Sure-Almost-Sure and Sure-Limit-Sure Window Mean Payoff in Markov Decision Processes

Pranshu Gaba
Tata Institute of Fundamental Research, Mumbai, India

Shibashis Guha
Tata Institute of Fundamental Research, Mumbai, India

Abstract

Given rationals α and β, the *sure-almost-sure* problem for a threshold Boolean objective φ in a Markov decision process (MDP) asks if one can simultaneously ensure that all outcomes of the MDP have φ-value at least α (i.e. *sure* α satisfaction), and with probability 1 the outcome has φ-value at least β (i.e. *almost-sure* β satisfaction). The *sure-limit-sure* problem asks if for all $\varepsilon > 0$, one can simultaneously ensure that all outcomes have φ-value at least α, and with probability at least $1 - \varepsilon$ the outcome has φ-value at least β. Moreover, if simultaneous satisfaction of objectives is possible, then one would also like to construct a strategy (for sure-almost-sure) or a family of strategies (for sure-limit-sure) that achieves this. Even if both sure satisfaction and almost-sure (resp., limit-sure) satisfaction for an objective are known, combining the two is often non-trivial and requires novel techniques and approaches.

In this paper, we solve the sure-almost-sure and sure-limit-sure problems for *window mean-payoff objectives*. While it is known that almost-sure satisfaction and limit-sure satisfaction for window mean-payoff coincide in MDPs, we show that sure-almost-sure satisfaction is distinct from sure-limit-sure satisfaction. The window mean-payoff objective strengthens the standard mean-payoff objective by requiring that eventually, from every point in the infinite run, the average payoff becomes greater than a given threshold within a finite window length. We study two variants of window mean payoff: in the *fixed* variant, the window length ℓ is given, while in the *bounded* variant, the length is not given but is required to be bounded throughout the run. We show that the sure-almost-sure problem and the sure-limit-sure problem are both in PTIME for the fixed variant (if ℓ is given in unary) and are both in NP $\cap$ coNP for the bounded variant, matching the computational complexity of sure satisfaction and almost-sure satisfaction when considered separately for these objectives. We also give bounds for the memory requirement of winning strategies for all considered problems.

2012 ACM Subject Classification Mathematics of computing → Stochastic processes

Keywords and phrases Beyond worst-case synthesis, sure-almost-sure satisfaction, window mean payoff, finitary objectives, Markov decision processes

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.36

Related Version *Extended Version*: `https://arxiv.org/abs/2605.12191` [20]

Funding We acknowledge support of CEFIPRA project no. 7302-I and of the Department of Atomic Energy, Government of India, project no. RTI4014.

Acknowledgements We thank anonymous reviewers for suggesting various improvements and pointing out a subtle bug in a proof.

1 Introduction

Beyond worst-case synthesis. Classical two-player zero-sum games [1, 21] involve decision-making against a purely antagonistic environment where a threshold performance or a specific behaviour of the system needs to be ensured against *all possible strategies of the environment*, that is even in the worst case. On the other hand, Markov decision processes (MDPs) [23] model uncertainty, and decision-making involves ensuring a *specified behaviour*

with sufficient probability or a *high expected performance* against a stochastic environment. Such a stochastic model of the environment, however, does not provide any guarantee on the worst-case performance, and a strategy that is adequate against an adversarial environment may provide a suboptimal performance against a stochastic environment.

In practice, both might be desired simultaneously: A system needs to provide a guarantee in the worst-case, as well as ensure some threshold performance with a high probability against a stochastic environment. The beyond worst-case (BWC) framework was introduced in [8] to provide strict worst-case guarantee as well as good expected performance for quantitative specifications. Subsequently, in [5], the beyond worst-case setting was studied for qualitative omega-regular objectives encoded as two parity objectives, where the first one needs to be satisfied surely, while the second one needs to be satisfied almost-surely or with a high threshold probability.

Window mean payoff. For objectives such as long-run limits of reward functions [18, 25], a play may satisfy the objective while still exhibiting undesired behaviours over arbitrarily long finite segments [10, 11]. Finitary (or window) objectives enforce that undesired behaviour persist only within intervals of bounded length (windows) along the play. In this work, we focus on *window mean-payoff objectives* [10], which are finitary versions of the classical mean-payoff objective. Given a window length $\ell \geq 1$ and a threshold $\lambda \in \mathbb{Q}$, the fixed window mean-payoff objective $\mathsf{FWMP}(\ell, \lambda)$ holds if, except for a finite prefix, from every position i in the play there exists a window starting at i of length at most ℓ whose mean payoff is at least λ. The bounded window mean-payoff objective $\mathsf{BWMP}(\lambda)$ holds for a play if there exists some $\ell \geq 1$ for which $\mathsf{FWMP}(\ell, \lambda)$ holds for the play.

Related work. The beyond worst-case framework under expectation and worst-case semantics was introduced in [8] for quantitative objectives. The problem where the players are restricted to finite-memory strategies was shown to be in $\mathsf{NP} \cap \mathsf{coNP}$ for the mean-payoff objective. The case of infinite memory strategy for the BWC synthesis problem was left open in [8] and was solved in [14]. Further, in [14], a natural relaxation of the BWC problem, the beyond almost-sure synthesis (BAS) problem was introduced that asks for a threshold performance almost surely, that is, with probability 1, while maximizing expectation and was shown to be in PTIME for the mean-payoff objective. The beyond probability threshold synthesis problem that asks for a threshold performance with a given probability p while maximizing expectation was studied for mean-payoff objective in [12]. In [5], the beyond worst-case synthesis problems were studied for qualitative omega-regular objectives encoded as parity objectives, and the problems were shown to be in $\mathsf{NP} \cap \mathsf{coNP}$; in [13, 17], these problems were studied in the context of stochastic games which are a generalization of MDPs where the environment is both stochastic and adversarial. In [3], Boolean combinations of omega-regular objectives that need to be enforced either surely, almost surely, existentially, or with non-zero probability were studied, and it was shown that both randomization and infinite memory may be required by an optimal strategy. In [4], a combination of parity objective and multiple reachability objectives along with threshold probabilities were considered where the parity objective needs to be satisfied surely and each reachability objective is to be satisfied with the corresponding threshold probability.

Mean-payoff objectives were studied initially in two-player games on graphs without stochasticity [18, 25], and finitary versions were introduced as window mean-payoff objectives [10] and have been studied extensively since the last decade. For window mean-payoff objectives, the satisfaction problem [7] and the expectation problem [6] were studied in MDPs.

Table 1 Previous and our results (shaded) for window mean-payoff satisfaction in MDPs. In the memory bounds, we denote by $|V|$ the number of vertices in the MDP, by ℓ the window length (given in unary), by $\mathbb{P}_{\min}$ the minimum non-zero probability on the edges in the MDP, and by $w_{\max}$ the maximum absolute payoff on the edges of the MDP.

Winning condition		Complexity	Memory bound	
			lower	upper
FWMP(ℓ)	sure [10]	PTIME	$\ell - 1$	ℓ
	almost-sure [7]	PTIME	$\ell - 1$	ℓ
	SAS (Theorem 10)	PTIME	$\Omega(\max\{\|V\|, \ell\})$	$\mathcal{O}(\|V\| \cdot \ell)$
	SLS (Theorem 15)	PTIME	$\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{\|V\|}} + \ell\right)$	$\mathcal{O}\left(\frac{\|V\| \cdot \log(1/\varepsilon)}{(\mathbb{P}_{\min})^{\|V\|}} + \ell\right)$
BWMP	sure [10]	NP ∩ coNP	memoryless	memoryless
	almost-sure [7]	NP ∩ coNP	memoryless	memoryless
	SAS (Theorem 13)	NP ∩ coNP	$\Omega(\|V\| \cdot w_{\max})$	$\mathcal{O}(\|V\|^3 \cdot \|w_{\max}\|)$
	SLS (Theorem 16)	NP ∩ coNP	$\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{\|V\|}}\right)$	$\mathcal{O}\left(\frac{\|V\| \cdot \log(1/\varepsilon)}{(\mathbb{P}_{\min})^{\|V\|}}\right)$

Both problems were shown to be in PTIME for the FWMP(ℓ) objective and in NP ∩ coNP for the BWMP objective. The satisfaction problem and the expectation problem for window mean-payoff objectives have recently been studied in [16] and [15] respectively for the more general setting that are stochastic games. In [19], BWC synthesis with the expectation semantics has been studied for window mean-payoff objectives in MDPs.

Contributions. We study two problems for the fixed and bounded window mean-payoff objectives each. Given an MDP $\mathcal{M}$, thresholds $\alpha, \beta \in \mathbb{Q}$, and an initial vertex v:

1. (Sure-almost-sure (SAS) synthesis) synthesize a strategy that ensures (i) satisfaction of the window mean-payoff objective with threshold α surely, i.e. against all strategies of an adversarial environment, and (ii) satisfaction of the window mean-payoff objective with threshold β almost surely, that is, with probability 1, against a stochastic model of the environment.
2. (Sure-limit-sure (SLS) synthesis) for every $\varepsilon > 0$, synthesize a strategy that ensures (i) satisfaction of the window mean-payoff objective with threshold α surely, and (ii) satisfaction of the window mean-payoff objective with threshold β with probability $1 - \varepsilon$.

While almost-sure satisfaction and limit-sure satisfaction are equivalent in MDPs [9], we show that sure-almost-sure satisfaction is strictly stronger than sure-limit-sure satisfaction (Example 2). We show that for both FWMP(ℓ) and BWMP, the SAS and SLS problems are no more complex than solving sure satisfaction [10] or almost-sure satisfaction [7] of the same objective separately. We show in Theorems 10 and 15 that the above problems are in PTIME for FWMP(ℓ) if ℓ is given in unary, and in Theorems 13 and 16 that the problems are in NP ∩ coNP for BWMP. Deterministic strategies suffice, and memory requirement may be higher for sure-almost-sure satisfaction compared to satisfying the objective either surely or almost surely (Examples 9 and 12). Our results are summarized in Table 1.

Organization. Section 2 defines technical preliminaries and Section 3 defines the window mean-payoff objectives. Sections 4 and 5 study the decision problems and strategies for sure-almost-sure and sure-limit-sure satisfaction of window mean-payoff objectives respectively. Omitted proofs are available in the extended version in [20].

