that the SAS property holds for the objective $\mathsf{BWMP}_{\mathcal{G}}$ for all stochastic games $\mathcal{G}$.

Lemma 6.8. *For all stochastic games $\mathcal{G}$, the objective BWMP satisfies the SAS property.*

Proof. We need to show that for all stochastic games $\mathcal{G}$, if $\langle\!\langle 2 \rangle\!\rangle_{\mathcal{G}_{\mathsf{NS}}}(\overline{\mathsf{BWMP}}) = V$, then $\langle\!\langle 2 \rangle\!\rangle^{\mathsf{AS}}_{\mathcal{G}}(\overline{\mathsf{BWMP}}) = V$. Since every play that satisfies $\overline{\mathsf{BWMP}}$ also satisfies $\overline{\mathsf{FWMP}(\ell)}$ for all $\ell \geq 1$, we have that $\langle\!\langle 2 \rangle\!\rangle_{\mathcal{G}_{\mathsf{NS}}}(\overline{\mathsf{BWMP}}) = V$ implies $\langle\!\langle 2 \rangle\!\rangle_{\mathcal{G}_{\mathsf{NS}}}(\overline{\mathsf{FWMP}(\ell)}) = V$. It follows that for each $\ell \geq 1$, Player 2 has a finite-memory strategy (say, with memory M_ℓ), that is winning for the $\overline{\mathsf{FWMP}(\ell)}$ objective from all vertices in $\mathcal{G}_{\mathsf{NS}}$. For every such strategy, we construct a reset strategy σ_2^ℓ of memory size at most M_ℓ as described in the proof of Lemma 6.1 that is almost-sure winning for the $\overline{\mathsf{FWMP}(\ell)}$ objective from all vertices. We use these strategies to construct an infinite-memory strategy σ_2^{AS} of Player 2 that is almost-surely winning for $\overline{\mathsf{BWMP}}$ from all vertices in the stochastic game $\mathcal{G}$.

Let p be the minimum probability over all edges in the game, and for all $\ell \geq 1$, let $q(\ell)$ denote $p^{\mathsf{M}_\ell \cdot |V| \cdot \ell}$. We partition a play of the game into phases $1, 2, \ldots$ such that for all $\ell \geq 1$, the length of phase ℓ is equal to $\mathsf{M}_\ell \cdot |V| \cdot \ell \cdot \lceil 1/q(\ell) \rceil$. We define the strategy σ_2^{AS} as follows: if the game is in phase ℓ, then σ_2^{AS} is σ_2^ℓ, the reset strategy that is almost-sure winning for $\overline{\mathsf{FWMP}(\ell)}$ in $\mathcal{G}$.

We show that σ_2^{AS} is almost-sure winning for Player 2 for $\overline{\mathsf{BWMP}}$ in $\mathcal{G}$. Let E_ℓ denote the event that phase ℓ contains an open window of length ℓ. Given a play π, if E_ℓ occurs in π for infinitely many $\ell \geq 1$, then for every suffix of π and for all $\ell \geq 1$, the suffix contains an open window of length ℓ, and π satisfies $\overline{\mathsf{BWMP}}$. For all $\ell \geq 1$, we compute the probability that E_ℓ occurs in the outcome. For all $\ell \geq 1$, we can divide phase ℓ into $\lceil 1/q(\ell) \rceil$ blocks of length $\mathsf{M}_\ell \cdot |V| \cdot \ell$ each. If at least one of these blocks contains an open window of length ℓ, then the event E_ℓ occurs. It follows from the proof of Lemma 6.1 that if Player 2 follows σ_2^ℓ,

then the probability that there exists an open window of length ℓ in the next $\mathsf{M}_\ell \cdot |V| \cdot \ell$ steps is at least $q(\ell)$. Hence, the probability that none of the blocks in the phase contains an open window of length ℓ is at most $(1-q(\ell))^{\lceil 1/q(\ell)\rceil}$. Thus, the probability that E_ℓ occurs in phase ℓ is at least $1-(1-q(\ell))^{\lceil 1/q(\ell)\rceil} > 1-\frac{1}{e} \approx 0.63 > 0$. It follows that with probability 1, for infinitely many values of $\ell \geq 1$, the event E_ℓ occurs in π.