2 Preliminaries

Markov decision processes. A (finite) *Markov decision process (MDP)* is a tuple $\mathcal{M} = ((V, E), (V_\bigcirc, V_\Diamond), \mathbb{P}, w)$, where:

- (V, E) is a directed graph with a (finite) 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 $N^+(v) = \{v' \in V \mid (v, v') \in E\}$ of out-neighbours of v is non-empty, i.e. no deadlocks.
- $(V_\bigcirc, V_\Diamond)$ is a partition of V. The vertices in $V_\bigcirc$ belong to the system or Player $\bigcirc$ (denoted $\mathcal{P}_\bigcirc$), and the vertices in $V_\Diamond$ belong to the environment and are called *probabilistic vertices*.
- $\mathbb{P}\colon V_\Diamond \to \mathcal{D}(V)$ is a *probability function* that describes the behaviour of probabilistic vertices in the game. It maps each probabilistic vertex $v \in V_\Diamond$ to a probability distribution $\mathbb{P}(v)$ over the set $N^+(v)$ of out-neighbours of v such that $\mathbb{P}(v)(v') > 0$ for all $v' \in N^+(v)$ (i.e., all out-neighbours have non-zero probability). We denote by $\mathbb{P}_{\min}$ the minimum non-zero probability in $\mathcal{M}$, that is, $\min\{\mathbb{P}(v)(v') \mid v, v' \in V,\ v' \in N^+(v)\}$.
- $w\colon E \to \mathbb{Q}$ is a *payoff function* assigning a rational payoff to every edge in the game. We denote by $w_{\max}$ the maximum absolute payoff on the edges of $\mathcal{M}$, that is, $\max_{e \in E}|w(e)|$.

A play in an MDP $\mathcal{M}$ begins by putting a token on a designated vertex v_0. The token proceeds in rounds as follows for all $i \geq 0$. At the beginning of round i, if the token is on a vertex v_i, and $v_i \in V_\bigcirc$, then $\mathcal{P}_\bigcirc$ chooses an out-neighbour $v_{i+1} \in N^+(v)$; otherwise $v_i \in V_\Diamond$, and an out-neighbour $v_{i+1} \in N^+(v)$ is chosen with probability $\mathbb{P}(v_i)(v_{i+1})$. Player $\mathcal{P}_\bigcirc$ receives the amount $w(v_i, v_{i+1})$ given by the payoff function and the token moves to the chosen v_{i+1} for the next round. This continues ad infinitum resulting in a *play* $\pi = v_0 v_1 v_2 \cdots \in V^\omega$ that is an infinite sequence of vertices such that $(v_i, v_{i+1}) \in E$ for all $i \geq 0$. 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 $\mathsf{Plays}_\mathcal{M}$ and $\mathsf{Prefs}_\mathcal{M}$ the set of all plays and the set of all finite prefixes in $\mathcal{M}$ respectively. We denote by $\mathsf{Last}(\rho)$ the last vertex of the prefix $\rho \in \mathsf{Prefs}_\mathcal{M}$. We denote by $\mathsf{Prefs}^i_\mathcal{M}$ $(i \in \{\bigcirc, \Diamond\})$ the set of all prefixes ρ such that $\mathsf{Last}(\rho) \in V_i$. Figure 1 shows an example of an MDP; vertices of $\mathcal{P}_\bigcirc$ are shown as circles and probabilistic vertices as diamonds. The payoff for each edge is shown in red and the probability distribution over out-neighbours of probabilistic vertices is shown in blue.

Objectives. An objective φ is a Borel-measurable subset of $\mathsf{Plays}_\mathcal{M}$ [22]. A play $\pi \in \mathsf{Plays}_\mathcal{M}$ *satisfies* an objective φ if $\pi \in \varphi$. For a target set $T \subseteq V$, we define some objectives:

- *reachability*, denoted $\mathsf{Reach}(T) = \{\pi = v_0 v_1 v_2 \cdots \mid \exists i \geq 0,\ v_i \in T\}$, is the set of all plays that visit a vertex in T; we also define a variant of $\mathsf{Reach}(T)$:
  - For a subset $F \subseteq E$ of edges, *edge-restricted reachability*, denoted $\mathsf{Reach}_F(T) = \{\pi = v_0 v_1 v_2 \cdots \mid \exists i \geq 0, \forall j \geq 0, v_i \in T \wedge (j < i \implies (v_j, v_{j+1}) \in F)\}$, is the set of all plays that visit a vertex in T and use only edges in F until the first visit to T;
- *safety*, denoted $\mathsf{Safe}(T) = \{\pi = v_0 v_1 v_2 \cdots \mid \forall i \geq 0,\ v_i \in T\}$, is the set of all plays that never visit a vertex not in T; and
- *Büchi*, denoted $\mathsf{Büchi}(T) = \{\pi = v_0 v_1 v_2 \cdots \mid \forall i \geq 0, \exists j \geq i,\ v_j \in T\}$, is the set of all plays that visit T infinitely often.

The objectives $\mathsf{Reach}(T)$ and $\mathsf{Safe}(V \setminus T)$ are duals of each other, that is, a play satisfies $\mathsf{Reach}(T)$ if and only if it does not satisfy $\mathsf{Safe}(V \setminus T)$.

For a rational threshold λ, we define some more objectives:

- *total payoff*, denoted $\mathsf{TP}(\lambda) = \{\pi = v_0 v_1 v_2 \cdots \mid \limsup_{n \to \infty} \sum_{i=0}^{n-1} w(v_i, v_{i+1}) \geq \lambda\}$, is the set of all plays with (supremum) total payoff at least λ; and

- *mean payoff*, denoted $\mathsf{MP}(\lambda) = \{\pi = v_0v_1v_2\cdots \mid \limsup_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} w(v_i, v_{i+1}) \geq \lambda\}$, is the set of all plays with (supremum) limit-average payoff at least λ.

In this work, we consider finitary versions of the mean-payoff objective called *window mean-payoff* objectives, which are defined in Section 3.

Prefix independence. 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 $\mathcal{M}$ (that is $\rho \cdot \pi \in \mathsf{Plays}_{\mathcal{M}}$), we have that $\rho \cdot \pi$ satisfies φ. An objective φ is *prefix-independent* if it is closed under both suffixes and prefixes. In other words, an objective φ is prefix-independent if it is independent of finite prefixes of the play, that is, it depends only on the infinite suffixes of a play. For a prefix-independent objective φ and for all plays π and π' with a common suffix (that is, π' can be obtained from π by removing and adding finite prefixes), we have that $\pi \in \varphi$ if and only if $\pi' \in \varphi$. The objectives $\mathsf{Büchi}(T)$ and $\mathsf{MP}(\lambda)$ are prefix-independent (closed under both suffixes and prefixes). In general, the objective $\mathsf{Safe}(T)$ is closed under suffixes but not prefixes, $\mathsf{Reach}(T)$ is closed under prefixes but not suffixes, and $\mathsf{Reach}_F(T)$ and $\mathsf{TP}(\lambda)$ are closed under neither suffixes nor prefixes.

Strategies. A (deterministic) *strategy*[1] for $\mathcal{P}_{\bigcirc}$ in an MDP $\mathcal{M}$ is a function $\sigma\colon \mathsf{Prefs}^{\bigcirc}_{\mathcal{M}} \to V$ that maps prefixes ρ with $\mathsf{Last}(\rho) \in V_{\bigcirc}$ to an out-neighbour of $\mathsf{Last}(\rho)$. Strategies can be realized as the output of a (possibly infinite-state) *Mealy machine*, that is a deterministic transition system with transitions labelled by input/output pairs. Intuitively, in each step, if the token is at a vertex v, then v is given as input to the Mealy machine. Then, the Mealy machine emits an output (an out-neighbour of v if $v \in V_{\bigcirc}$, an empty string if $v \in V_{\Diamond}$) and the state of the Mealy machine is also updated.

The *memory size* of a strategy σ is the smallest number of states a Mealy machine defining σ can have. A strategy σ is *memoryless* if for all prefixes $\rho, \rho' \in \mathsf{Prefs}^{\bigcirc}_{\mathcal{M}}$ such that $\mathsf{Last}(\rho) = \mathsf{Last}(\rho')$, we have $\sigma(\rho) = \sigma(\rho')$. Memoryless strategies can be defined by Mealy machines with only one state. A play $\pi = v_0v_1\cdots$ is an *outcome* of a strategy σ if for all $j \geq 0$ with $v_j \in V_{\bigcirc}$, we have $v_{j+1} = \sigma(\pi(0,j))$. Fixing a strategy σ and an initial vertex v in an MDP $\mathcal{M}$ yields a distribution $\mathsf{Pr}^{\sigma}_{\mathcal{M},v}(\cdot)$ over plays in $\mathcal{M}$ (see e.g. [2]). For an objective φ, we have that $\mathsf{Pr}^{\sigma}_{\mathcal{M},v}(\varphi)$ is the probability that an outcome of σ from v in $\mathcal{M}$ satisfies φ.

Winning conditions. A strategy σ of $\mathcal{P}_{\bigcirc}$ from v in $\mathcal{M}$ is *winning for positive-φ* if $\mathsf{Pr}^{\sigma}_{\mathcal{M},v}(\varphi) > 0$, *winning for probability(p)-φ* if $\mathsf{Pr}^{\sigma}_{\mathcal{M},v}(\varphi) \geq p$, *winning for almost-sure-φ* if $\mathsf{Pr}^{\sigma}_{\mathcal{M},v}(\varphi) = 1$, and *winning for sure-$\varphi$* if all outcomes of σ from v in $\mathcal{M}$ satisfy φ (that is, the edge probabilities are ignored and the environment is considered adversarial instead of probabilistic). If such a winning strategy σ from v exists, then the vertex v is said to be winning for positive-φ, probability(p)-φ, almost-sure-φ, or sure-φ respectively in $\mathcal{M}$. In addition, we also say that a vertex v is winning for *limit-sure-φ* in $\mathcal{M}$ if for all $\varepsilon > 0$, we have that v is winning for probability($1-\varepsilon$)-φ. Similar to winning vertices, an edge e is winning for positive-φ (resp., probability(p)-φ, limit-sure-φ, almost-sure-φ, sure-φ) if starting from e, there exists a strategy σ that is winning for positive-φ (resp., probability(p)-φ, limit-sure-φ, almost-sure-φ, sure-φ). The *winning region* for positive-φ in $\mathcal{M}$ is the set of all vertices in $\mathcal{M}$ that are

[1] Strategies can be randomized in general. However, randomization is not useful for the winning conditions considered in this paper, and thus we restrict our attention to deterministic strategies only.

winning for positive-φ. The winning regions for probability(p)-φ, limit-sure-φ, almost-sure-φ, and sure-φ are defined similarly. The winning region for sure-$\mathsf{Reach}(T)$ of a set $T \subseteq V$ is used often and thus has a special name, the *sure-attractor* of T, and is denoted by $\mathsf{SureAttr}(T)$.

Two-player games. When computing the winning region for sure-φ in an MDP $\mathcal{M} = ((V, E), (V_\bigcirc, V_\Diamond), \mathbb{P}, w)$, it is useful to view $\mathcal{M}$ as a *two-player game* played by $\mathcal{P}_\bigcirc$ against an adversary $\mathcal{P}_\Diamond$ by forgetting the probability function $\mathbb{P}$ and treating vertices in $v \in V_\Diamond$ as ones belonging $\mathcal{P}_\Diamond$ and the outgoing transitions of v as the choices of $\mathcal{P}_\Diamond$.

SubMDPs. Given an MDP $\mathcal{M} = ((V, E), (V_\bigcirc, V_\Diamond), \mathbb{P}, w)$, a subset $V' \subseteq V$ of vertices induces a *subMDP* of $\mathcal{M}$ if for all vertices $v' \in V'$, player $\mathcal{P}_\bigcirc$ has a strategy from v' to ensure that the token always remains in V' for all outcomes, that is, the set V' is a winning region for sure-$\mathsf{Safe}(V')$ in $\mathcal{M}$. In graph-theoretic terms, the subset V' induces a subMDP if (i) every vertex $v' \in V'$ has an out-neighbour in V', that is $N^+(v') \cap V' \neq \varnothing$, and (ii) every probabilistic vertex $v' \in V_\Diamond \cap V'$ has all its out-neighbours in V', that is $N^+(v') \subseteq V'$. The *induced subMDP* is denoted by $\mathcal{M}[V']$ and is obtained by restricting $\mathcal{M}$ to V'. Formally, $\mathcal{M}[V'] = ((V', E'), (V_\bigcirc \cap V', V_\Diamond \cap V'), \mathbb{P}', w')$, where $E' = E \cap (V' \times V')$, and $\mathbb{P}'$ and w' are restrictions of $\mathbb{P}$ and w to (V', E').

▶ **Proposition 1.** *If φ_{suf} is an objective that is closed under suffixes, then the winning regions for sure-φ_{suf} and for almost-sure-φ_{suf} in $\mathcal{M}$ both induce subMDPs of $\mathcal{M}$.*

SubMDP closure. Consider a subset $V' \subseteq V$ that satisfies Condition (i) but not necessarily Condition (ii), that is, every vertex $v' \in V'$ has an out-neighbour in V' but there may exist probabilistic vertices in V' with some out-neighbours not in V'. Such a subset V' induces a *subMDP closure* of $\mathcal{M}$, denoted by $\mathcal{M}\lceil V' \rceil$. The subMDP closure is itself an MDP and is obtained by first restricting $\mathcal{M}$ to V', and then for each probabilistic vertex $v' \in V'$, we scale up probabilities over the out-neighbours of v' to obtain a probability distribution over the out-neighbours of v' that belong to V'. Formally, $\mathcal{M}\lceil V' \rceil = ((V', E'), (V_\bigcirc \cap V', V_\Diamond \cap V'), \mathbb{P}', w')$, where $E' = E \cap (V' \times V')$, and $\mathbb{P}'$ is given by $\mathbb{P}'(v')(u') = \mathbb{P}(v')(u') / \sum_{x' \in N^+(v') \cap V'} \mathbb{P}(v')(x')$ for all $v' \in V' \cap V_\Diamond$, $u' \in N^+(v') \cap V'$, and w' is a restriction of w to (V', E'). It can be seen that the complement of a sure-attractor induces a subMDP closure.

3 Window mean-payoff objectives

In this section, we recall window mean-payoff objectives introduced in [10] in the context of two-player games. We study two types of window mean-payoff objectives: *fixed*, in which a window length $\ell \geq 1$ is given, and *bounded*, in which for every play we need a bound on the window length. First, we recall the notion of windows in a play.

Windows. We extend the definition of λ-*windows* as given in [10] for arbitrary thresholds $\lambda \in \mathbb{Q}$. Given a play $\pi = v_0 v_1 \cdots$, for all $i \geq 0$, a λ-window *starts* at position i of the play. The λ-window starting at position i *closes* at a position j if j is the first position after i such that the mean payoff of the infix $\pi(i, j)$ is at least λ. Equivalently, if for all $i < k \leq j$ it is the case that the mean payoff of $\pi(i, k)$ is less than λ, then the λ-window starting at i is *open* at j. Given $j > 0$, we say a λ-window is open at j if there exists an open λ-window $\pi(i, j)$ starting at i for some $i < j$. When there is no confusion, we also informally say that the λ-window opens at vertex v_i and closes at vertex v_j to mean that the window opens at position i and closes at position j. An interesting and useful observation is the *inductive property of λ-windows*: if the λ-window starting at position i closes at position j, then for all $i < k < j$, the λ-window starting at k is closed at or before j.

Fixed window length. Let $\ell \geq 1$ be a given window length and $\lambda \in \mathbb{Q}$ be a given threshold. A play $\pi = v_0 v_1 \cdots$ in $\mathcal{M}$ satisfies the *good window* objective $\mathsf{GW}_{\mathcal{M}}(\ell, \lambda)$ if the λ-window that opens at v_0 in π closes in at most ℓ steps.

$$\mathsf{GW}_{\mathcal{M}}(\ell, \lambda) = \{\pi \in \mathsf{Plays}_{\mathcal{M}} \mid \exists j \in \{1, \ldots, \ell\},\ \frac{1}{j} \sum_{k=0}^{j-1} w(v_k, v_{k+1}) \geq \lambda\}$$

A play π in $\mathcal{M}$ satisfies the *direct fixed window mean-payoff* objective $\mathsf{DirFWMP}_{\mathcal{M}}(\ell, \lambda)$ if every λ-window in π closes in at most ℓ steps.

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

The prefix-independent version of this objective is the *fixed window mean-payoff* objective $\mathsf{FWMP}_{\mathcal{M}}(\ell, \lambda)$. A play π satisfies $\mathsf{FWMP}_{\mathcal{M}}(\ell, \lambda)$ if there exists a suffix of π that satisfies $\mathsf{DirFWMP}_{\mathcal{M}}(\ell, \lambda)$. In other words, the play π satisfies $\mathsf{FWMP}_{\mathcal{M}}(\ell, \lambda)$ if from some point on in the play, all λ-windows close in at most ℓ steps.

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

We omit the subscript $\mathcal{M}$ when it is clear from the context. Note that $\mathsf{FWMP}(\ell, \lambda) \subseteq \mathsf{FWMP}(\ell', \lambda)$ for $\ell \leq \ell'$ as a smaller window length is a stronger constraint. We also have that $\mathsf{FWMP}(\ell, \lambda) \supseteq \mathsf{FWMP}(\ell, \lambda')$ if $\lambda \leq \lambda'$, since closing of λ'-windows implies closing of λ-windows.

Bounded window length. We also consider $\mathsf{DirBWMP}(\lambda)$ (*direct bounded window mean-payoff* objective) and $\mathsf{BWMP}(\lambda)$ (*bounded window mean-payoff* objective) where one requires that there exist a window length $\ell \geq 1$ such that the play satisfies $\mathsf{DirFWMP}(\ell, \lambda)$ or $\mathsf{FWMP}(\ell, \lambda)$ respectively.

Decision and synthesis problems. We consider the following problems that are a combination of sure and almost-sure (and limit-sure) satisfaction of $\mathsf{FWMP}(\ell)$ objectives.

1. *sure-almost-sure* satisfaction: Given an MDP $\mathcal{M}$, a vertex v, a sure threshold α and an almost-sure threshold β, does there exist a strategy σ of $\mathcal{P}_{\bigcirc}$ such that (i) all outcomes of σ from v satisfy $\mathsf{FWMP}(\ell, \alpha)$, and (ii) with probability 1, the outcome of σ from v satisfies $\mathsf{FWMP}(\ell, \beta)$ from v (i.e., $\Pr^{\sigma}_{\mathcal{M},v}(\mathsf{FWMP}(\ell, \beta)) = 1$)? If the answer is yes, then we would also like to synthesize such a strategy.
2. *sure-limit-sure* satisfaction: Given an MDP $\mathcal{M}$, a vertex v, a sure threshold α and a limit-sure threshold β, for all $0 < \varepsilon < 1$, does there exist a strategy σ_ε of $\mathcal{P}_{\bigcirc}$ such that (i) all outcomes of σ_ε from v satisfy $\mathsf{FWMP}(\ell, \alpha)$, and (ii) with probability at least $1 - \varepsilon$, the outcome of σ_ε from v satisfies $\mathsf{FWMP}(\ell, \beta)$ from v (i.e., $\Pr^{\sigma_\varepsilon}_{\mathcal{M},v}(\mathsf{FWMP}(\ell, \beta)) \geq 1 - \varepsilon$)? If the answer is yes, then we would also like to synthesize such a family of strategies.

We assume without loss of generality that $\alpha < \beta$, because otherwise the problems reduce to checking satisfaction of sure-$\mathsf{FWMP}(\ell, \alpha)$. We also study analogous problems for BWMP.

▶ **Example 2.** While almost-sure satisfaction and limit-sure satisfaction of $\mathsf{FWMP}(\ell, \alpha)$ are the same in MDPs [9], we illustrate using the MDP in Figure 1 with values $\ell = 2$, $\alpha = 1$, and $\beta = 5$ that sure-almost-sure satisfaction is strictly stronger than sure-limit-sure satisfaction for $\mathsf{FWMP}(\ell)$. We have an MDP where the vertex v_1 is winning for sure-$\mathsf{FWMP}(2, 1)$-limit-sure-$\mathsf{FWMP}(2, 5)$ but not for sure-$\mathsf{FWMP}(2, 1)$-almost-sure-$\mathsf{FWMP}(2, 5)$.