To show that E_ℓ occurs for infinitely many $\ell \geq 1$ in the outcome with probability 1,[2] we show an equivalent statement: the probability that E_ℓ occurs for only finitely many values of $\ell \geq 1$ in the outcome is 0. Let F be the set of all plays consistent with σ_2^{AS} in which only finitely many E_ℓ occur. We construct countably many subsets $F_0, F_1, \ldots$ of F as follows: let F_0 be the set of all plays in F in which E_k does not occur for all $k \geq 1$; and for all $j \geq 1$, let F_j consist of all plays in F in which E_j occurs, but for all $k > j$, the events E_k do not occur (and for $i < j$, the event E_i may or may not occur). Observe that $\bigcup_{j\geq 0} F_j = F$ and $F_i \cap F_j = \varnothing$ for all $i \neq j$.

For all $k \geq 1$, the probability that E_k does not occur is at most $(1-q(k))^{\lceil 1/q(k)\rceil}$ which is at most 0.37, irrespective of whether any other E_j's occur or not (again, this is because the probability that a block contains an open window of length ℓ is at least $q(\ell)$, independent of what happens in the rest of the play). For all $j \geq 0$, in the event F_j, for all $k > j$, we have that E_k does not occur. Since each E_k does not occur with probability at most 0.37, the probability of F_j is at most $\prod_{k>j}(0.37)$, which is 0. The event that finitely many E_ℓ's occur is the countable union of disjoint events $\bigcup_{j\geq 0} F_j$. Since the probability measure of each F_j is zero, and a countable sum of zero measure events has zero measure [Rud87], this implies that finitely many E_ℓ occur with probability zero. Thus, the probability that E_ℓ occurs for infinitely many $i \geq 1$ is 1.

Hence, the objective $\overline{\mathsf{BWMP}}$ is satisfied with probability 1 from all vertices in the stochastic game $\mathcal{G}$, and we have that $\langle\langle 2\rangle\rangle_{\mathcal{G}}^{\mathsf{AS}}(\overline{\mathsf{BWMP}}) = V$. □

Note that Lemma 2 in [CHH09b] is similar to Lemma 6.8 but refers to a different objective (finitary Streett instead of BWMP). However, the proofs have the following differences. In our proof, each phase of a play lasts for a fixed predetermined length and we show that for all $\ell \geq 1$, in phase ℓ, the probability that an open window of length ℓ occurs is at least 0.37, which is independent of ℓ. We use this to conclude that with probability 1, for infinitely many $\ell \geq 1$, phase ℓ contains an open window of length ℓ, and thus with probability 1, the play satisfies $\overline{\mathsf{BWMP}}$. In the proof in [CHH09b], a play continues to be in the ℓ^{th} phase until an open window of length ℓ appears. The ℓ^{th} phase does not end until an open window of length ℓ is observed in the phase. They show that for each phase in the play, the phase ends with probability 1, and thus with probability 1, the play contains open windows of length ℓ for all $\ell \geq 1$.

Note that solving a non-stochastic game with the BWMP objective is in $\mathsf{NP} \cap \mathsf{coNP}$ [CDRR15]. Thus by Corollary 5.4, quantitative satisfaction for BWMP is in $\mathsf{NP}^{\mathsf{NP}\cap\mathsf{coNP}} \cap \mathsf{coNP}^{\mathsf{NP}\cap\mathsf{coNP}}$. From [Sch83], we have that $\mathsf{NP}^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{NP}$ and $\mathsf{coNP}^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{coNP}$. To see this, suppose for an alphabet Σ, if $L \subseteq \Sigma^*$ is a language in $\mathsf{NP} \cap \mathsf{coNP}$, then for all $x \in \Sigma^*$, there either exists a short witness for x belonging to L or a short witness for x not belonging to L. A nondeterministic Turing machine can guess one of these witnesses

[2] The sum $\sum_{\ell\geq 1} \mathsf{Pr}(E_\ell)$ diverges to infinity. If we can show that the events E_ℓ are independent, then by the second Borel-Cantelli lemma [Dur10], this would directly imply that the probability of infinitely many of E_ℓ occurring is 1. However, we do not know if they are independent, so we are not able to apply the Borel-Cantelli lemma.