To see that sure-$\mathsf{FWMP}(2, 1)$ and almost-sure-$\mathsf{FWMP}(2, 5)$ cannot be satisfied simultaneously from v_1: in order to satisfy almost-sure-$\mathsf{FWMP}(2, 5)$, the token must reach v_3 with probability 1, and that requires that as long as the token has not reached v_3, the token

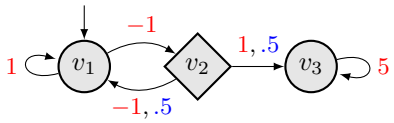

Figure 1 All vertices are winning for sure-FWMP$(2,1)$, almost-sure-FWMP$(2,5)$, and sure-FWMP$(2,1)$-limit-sure-FWMP$(2,5)$. However, vertices v_1 and v_2 are not winning for sure-FWMP$(2,1)$-almost-sure-FWMP$(2,5)$.

must be forwarded to v_2 from v_1 repeatedly. However, consider the outcome of this strategy where the token always moves from v_2 to v_1. In this outcome, the sequence $v_1v_2v_1$ appears infinitely often, and each such occurrence $v_1v_2v_1$ is an open 1-window of length 2. Thus, this outcome does not satisfy FWMP$(2,1)$, and we have that sure-FWMP$(2,1)$ is violated.

On the other hand, the vertex v_1 is winning for sure-limit-sure in the following way: given $0 < \varepsilon < 1$, visit v_2 a large integer N number of times such that the probability of reaching v_3 is greater than $1-\varepsilon$. If the vertex v_2 has been visited N times without reaching v_3, then loop on v_1 for the rest of the play. This strategy simultaneously satisfies both sure-FWMP$(2,1)$ and limit-sure-FWMP$(2,5)$. ⌟

▶ **Example 3.** The winning region for sure-FWMP(ℓ,α)-almost-sure-FWMP(ℓ,β) is not merely the intersection or the composition of the winning regions for sure-FWMP(ℓ,α) and for almost-sure-FWMP(ℓ,β): in the MDP in Figure 1, we have that every vertex is winning for sure-FWMP$(2,1)$ and for almost-sure-FWMP$(2,5)$ separately, and yet there exist vertices v_1 and v_2 that are not winning for sure-FWMP$(2,1)$-almost-sure-FWMP$(2,5)$. Indeed, we need to make good use of prefix-independence and the finitary nature of window mean-payoff objectives to find the winning region for the combination of winning conditions. ⌟

4 Sure-almost-sure satisfaction

4.1 Sure-almost-sure satisfaction of FWMP

In this section, we present Algorithm 1 (`SureAlmostSureFWMP`) that returns the set of all vertices v from which $\mathcal{P}_{\bigcirc}$ can satisfy sure-FWMP(ℓ,α)-almost-sure-FWMP(ℓ,β). We note that for sure-almost-sure satisfaction of window mean-payoff objectives, the exact probabilities out of probabilistic vertices do not matter, and thus we may omit them in figures.

Algorithm 1: Sure-almost-sure FWMP

Description of Algorithm 1. This algorithm computes and returns a set W (Line 10) of vertices that are winning for sure-FWMP(ℓ,α)-almost-sure-FWMP(ℓ,β) in $\mathcal{M}$.

We begin in Line 1 by computing the set $W_{\mathsf{S}\alpha}$ that is the winning region for $\mathcal{P}_{\bigcirc}$ for sure-FWMP(ℓ,α) in $\mathcal{M}$ by viewing the probabilistic vertices of $\mathcal{M}$ as adversarial, and then using [10, Algorithm 1] that finds the winning region for $\mathcal{P}_{\bigcirc}$ for FWMP(ℓ,α) in two-player games. If $\mathcal{P}_{\bigcirc}$ cannot satisfy sure-FWMP(ℓ,α) from a vertex v in $\mathcal{M}$, then $\mathcal{P}_{\bigcirc}$ also cannot satisfy sure-FWMP(ℓ,α)-almost-sure-FWMP(ℓ,β) from v in $\mathcal{M}$, and we do not include in W any vertex belonging to $V \setminus W_{\mathsf{S}\alpha}$. Moreover, since FWMP$(\ell,\alpha)$ is prefix-independent (and thus also closed under suffixes), it follows from Proposition 1 that $W_{\mathsf{S}\alpha}$ induces a subMDP $\mathcal{M}[W_{\mathsf{S}\alpha}]$ in which $\mathcal{P}_{\bigcirc}$ has a winning strategy for sure-FWMP(ℓ,α) from every vertex, and we subsequently only work with this subMDP. In Line 2, we compute the set $W_{\mathsf{S}\beta}$ that is the winning region for sure-FWMP(ℓ,β) in $\mathcal{M}[W_{\mathsf{S}\alpha}]$, and in Line 3, we initialize W to $W_{\mathsf{S}\beta}$.

Algorithm 1 `SureAlmostSureFWMP`$(\mathcal{M}, \ell, \alpha, \beta)$.

In: An MDP $\mathcal{M} = ((V, E), (V_\bigcirc, V_\Diamond), \mathbb{P}, w)$, window length ℓ, sure threshold α, and almost-sure threshold β

Out: The winning region for sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) in $\mathcal{M}$

1: $W_{\mathsf{S}\alpha} \leftarrow$ `SureFWMP`$(\mathcal{M}, \ell, \alpha)$
2: $W_{\mathsf{S}\beta} \leftarrow$ `SureFWMP`$(\mathcal{M}[W_{\mathsf{S}\alpha}], \ell, \beta)$
3: $W \leftarrow W_{\mathsf{S}\beta}$
4: **repeat**
5: $\quad P \leftarrow \mathsf{X}^{\leq 1}_{>0}(\mathcal{M}[W_{\mathsf{S}\alpha}], W)$
6: $\quad W_{\mathsf{sdab}} \leftarrow$ `SureDirFWMPASBüchi`$(\mathcal{M}\lceil W_{\mathsf{S}\alpha} \setminus W\rceil, \ell, \alpha, P)$
7: $\quad W \leftarrow W \cup W_{\mathsf{sdab}}$
8: $\quad W \leftarrow$ `SureAttr`$(\mathcal{M}[W_{\mathsf{S}\alpha}], W)$
9: **until** $W_{\mathsf{sdab}} = \varnothing$
10: **return** W

We then enter a **repeat** block (Lines 5 to 8) in which some vertices may be added to W in the following way.

- In Line 5, we find the set P of probabilistic vertices in $\mathcal{M}[W_{\mathsf{S}\alpha}]$ from which the token reaches W in one step with positive probability but not surely. In terms of the underlying graph, P is the set of all vertices in $(W_{\mathsf{S}\alpha} \cap V_\Diamond) \setminus W$ with at least one out-neighbour in W and at least one out-neighbour not in W. We denote this by $\mathsf{X}^{\leq 1}_{>0}$ in the algorithm.
- We note that every vertex v in $W_{\mathsf{S}\alpha} \setminus W$ has at least one out-neighbour in $W_{\mathsf{S}\alpha} \setminus W$ because otherwise $\mathcal{P}_\bigcirc$ could ensure that the token reaches W surely from v, in which case v should be included in W. Thus, the set $W_{\mathsf{S}\alpha} \setminus W$ induces a subMDP closure of $\mathcal{M}[W_{\mathsf{S}\alpha}]$, and we recall that we denote this by $\mathcal{M}\lceil W_{\mathsf{S}\alpha} \setminus W\rceil$. Suppose there exists a vertex v in $\mathcal{M}\lceil W_{\mathsf{S}\alpha} \setminus W\rceil$ from which $\mathcal{P}_\bigcirc$ can simultaneously ensure with probability 1 that P is visited infinitely often (that is, almost-sure-Büchi(P)) and surely that all α-windows in the outcome always close in at most ℓ steps (that is, sure-DirFWMP(ℓ, α)). Then starting from v in $\mathcal{M}[W_{\mathsf{S}\alpha}]$, we have that the token almost-surely eventually reaches W (from where sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) is satisfied), and even if the token does not reach W (which happens with probability 0), the outcome still surely satisfies DirFWMP(ℓ, α) (and thus also FWMP(ℓ, α)). Thus, in Line 6, we call Algorithm 2 (`SureDirFWMPASBüchi`) to compute the set W_{sdab} of vertices that is the winning region for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(P) in $\mathcal{M}\lceil W_{\mathsf{S}\alpha} \setminus W\rceil$, and in Line 7, we add to W all vertices in W_{sdab}. (The subscript sdab is short for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(P).)
- Finally, we compute SureAttr(W), the sure-attractor of W, that is the set of all vertices in $W_{\mathsf{S}\alpha}$ from which $\mathcal{P}_\bigcirc$ can surely reach W. We note that if $\mathcal{P}_\bigcirc$ can satisfy sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) from all vertices in W, then $\mathcal{P}_\bigcirc$ can also satisfy the same from all vertices in SureAttr(W). This holds because FWMP(ℓ, α) and FWMP(ℓ, β) are prefix-independent objectives; starting from a vertex v in SureAttr$(W) \setminus W$, player $\mathcal{P}_\bigcirc$ can ensure that every outcome from v reaches a vertex in W, following which $\mathcal{P}_\bigcirc$ can forget the entire history before reaching W (due to prefix-independence) and switch to a strategy that is winning for sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) from this vertex in W. Thus, we update W to include all vertices in SureAttr(W) (Line 8).

We repeat Lines 5 to 8 until we reach a fixed point for W, that is when the set W_{sdab} is empty (Line 9) and thus no vertices are added to W. At this point, we exit the **repeat** block and return W (Line 10).

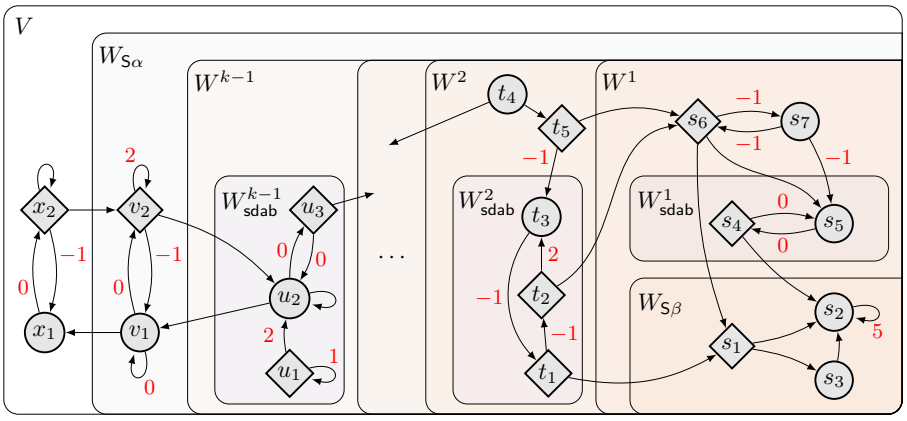

Figure 2 An illustration of various features of Algorithm 1 to compute the winning region for sure-FWMP($\ell = 3, \alpha = 0$)-almost-sure-FWMP($\ell = 3, \beta = 5$). Only relevant edge payoffs are shown.

We define some notation for the sets computed by Algorithm 1. Suppose the **repeat** block runs k times for some integer $k \geq 1$. For $1 \leq i \leq k$, let P^i, W^i_{sdab}, and W^i denote the sets P, W_{sdab}, and W computed in the i^{th} iteration of the **repeat** block. We also denote the set $W_{\mathsf{S}\beta}$ by W^0 for convenience. We note some relations between the various sets.

- For $1 \leq i \leq k$, the sets W^{i-1} and W^i_{sdab} are disjoint, since W^i_{sdab} is the winning region for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(P^i) in $\mathcal{M}\lceil W_{\mathsf{S}\alpha} \setminus W^{i-1} \rceil$.
- For $1 \leq i \leq k$, if W^i_{sdab} is non-empty, then P^i and W^i_{sdab} have a non-empty intersection since a vertex in P^i that is visited infinitely often to satisfy sure-DirFWMP(ℓ, α)-almost-sure-Büchi(P^i) from a vertex in W^i_{sdab} also belongs to W^i_{sdab}.
- For $1 \leq i \leq k$, we have that $W^{i-1} \cup W^i_{\mathsf{sdab}} \subseteq W^i$ since W^i is obtained by computing the sure-attractor of $W^{i-1} \cup W^i_{\mathsf{sdab}}$. Let us denote by A^i the set $W^i \setminus (W^{i-1} \cup W^i_{\mathsf{sdab}})$, that is the set of all vertices in W^i but not in $W^{i-1} \cup W^i_{\mathsf{sdab}}$ from which $\mathcal{P}_{\bigcirc}$ has a sure-attractor strategy to reach $W^{i-1} \cup W^i_{\mathsf{sdab}}$. Thus, we have for all $1 \leq i \leq k$ that W^{i-1}, W^i_{sdab}, and A^i form a partition of W^i. Moreover, the entire set W can be divided into pairwise disjoint sets W^0, W^1_{sdab}, A^1, W^2_{sdab}, A^2, ..., W^k_{sdab}, and A^k.
- Since the algorithm terminates in the k^{th} iteration, we have that $W^k_{\mathsf{sdab}} = \varnothing$ and thus also $A^k = \varnothing$. We get that $W^k = W^{k-1}$.
- For $1 \leq i < j \leq k$, the sets P^i and P^j are (in general) incomparable, that is, there may exist vertices u, v such that $u \in P^i$ and $u \notin P^j$, and $v \notin P^i$ and $v \in P^j$. We may have $u \in P^i$ and $u \notin P^j$ if u is a vertex that has out-neighbours in W^{i-1} and in $W^{j-1} \setminus W^{i-1}$ and nowhere else. We may have $v \notin P^i$ and $v \in P^j$ if v is a vertex that has out-neighbours in $W^{j-1} \setminus W^{i-1}$ and in $V \setminus W^{j-1}$ and nowhere else.

▶ **Example 4.** We show some features of Algorithm 1 in Figure 2 for window length $\ell = 3$ and for thresholds $\alpha = 0$ and $\beta = 5$. Vertices $\{x_1, x_2\}$ are not winning for sure-FWMP(ℓ, α) and are removed. The set $W_{\mathsf{S}\beta}$ (also referred to as W^0) consists of $\{s_1, s_2, s_3\}$ since from all three vertices the token reaches s_2 surely where all β-windows close in at most ℓ steps. Next, we compute the set P^1 to be $\{s_4, s_6, t_1\}$, the set of all probabilistic vertices with at least one out-edge in W^0 and at least one out-edge not in W^0. In the subMDP closure induced by $W_{\mathsf{S}\alpha} \setminus W^0$, we compute W^1_{sdab}, the winning region for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(P^1), to be $\{s_4, s_5\}$. The set A^1 of all vertices in $W_{\mathsf{S}\alpha} \setminus (W^1_{\mathsf{sdab}} \cup W^0)$ from which $\mathcal{P}_{\bigcirc}$ can ensure that the token reaches $W^1_{\mathsf{sdab}} \cup W^0$ surely is $\{s_6, s_7\}$. We come to the end of the first iteration of the **repeat** loop and we have that $W^1 = A^1 \cup W^1_{\mathsf{sdab}} \cup W^0 = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$. Since W^1_{sdab}

Algorithm 2 `SureDirFWMPASBüchi`$(\mathcal{M}, \ell, \alpha, T)$.

In: An MDP $\mathcal{M} = ((V, E), (V_\bigcirc, V_\Diamond), \mathbb{P}, w)$, window length ℓ, threshold α, and target set T
Out: The winning region for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T) in $\mathcal{M}$

1: $W_{\mathsf{sdpr}} \leftarrow$ `SureDirFWMPPosReach`$(\mathcal{M}, \ell, \alpha, T)$
2: **if** $V = W_{\mathsf{sdpr}}$ **then**
3: **return** V
4: **else**
5: $S \leftarrow$ `SureSafe`$(\mathcal{M}, W_{\mathsf{sdpr}})$
6: **return** `SureDirFWMPASBüchi`$(\mathcal{M}[S], \ell, \alpha, T \cap S)$

is non-empty, we have another iteration of the **repeat** loop. We compute $P^2 = \{t_1, t_2, t_5\}$ for W^1. The winning region for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(P^2) in $\mathcal{M}\lceil W_{\mathsf{S}\alpha} \setminus W^1 \rceil$ is $W^2_{\mathsf{sdab}} = \{t_1, t_2, t_3\}$, and the set A^2 is $\{t_4, t_5\}$. We get that $W^2 = \{t_1, t_2, t_3, t_4, t_5\} \cup W^1$. We repeat this until we reach W^{k-1}. Finally, we have that $P^k = \{v_2\}$ and $W^k_{\mathsf{sdab}} = \varnothing$. Thus, the algorithm terminates here and returns W^k. ⌟

We have given intuition on why vertices in W^k are winning for sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) in $\mathcal{M}$. For the converse, since vertices in $V \setminus W_{\mathsf{S}\alpha}$ are not winning for sure-FWMP(ℓ, α), they are also not winning for sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β). For the remaining set $W_{\mathsf{S}\alpha} \setminus W^k$ of vertices, we show in [20] that starting from $W_{\mathsf{S}\alpha} \setminus W^k$, player $\mathcal{P}_\bigcirc$ cannot satisfy sure-FWMP(ℓ, α)-almost-sure-Reach(W^k) and thus, if $\mathcal{P}_\bigcirc$ wants to satisfy sure-FWMP(ℓ, α), then the token remains in $W_{\mathsf{S}\alpha} \setminus W^k$ with some positive probability. Then, we show that the winning region for sure-FWMP(ℓ, β) in $\mathcal{M}\lceil W_{\mathsf{S}\alpha} \setminus W^k \rceil$ is empty, and the winning region for sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) in $\mathcal{M}\lceil W_{\mathsf{S}\alpha} \setminus W^k \rceil$ is also empty. This implies that if the token remains in $W_{\mathsf{S}\alpha} \setminus W^k$, then sure-FWMP$(\ell, \alpha)$-almost-sure-FWMP$(\ell, \beta)$ cannot be satisfied. Theorem 5 states the correctness of Algorithm 1.

▶ **Theorem 5.** *Algorithm 1 computes the winning region for sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) in $\mathcal{M}$.*

Algorithm 2: Sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T)

Recall that Algorithm 1 uses Algorithm 2 (`SureDirFWMPASBüchi`) as a subroutine, and we describe this algorithm now. Algorithm 2 returns the set of all vertices in an MDP that are winning for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T) for a target set $T \subseteq V$. We note that it may be the case that every vertex in an MDP is winning for sure-DirFWMP(ℓ, α) and almost-sure-Büchi(T) separately, and yet no vertex in the MDP is winning for the combination sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T) (see Figure 7 in Appendix A).

Description of Algorithm 2. We begin by calling Algorithm 3 (`SureDirFWMPPosReach`) in Line 1 to compute W_{sdpr}, the winning region for sure-DirFWMP(ℓ, α)-positive-Reach(T) in $\mathcal{M}$. (The subscript sdpr is short for sure-DirFWMP(ℓ, α)-positive-Reach(T).) If every vertex in $\mathcal{M}$ belongs to W_{sdpr}, then we go to Line 3 and return V, that is all vertices in $\mathcal{M}$. Otherwise, we have that W_{sdpr} is a strict subset of V and we go to Line 5 to compute S, the winning region for sure-Safe(W_{sdpr}), that is the set of all vertices in $\mathcal{M}$ from which $\mathcal{P}_\bigcirc$ can ensure for all outcomes that the token visits only vertices in W_{sdpr}. The set S induces a subMDP of $\mathcal{M}$, and in Line 6 we recursively call Algorithm 2 (`SureDirFWMPASBüchi`) on the subMDP $\mathcal{M}[S]$.

▶ **Theorem 6.** *Algorithm 2 computes the winning region for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T).*

Algorithm 3 `SureDirFWMPPosReach`$(\mathcal{M}, \ell, \alpha, T)$.