and verify in polynomial time whether $x \in L$ or $x \notin L$. Hence, an $\mathsf{NP} \cap \mathsf{coNP}$ oracle can be simulated by an NP machine, and we have $\mathsf{NP}^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{NP}$. For an oracle $\mathcal{A}$, a language L belongs to $\mathsf{coNP}^{\mathcal{A}}$ if and only if its complement $\overline{L}$ belongs to $\mathsf{NP}^{\mathcal{A}}$. Since the quantitative satisfaction problem belongs to $\mathsf{coNP}^{\mathcal{A}}$, its complement belongs to $\mathsf{NP}^{\mathcal{A}}$, which is NP for $\mathcal{A}$ in $\mathsf{NP} \cap \mathsf{coNP}$, and thus, the value problem belongs to coNP. Therefore, quantitative satisfaction of BWMP is in $\mathsf{NP} \cap \mathsf{coNP}$.

Moreover, from [CDRR15], Player 1 has a memoryless strategy and Player 2 needs infinite memory to play optimally in non-stochastic games with BWMP objective. From the proof of Lemma 6.8, by using the strategy σ_2^{AS}, Player 2 almost-surely wins $\overline{\mathsf{BWMP}}$ from all vertices in $\langle\!\langle 2 \rangle\!\rangle_{\mathcal{G}}^{\mathsf{AS}}(\overline{\mathsf{BWMP}})$. We can construct a positive winning strategy σ_2^{Pos} for Player 2 from all vertices in $\langle\!\langle 2 \rangle\!\rangle_{\mathcal{G}}^{\mathsf{Pos}}(\overline{\mathsf{BWMP}})$ in a similar manner as done for the positive winning strategy for $\overline{\mathsf{FWMP}(\ell)}$ in Section 6.1. Using similar reasoning as in Remark 6.4 in Section 6.1, it follows that deterministic strategies suffice for Player 2 for the positive and almost-sure satisfaction of the $\overline{\mathsf{BWMP}}$ objective. We summarize the results in the following theorem:

Theorem 6.9. *Given a stochastic game $\mathcal{G}$ and a threshold $p \in [0,1]$, for $\mathsf{BWMP}_{\mathcal{G}}$, the positive, almost-sure, and quantitative satisfaction for Player 1 are in $\mathsf{NP} \cap \mathsf{coNP}$. Moreover, a memoryless strategy suffices for Player 1, while Player 2 requires an infinite memory strategy to play optimally.*

Remark 6.10. Solving non-stochastic games with BWMP objective is at least as hard as solving traditional mean-payoff games [CDRR15]. Since non-stochastic games are a special case of stochastic games, it follows that solving the positive, almost-sure, and quantitative satisfaction problems for stochastic games with BWMP objective is at least as hard as solving traditional mean-payoff games.

Acknowledgment

We thank Mickael Randour for pointing out reference [BHR16b], making us aware of the bugs in the algorithms of [CDRR15] and the correct version of these algorithms as presented here.

References

[AH94] R. Alur and T. A. Henzinger. Finitary Fairness. In *LICS*, pages 52–61. IEEE Computer Society, 1994. doi:10.1109/LICS.1994.316087.

[BDOR20] T. Brihaye, F. Delgrange, Y. Oualhadj, and M. Randour. Life is Random, Time is Not: Markov Decision Processes with Window Objectives. *Logical Methods in Computer Science*, Volume 16, Issue 4, Dec 2020. doi:10.23638/LMCS-16(4:13)2020.

[BGR19] B. Bordais, S. Guha, and J-F. Raskin. Expected Window Mean-Payoff. In *FSTTCS*, volume 150 of *LIPIcs*, pages 32:1–32:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSTTCS.2019.32.

[BHR16a] V. Bruyère, Q. Hautem, and M. Randour. Window Parity Games: An Alternative Approach Toward Parity Games with Time Bounds. In *GandALF*, volume 226 of *EPTCS*, pages 135–148, 2016. doi:10.4204/EPTCS.226.10.

[BHR16b] V. Bruyère, Q. Hautem, and J-F. Raskin. On the Complexity of Heterogeneous Multidimensional Games. In *CONCUR*, volume 59 of *LIPIcs*, pages 11:1–11:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.11.

[Bil86] P. Billingsley. *Probability and Measure*. John Wiley and Sons, second edition, 1986.