In: An MDP $\mathcal{M} = ((V, E), (V_\bigcirc, V_\Diamond), \mathbb{P}, w)$, window length ℓ, threshold α, and target set T
Out: The winning region for sure-DirFWMP(ℓ, α)-positive-Reach(T) in $\mathcal{M}$

1: $W_{\mathsf{SD}\alpha} \leftarrow$ `SureDirFWMP`$(\mathcal{M}, \ell, \alpha)$
2: $\mathcal{M} \leftarrow \mathcal{M}[W_{\mathsf{SD}\alpha}]$, $T \leftarrow T \cap W_{\mathsf{SD}\alpha}$
3: $R \leftarrow T$, $E_g, E_b \leftarrow \varnothing$
4: **while** $\{e = (s,t) \in E \mid s \in V \setminus T, t \in R$, and $e \in E \setminus (E_g \cup E_b)\} \neq \varnothing$ **do**
5: Choose such an edge e arbitrarily.
6: **if** `IsGoodEdge`(e, E_g) **then**
7: $E_g \leftarrow E_g \cup \{e\}$, $R \leftarrow R \cup \{s\}$, $E_b \leftarrow \varnothing$
8: **else**
9: $E_b \leftarrow E_b \cup \{e\}$
10: **return** R

Algorithm 3: Sure-DirFWMP(ℓ, α)-positive-Reach(T)

Finally, we describe Algorithm 3 (`SureDirFWMPPosReach`) that returns the set of all vertices in an MDP that are winning for sure-DirFWMP(ℓ, α)-positive-Reach(T) for a target set $T \subseteq V$. This algorithm is used as a subroutine in Algorithm 2 (`SureDirFWMPASBüchi`). The MDP shown in Figure 7 in Appendix A shows that every vertex in an MDP may be winning for sure-DirFWMP(ℓ, α) and for positive-Reach(T) separately, and yet it may be the case that no vertex in the MDP is winning for the combination sure-DirFWMP(ℓ, α)-positive-Reach(T).

Description of Algorithm 3. We first compute in Line 1 the set $W_{\mathsf{SD}\alpha}$ of vertices from which $\mathcal{P}_\bigcirc$ can satisfy sure-DirFWMP(ℓ, α). Since DirFWMP(ℓ, α) is closed under suffixes, from Proposition 1 we have that $W_{\mathsf{SD}\alpha}$ induces a subMDP $\mathcal{M}[W_{\mathsf{SD}\alpha}]$ of $\mathcal{M}$. We are not interested in vertices in $V \setminus W_{\mathsf{SD}\alpha}$, and thus we restrict the MDP $\mathcal{M}$ to $\mathcal{M}[W_{\mathsf{SD}\alpha}]$ (that is, we refer to the subMDP $\mathcal{M}[W_{\mathsf{SD}\alpha}]$ as $\mathcal{M}$) where all vertices are winning for sure-DirFWMP(ℓ, α), and we refer to $T \cap W_{\mathsf{SD}\alpha}$ as T (Line 2).

We define a set R of vertices that at the end of the algorithm will consist of precisely all the vertices in $\mathcal{M}$ that are winning for sure-DirFWMP(ℓ, α)-positive-Reach(T) in $\mathcal{M}$. Since all vertices in T satisfy sure-DirFWMP(ℓ, α) and also trivially satisfy positive-Reach(T) (as they are already in T), we initialize R with T in Line 3. As we discover more vertices that are winning for sure-DirFWMP(ℓ, α)-positive-Reach(T), we add them to R. We always only add vertices to R and never remove vertices from R.

We also define sets E_g and E_b of edges (*good* and *bad* edges respectively). We initialize E_g and E_b with empty sets in Line 3, and add edges to these sets over the course of the algorithm in the **while** loop (Lines 4 to 9) as we learn more about the MDP $\mathcal{M}$. The **while** loop runs as long as there exists an edge $e = (s,t)$ in $\mathcal{M}$ such that s is in $V \setminus T$ and t is in R, and e does not belong to E_g or E_b. If such an edge exists, then one such edge $e = (s,t)$ is chosen arbitrarily. For this edge e, we run the procedure `IsGoodEdge` (given in Appendix B) to determine if e is good or bad with respect to the set E_g. Intuitively, an edge e is good with respect to E_g if starting from s, with positive probability it is the case that the token takes the edge e to reach R and eventually T using only edges in E_g while simultaneously satisfying sure-DirFWMP(ℓ, α). Formally, an edge e in $\mathcal{M}$ is *good* with respect to E_g if there exists a winning strategy σ_e for sure-DirFWMP(ℓ, α)-positive-Reach$_{\{e\} \cup E_g}(T)$ starting from edge e in $\mathcal{M}$. That is, all outcomes of σ_e with initial edge e satisfy DirFWMP(ℓ, α), and with positive probability, an outcome of σ_e with initial edge e reaches T by subsequently taking only edges in E_g. The edge e is *bad* with respect to E_g if no such winning strategy σ_e exists.

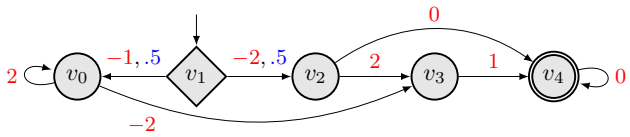

Figure 3 An MDP where we want to determine the winning region for sure-DirFWMP($\ell = 3, \alpha = 0$)-positive-Reach($T = \{v_4\}$). The winning region is $\{v_1, v_2, v_3, v_4\}$.

If `IsGoodEdge` returns false, then the edge is added to E_b (Line 9). Otherwise, if `IsGoodEdge` returns true, then the edge e is added to E_g and the vertex s is added to the winning region R for sure-DirFWMP(ℓ, α)-positive-Reach(T) in $\mathcal{M}$ if s is not already in R (Line 7). This is because the winning strategy σ_e for sure-DirFWMP(ℓ, α)-positive-Reach$_{\{e\} \cup E_g}(T)$ from edge (s, t) immediately gives a winning strategy for sure-DirFWMP(ℓ, α)-positive-Reach(T) from s. Note that given the set E_g, we can reconstruct R as $T \cup \{u \in V \mid (u, v) \in E_g\}$.

Finally, we note that if an edge e is good with respect to a set E_g, then e is also good with respect to all $E'_g \supseteq E_g$, since satisfaction of Reach$_{\{e\} \cup E_g}(T)$ implies satisfaction of Reach$_{\{e\} \cup E'_g}(T)$. Thus, once we add an edge to E_g, it stays in E_g for the rest of the algorithm. On the other hand, an edge e that is bad with respect to E_g may not be bad for $E'_g \supseteq E_g$, as non-existence of a winning strategy for sure-DirFWMP(ℓ, α)-positive-Reach$_{\{e\} \cup E_g}(T)$ does not imply the non-existence of a winning strategy for sure-DirFWMP(ℓ, α)-positive-Reach$_{\{e\} \cup E'_g}(T)$. Therefore, we reset E_b to the empty set each time an edge is added to E_g (Line 7), and we check whether these edges are good or bad with respect to the larger set $E_g \cup \{e\}$ in a later iteration of the **while** loop.

In each iteration of the **while** loop, either an edge is added to E_b, or an edge is added to E_g and the set E_b is reset to empty. We exit the **while** loop when every edge starting in $V \setminus T$ and ending in R belongs to either E_g or E_b. Since there are finitely many edges in the MDP $\mathcal{M}$, the **while** loop is eventually exited. In fact, the **while** loop runs for at most $|E|^2$ iterations, since after at most every $|E|$ additions to E_b, an edge is added to E_g and the set E_b is reset to empty. After exiting the **while** loop, we return the set R of vertices (Line 10).

▶ **Example 7.** Consider the MDP in Figure 3. We explain how Algorithm 3 works on this MDP. We consider the winning condition sure-DirFWMP$(\ell = 3, \alpha = 0)$-positive-Reach$(T = \{v_4\})$. Initially, we have $T = R = \{v_4\}$ and $E_g = E_b = \varnothing$. Suppose the algorithm first considers the edge (v_2, v_4). Note that $v_2 \in V \setminus T$ and $v_4 \in R$. The procedure `IsGoodEdge` returns that this is a good edge with respect to $E_g = \varnothing$ since starting with this edge (v_2, v_4) ensures that T is reached with a positive probability (in fact, surely) and then looping at v_4 ensures satisfaction of sure-DirFWMP(ℓ, α). Hence, we add v_2 to R to get $R = \{v_2, v_4\}$ and add the edge (v_2, v_4) to E_g. Suppose the edge (v_1, v_2) is considered next. This edge is not good with respect to $E_g = \{(v_2, v_4)\}$ since starting with (v_1, v_2), it is not possible to satisfy sure-DirFWMP$(3, 0)$-positive-Reach$_{\{(v_1,v_2),(v_2,v_4)\}}(T)$. Indeed, in order to reach T starting from the edge (v_1, v_2) and then using only edges in E_g, the path $v_1 v_2 v_4$ must be taken, but this forces the token to then loop on v_4 forever, which results in the 0-window starting at v_1 never closing and sure-DirFWMP$(3, 0)$ not being satisfied. Hence, the edge (v_1, v_2) is added to the set E_b. Suppose the edge (v_3, v_4) is considered next. It is a good edge with respect to E_g since it leads to T from v_3 and sure-DirFWMP$(3, 0)$ can be simultaneously from v_3, and thus, E_g is updated to $\{(v_2, v_4), (v_3, v_4)\}$, the set E_b is reset to $\varnothing$, and R is updated to $\{v_2, v_3, v_4\}$. Next consider the edge (v_2, v_3) which is a good edge with respect to E_g and thus E_g is updated to $\{(v_2, v_4), (v_3, v_4), (v_2, v_3)\}$ while E_b remains $\varnothing$ and R is still $\{v_2, v_3, v_4\}$.

Now suppose (v_1, v_2) is considered again. This time we see that this edge is good with respect to E_g since taking the path $v_1v_2v_3v_4$ and then looping on v_4 ensures reaching T through only edges in $\{(v_1, v_2)\} \cup E_g$ as well as satisfaction of sure-DirFWMP$(3, 0)$. Thus, E_g is updated to $\{(v_2, v_4), (v_3, v_4), (v_2, v_3), (v_1, v_2)\}$ and R is updated to $\{v_1, v_2, v_3, v_4\}$. Finally, the edge (v_0, v_3) is considered and found to be not good with respect to E_g, and is added to E_b. All edges starting in $V \setminus T$ and ending in R belong to either E_g or E_b. The edges (v_1, v_0) and (v_0, v_0) are not considered for checking if they are good since $v_0 \notin R$ and the **while** loop terminates. The set $R = \{v_1, v_2, v_3, v_4\}$ returned by Algorithm 3 is exactly the set of vertices from which sure-DirFWMP$(\ell = 3, \alpha = 0)$-positive-Reach$(T = \{v_4\})$ can be satisfied. ⌟

Description of `IsGoodEdge` procedure. Given an MDP $\mathcal{M}$, the procedure `IsGoodEdge`$(e = (s, t), E_g)$ determines if the edge e is winning for sure-DirFWMP(ℓ, α)-positive-Reach$_{\{e\} \cup E_g}(T)$ in $\mathcal{M}$. The procedure augments some vertices and edges to $\mathcal{M}$ to construct an MDP $\mathcal{M}_e$ that has at most thrice as many vertices as $\mathcal{M}$, and returns true if and only if a designated vertex $\hat{s}$ is winning for sure-DirFWMP(ℓ, α) in $\mathcal{M}_e$. This reduces satisfaction of sure-DirFWMP(ℓ, α)-positive-Reach$_{E_g}(T)$ in $\mathcal{M}$ to satisfaction of sure-DirFWMP(ℓ, α) in $\mathcal{M}_e$, which we know how to solve from [10, Algorithm 2].

▶ **Theorem 8.** *Algorithm 3 computes the winning region for $\mathcal{P}_{\bigcirc}$ for sure-DirFWMP(ℓ, α)-positive-Reach(T) in $\mathcal{M}$.*

We conclude this section by discussing bounds on the memory requirement and the complexity for the sure-almost-sure satisfaction of FWMP(ℓ) objectives.

Memory requirement and complexity for sure-almost-sure FWMP(ℓ)

Memory requirement: lower bound. It is known that memory of size ℓ suffices for satisfying sure-FWMP(ℓ, α) [16, Theorem 4.4] and for almost-sure-FWMP(ℓ, α) [16, Theorem 6.5], and this is independent of the size of the MDP. In Example 9, we show that $\Omega(\max\{|V|, \ell\})$ memory may be necessary for the sure-almost-sure satisfaction of the FWMP(ℓ) objective.

▶ **Example 9.** Consider the MDP in Figure 4. Note that the only vertex belonging to $\mathcal{P}_{\bigcirc}$ with more than one out-edge is a. Thus, a strategy of $\mathcal{P}_{\bigcirc}$ can be fully specified by describing the behaviour of the token when it reaches a. Starting from vertex v_3, the memoryless strategy that always chooses v_2 from vertex a satisfies sure-FWMP$(2, 0)$ and the memoryless strategy that always chooses vertex b from a satisfies almost-sure-FWMP$(2, 5)$. However, neither of these strategies are winning for sure-FWMP$(2, 0)$-almost-sure-FWMP$(2, 5)$.

In order to satisfy sure-FWMP$(2, 0)$, player $\mathcal{P}_{\bigcirc}$ must eventually close all 0-windows in at most 2 steps, and in order to satisfy almost-sure-FWMP$(2, 5)$ from v_3, the token must reach vertex b repeatedly as long as it has not reached c. However, each time the token moves to v_3, the token then has to take the edge (v_3, a) which opens a 0-window. In order to reach b from v_3 with all 0-windows closed in at most 2 steps, the token must visit v_2 and v_1 on the way. This is because when the token reaches a from v_3, the token must move to v_2 to close the 0-window starting at v_3. The token then reaches a, from where moving to either v_2 or v_1 results in closing the 0-window opened at v_2. If the token moves between v_2 and a forever, then the token makes no progress towards reaching b, and thus it must eventually move from a to v_1. From v_1, the token can move to a and then to b while also ensuring that the 0-window starting at v_1 closes in 2 steps. Thus, the token needs to make visits to v_2, v_1, and b from a to close all 0-windows in 2 steps, and this requires a strategy of memory size 3.

This example can be generalized to an MDP $\mathcal{M}_n$ with $n + 3$ vertices $\{v_n, v_{n-1}, \ldots, v_1, a, b, c\}$, and edges (v_i, a) with payoff $-i$ and edges (a, v_{i-1}) with payoff $+i$ for $2 \leq i \leq n$, an edge (a, b) with payoff 1, an edge (b, v_n) with payoff 0, an edge (b, c) with payoff 0, and

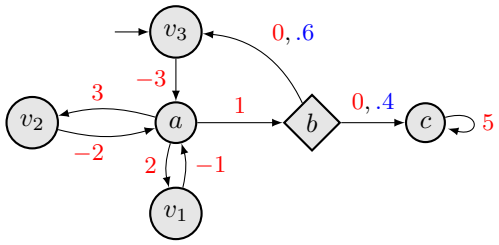

Figure 4 There exist memoryless winning strategies for sure-FWMP$(2,0)$ and for almost-sure-FWMP$(2,5)$ from all vertices in the MDP, but memory of size at least 3 is required for a strategy that is winning for sure-FWMP$(2,0)$-almost-sure-FWMP$(2,5)$ from v_3.

a self-loop on c with payoff 5. It can be seen that memory of size n is necessary to satisfy sure-FWMP$(2,0)$-almost-sure-FWMP$(2,5)$ in $\mathcal{M}_n$ from v_n. Since satisfying sure-FWMP(ℓ,α) requires at least ℓ memory in general, at least $\Omega(\max\{|V|,\ell\})$ memory is necessary to satisfy sure-FWMP(ℓ,α)-almost-sure-FWMP(ℓ,β). ⌟

Memory requirement: upper bound. We give an upper bound for the memory requirement for winning strategies for sure-almost-sure satisfaction of FWMP(ℓ). We construct in [20] a winning strategy σ_{sdpr} for sure-DirFWMP(ℓ,α)-positive-Reach(T) having memory size at most $3\cdot|V|\cdot\ell$. Then, we use this strategy σ_{sdpr} to construct a winning strategy σ_{sdab} for sure-DirFWMP(ℓ,α)-almost-sure-Büchi(T) of memory size no greater than that of σ_{sdpr}. Now, we use σ_{sdab} to describe a winning strategy σ_{sas} for sure-almost-sure satisfaction of FWMP(ℓ).

Recall that the winning region for sure-almost-sure satisfaction of FWMP(ℓ) has three types of regions: A^i (attractor region) for $1\le i\le k$, W^i_{sdab} (sure-DirFWMP(ℓ,α)-almost-sure-Büchi(P^i) winning region) for $1\le i\le k$, and W^0 (sure-FWMP(ℓ,β) winning region). We describe a winning strategy σ_{sas} for sure-FWMP(ℓ,α)-almost-sure-FWMP(ℓ,β):

- If the token is in A^i for some $1\le i\le k$, then σ_{sas} follows a sure-attractor strategy to reach $W^i_{\mathsf{sdab}}\cup W^{i-1}$.
- If the token is in W^i_{sdab} for some $1\le i\le k$, then σ_{sas} follows the winning strategy σ_{sdab} for sure-DirFWMP(ℓ,α)-almost-sure-Büchi(P^i). This ensures that DirFWMP(ℓ,α) is satisfied in all outcomes, and with probability 1, the token reaches W^{i-1}.
- If the token is in W^0, then σ_{sas} follows a winning strategy for sure-FWMP(ℓ,β).

The memory requirement of σ_{sas} is no greater than that of σ_{sdab}, that is at most $3\cdot|V|\cdot\ell$.

Complexity. The **repeat** loop in Algorithm 1 runs for at most $|V|$ iterations and is dominated by the call to Algorithm 2. Now, Algorithm 2 is a recursive algorithm with recursion depth at most $|V|$ and it is dominated by the call to Algorithm 3. In Algorithm 3, we have a **repeat** loop that runs at most $|E|^2$ times. In each iteration of the loop, we make a call to `IsGoodEdge` which involves calling `SureDirFWMP` on the augmented MDP $\mathcal{M}_e$ that has at most thrice as many vertices as the original MDP $\mathcal{M}$. We note that `SureDirFWMP` [10, Lemma 5] and `SureFWMP` [10, Lemma 6] both run in time that is polynomial in the size of the input when ℓ is given in unary. Thus, the total time complexity of Algorithm 1 (`SureAlmostSureFWMP`) is polynomial in the size of the input when ℓ is given in unary.

▶ **Theorem 10.** *The sure-almost-sure problem for the FWMP(ℓ) objective is in* PTIME *when ℓ is given in unary. The memory required is at least $\Omega(\max\{|V|,\ell\})$ and at most $\mathcal{O}(|V|\cdot\ell)$.*

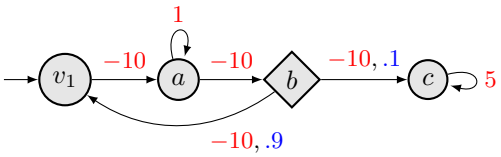

Figure 5 There exist memoryless winning strategies for sure-BWMP(0) and for almost-sure-BWMP(5) from all vertices in the MDP, but memory of size at least 31 is required for a strategy that is winning for sure-BWMP(0)-almost-sure-BWMP(5) from v_1.

4.2 Sure-almost-sure satisfaction of BWMP

Recall that a play π satisfies BWMP(α), the bounded window mean-payoff objective with threshold α, if there exists a window length $\ell \geq 1$ such that π satisfies FWMP(ℓ, α). To solve the sure-almost-sure satisfaction problem for BWMP, we consider Algorithm 1 for FWMP(ℓ) and replace every occurrence of FWMP(ℓ) with BWMP. Thus, the algorithm for sure-almost-sure BWMP has the same structure as Algorithm 1, that is, it has subalgorithms for sure-DirBWMP(α)-almost-sure-Büchi(T) and sure-DirBWMP(α)-positive-Reach(T) (analogous to sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T) and sure-DirFWMP(ℓ, α)-positive-Reach(T) respectively, obtained again by replacing all occurrences of FWMP(ℓ) with BWMP).

We recall [10, Lemma 10] which states that if a vertex v is winning for sure-TP(0) in $\mathcal{M}$ then v is winning for sure-GW($\ell', 0$) in $\mathcal{M}$ for $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$. This lets us replace occurrences of BWMP with FWMP(ℓ'') (where $\ell'' = 3 \cdot |V| \cdot (3 \cdot |V| \cdot w_{\max} + 1)$) and reuse results for the sure-almost-sure satisfaction of FWMP(ℓ''). We use ℓ'' instead of ℓ' as Algorithm 3 works with MDPs $\mathcal{M}_e$ that have at most thrice as many vertices as $\mathcal{M}$. This leads us to Lemma 11. More details can be found in [20].