[BK08] C. Baier and J-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[CDRR15] K. Chatterjee, L. Doyen, M. Randour, and J-F. Raskin. Looking at Mean-Payoff and Total-Payoff through Windows. *Information and Computation*, 242:25–52, 2015. doi:10.1016/j.ic.2015.03.010.

[CH08] K. Chatterjee and T. A. Henzinger. Value Iteration. In *25 Years of Model Checking - History, Achievements, Perspectives*, volume 5000 of *Lecture Notes in Computer Science*, pages 107–138. Springer, 2008. doi:10.1007/978-3-540-69850-0_7.

[CH12] K. Chatterjee and T. A. Henzinger. A Survey of Stochastic ω-Regular Games. *Journal of Computer and System Sciences*, 78(2):394–413, 2012. doi:10.1016/j.jcss.2011.05.002.

[Cha07] K. Chatterjee. Concurrent games with tail objectives. *Theoretical Computer Science*, 388(1):181–198, 2007. doi:10.1016/j.tcs.2007.07.047.

[CHH09a] K. Chatterjee, T. A. Henzinger, and F. Horn. Finitary Winning in ω-Regular Games. *ACM Transactions on Computational Logic*, 11(1):1:1–1:27, 2009. doi:10.1145/1614431.1614432.

[CHH09b] K. Chatterjee, T. A. Henzinger, and F. Horn. Stochastic Games with Finitary Objectives. In *MFCS*, pages 34–54. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-03816-7_4.

[Con92] A. Condon. The Complexity of Stochastic Games. *Information and Computation*, 96(2):203–224, 1992. doi:10.1016/0890-5401(92)90048-K.

[Dur10] R. Durrett. *Probability: Theory and Examples*. Cambridge University Press, 2010.

[EM79] A. Ehrenfeucht and J. Mycielski. Positional Strategies for Mean Payoff Games. *International Journal of Game Theory*, 8(2):109–113, 1979. doi:10.1007/BF01768705.

[FV96] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1996. doi:10.1007/978-1-4612-4054-9.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. doi:10.1007/3-540-36387-4.

[Hau18] Q. Hautem. *The Complexity of Combining Objectives in Two-Player Games*. PhD thesis, Université de Mons, 2018.

[Hor07] F. Horn. Faster Algorithms for Finitary Games. In *TACAS*, volume 4424 of *Lecture Notes in Computer Science*, pages 472–484. Springer, 2007. doi:10.1007/978-3-540-71209-1_36.

[HTWZ15] F. Horn, W. Thomas, N. Wallmeier, and M. Zimmermann. Optimal Strategy Synthesis for Request-Response Games. *RAIRO - Theoretical Informatics and Applications*, 49(3):179–203, 2015. doi:10.1051/ita/2015005.

[Jur98] M. Jurdzinski. Deciding the Winner in Parity Games is in UP $\cap$ co-UP. *Information Processing Letters*, 68(3):119–124, 1998. doi:10.1016/S0020-0190(98)00150-1.

[KPV09] O. Kupferman, N. Piterman, and M. Y. Vardi. From Liveness to Promptness. *Formal Methods in System Design*, 34(2):83–103, 2009. doi:10.1007/s10703-009-0067-z.

[Mar98] D. A. Martin. The Determinacy of Blackwell Games. *Journal of Symbolic Logic*, 63(4):1565–1581, 1998. doi:10.2307/2586667.

[Rud87] W. Rudin. *Real and Complex Analysis*. McGraw-Hill Book Company, 1987.

[Sch83] U. Schöning. A Low and a High Hierarchy within NP. *Journal of Computer and System Sciences*, 27(1):14–28, 1983. doi:10.1016/0022-0000(83)90027-2.

[WZ17] A. Weinert and M. Zimmermann. Easy to Win, Hard to Master: Optimal Strategies in Parity Games with Costs. *Logical Methods in Computer Science*, Volume 13, Issue 3, Sep 2017. doi:10.23638/LMCS-13(3:29)2017.

[ZP96] U. Zwick and M. Paterson. The Complexity of Mean Payoff Games on Graphs. *Theoretical Computer Science*, 158(1&2):343–359, 1996. doi:10.1016/0304-3975(95)00188-3.