▶ **Lemma 11.** *For every vertex v in an MDP $\mathcal{M}$, the vertex v is winning for sure-BWMP(α)-almost-sure-BWMP(β) if and only if v is winning for sure-FWMP(ℓ'', α)-almost-sure-FWMP(ℓ'', β) for $\ell'' = 3 \cdot |V| \cdot (3 \cdot |V| \cdot w_{\max} + 1)$.*

Memory requirement and complexity for sure-almost-sure BWMP

Memory requirement: lower bound. It is known that memoryless strategies suffice for satisfying sure-BWMP(α) [10, Theorem 19] and for satisfying almost-sure-BWMP(α) [7, Theorem 6.2(b)]. In Example 12, we show that memory of size at least $\Omega(|V| \cdot w_{\max})$ may be necessary for the sure-almost-sure satisfaction of BWMP.

▶ **Example 12.** Consider the MDP in Figure 5. Starting from vertex v_1, a memoryless strategy that always chooses the self-loop on vertex a satisfies sure-BWMP(0) while a memoryless strategy that always chooses vertex b from a satisfies almost-sure-BWMP(5). However, neither of these strategies are winning for sure-BWMP(0)-almost-sure-BWMP(5).

In order to satisfy almost-sure-BWMP(5) from v_1, the token needs to repeatedly reach vertex b until it reaches vertex c. However, each time the token moves from b to v_1, the token has to go through the edges (b, v_1), (v_1, a), and (a, b) before it can reach b again. This accrues a payoff of -30 and is thus an open 0-window. In order to also ensure satisfaction of sure-BWMP(0), player $\mathcal{P}_\bigcirc$ needs to eventually always be closing 0-windows. If the token takes the self-loop on a fewer than 30 times each time it reaches a before moving on to b, then this results in an outcome that contains infinitely many open 0-windows that never close. On the other hand, if each time the token reaches a the token takes the self-loop on a at least 30 times before forwarding the token to b, then as long as the token does not reach c

Algorithm 4 `SureLimitSureFWMP`$(\mathcal{M}, \ell, \alpha, \beta)$.

In: An MDP $\mathcal{M} = ((V, E), (V_\bigcirc, V_\Diamond), \mathbb{P}, w)$, window length ℓ, sure threshold α, and limit-sure threshold β

Out: The winning region for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in $\mathcal{M}$

1: $W_{\mathsf{S}\alpha} \leftarrow$ `SureFWMP`$(\mathcal{M}, \ell, \alpha)$
2: $W_{\mathsf{AS}\beta} \leftarrow$ `AlmostSureFWMP`$(\mathcal{M}[W_{\mathsf{S}\alpha}], \ell, \beta)$
3: **return** $W_{\mathsf{AS}\beta}$

all 0-windows get closed in at most 33 steps, and once the token reaches c, all 5-windows get closed in 1 step. We see that this memoryful strategy which uses a counter of size 31 is winning for sure-BWMP(0)-almost-sure-BWMP(5).

This example can be generalized to an MDP $\mathcal{M}_{n,w_{\max}}$ with $n+3$ vertices $\{v_n, v_{n-1}, \ldots, v_1, a, b, c\}$ and edges (v_i, v_{i-1}) for $2 \leq i \leq n$, (v_1, a), (a, b), (b, v_n), and (b, c) each with payoff $-w_{\max}$, a self-loop on a with payoff 1, and a self-loop on c with payoff 5. A winning strategy for sure-BWMP(0)-almost-sure-BWMP(5) from v_n in $\mathcal{M}_{n,w_{\max}}$ requires memory of size at least $(n+2) \cdot w_{\max} + 1$, which is $\Omega(|V| \cdot w_{\max})$. ⌟

Memory requirement: upper bound. By Lemma 11, if there exists a winning strategy σ_{B} from a vertex v for sure-almost-sure BWMP, then there also exists a winning strategy σ_{F} from v for sure-almost-sure FWMP(ℓ'') for $\ell'' = 3 \cdot |V| \cdot (3 \cdot |V| \cdot w_{\max} + 1)$, and moreover, this strategy σ_{F} is also winning strategy for sure-almost-sure BWMP from v. From Theorem 5, we have that the memory of size $|V| \cdot \ell''$ (which is $\mathcal{O}(|V|^3 \cdot |w_{\max}|)$) is sufficient for sure-almost-sure FWMP(ℓ'') and thus is also sufficient for sure-almost-sure BWMP.

Complexity. From [10, Theorem 19], we have that satisfaction of sure-BWMP(α) and of sure-DirBWMP(α) are both in NP ∩ coNP. Similar to the case of the sure-almost-sure satisfaction of FWMP(ℓ), the algorithm for the sure-almost-sure satisfaction of BWMP makes a polynomial number of calls to algorithms for satisfaction of sure-BWMP(α), sure-BWMP(β), and sure-DirBWMP(α). Thus, we have that sure-almost-sure satisfaction of BWMP is in $\mathsf{PTIME}^{\mathsf{NP}\cap\mathsf{coNP}}$, which is the same as NP ∩ coNP [24].

▶ **Theorem 13.** *The sure-almost-sure problem for the BWMP objective is in* NP ∩ coNP. *The memory required is at least* $\Omega(|V| \cdot w_{\max})$ *and at most* $\mathcal{O}(|V|^3 \cdot |w_{\max}|)$.

5 Sure-limit-sure for FWMP and BWMP

In this section, we see an algorithm to solve the sure-limit-sure satisfaction of FWMP(ℓ) and BWMP objectives. Algorithm 4 (`SureLimitSureFWMP`) returns the set of all vertices v that are winning for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in an MDP $\mathcal{M}$. An analogous algorithm to compute the winning region for sure-limit-sure satisfaction of BWMP is obtained by replacing `SureFWMP` and `AlmostSureFWMP` with `SureBWMP` and `AlmostSureBWMP` respectively.

Description of Algorithm 4. We first compute the set $W_{\mathsf{S}\alpha}$ of vertices that are winning for sure-FWMP(ℓ, α) in $\mathcal{M}$ using [10, Algorithm 1]. Then, in the subMDP $\mathcal{M}[W_{\mathsf{S}\alpha}]$ induced by $W_{\mathsf{S}\alpha}$, we compute using [7, Algorithm 1] the set $W_{\mathsf{AS}\beta}$ of vertices that are winning for almost-sure-FWMP(ℓ, β), and we return this set $W_{\mathsf{AS}\beta}$. We note from [7, Lemma 6.1] that there exists a set $W_{\mathsf{S}\beta} \subseteq W_{\mathsf{AS}\beta}$ of vertices that are winning for sure-FWMP(ℓ, β) in $\mathcal{M}[W_{\mathsf{S}\alpha}]$.

▶ **Theorem 14.** *Algorithm 4 computes the winning region for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in $\mathcal{M}$.*

Memory requirement and complexity for sure-limit-sure satisfaction

SLS-FWMP(ℓ). For every vertex v in the set $W_{\mathsf{AS}\beta}$ returned by Algorithm 4, for all $\varepsilon > 0$, we describe a winning strategy σ_ε for sure-FWMP(ℓ, α)-probability$(1-\varepsilon)$-FWMP(ℓ, β) from v. We fix a sufficiently large integer N that depends on the number of vertices in $W_{\mathsf{AS}\beta}$, the minimum edge probability in $W_{\mathsf{AS}\beta}$, and ε. The strategy σ_ε is defined in two phases: in the first phase σ_ε follows a memoryless almost-sure attractor strategy to $W_{\mathsf{S}\beta}$ until either the token reaches $W_{\mathsf{S}\beta}$ or N steps of the play elapse (whichever occurs first), after which the second phase begins where σ_ε follows a winning strategy for either sure-FWMP(ℓ, β) or sure-FWMP(ℓ, α) (depending on whether the token is in $W_{\mathsf{S}\beta}$ or $W_{\mathsf{AS}\beta} \setminus W_{\mathsf{S}\beta}$ respectively at the end of the first phase). In [20], we show lower and upper bounds for N, respectively $\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{|V|}}\right)$ and $\mathcal{O}\left(\frac{|V|\cdot\log(1/\varepsilon)}{(\mathbb{P}_{\min})^{|V|}}\right)$, in order for the token to reach $W_{\mathsf{S}\beta}$ with probability at least $1-\varepsilon$ in N steps. A counter of size N counts the length of the first phase, and memory of size ℓ is sufficient for a winning strategy for sure-FWMP(ℓ, β) or sure-FWMP(ℓ, α) [16, Theorem 4.4] in the second phase. Thus, memory size of $N + \ell$ suffices for σ_ε.

Since sure satisfaction and almost-sure satisfaction of the FWMP(ℓ) objectives can be checked in polynomial time [7, 10], we have that Algorithm 4 runs in polynomial time.

▶ **Theorem 15.** *Sure-limit-sure satisfaction of FWMP(ℓ) objectives is in* PTIME. *The memory required is at least $\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{|V|}} + \ell\right)$ and at most $\mathcal{O}\left(\frac{|V|\cdot\log(1/\varepsilon)}{(\mathbb{P}_{\min})^{|V|}} + \ell\right)$.*

SLS-BWMP. A winning strategy for sure-BWMP(α)-probability$(1-\varepsilon)$-BWMP(β) follows a similar two-phase structure as that for FWMP(ℓ). The first phase requires a counter of size N (with the same bounds as above), and memoryless strategies exist for the satisfaction of sure-BWMP(α) or sure-BWMP(β) [10, Theorem 19] in the second phase. Thus, we have that memory of size $N + 1$ suffices for the sure-limit-sure satisfaction of BWMP.

Since sure satisfaction and almost-sure satisfaction of the BWMP objectives are in NP $\cap$ coNP [7, 10], and the algorithm for sure-limit-sure satisfaction for BWMP makes one call each to the two algorithms, we have that sure-limit-sure satisfaction of BWMP objectives is in $\mathsf{PTIME}^{\mathsf{NP}\cap\mathsf{coNP}}$, which is the same as NP $\cap$ coNP [24].

▶ **Theorem 16.** *Sure-limit-sure satisfaction of BWMP objectives is in* NP $\cap$ coNP. *The memory required is at least $\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{|V|}}\right)$ and at most $\mathcal{O}\left(\frac{|V|\cdot\log(1/\varepsilon)}{(\mathbb{P}_{\min})^{|V|}}\right)$.*

References

1 K. R. Apt and E. Grädel. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011. doi:10.1017/CBO9780511973468.

2 C. Baier and J-P. Katoen. *Principles of model checking*. MIT Press, 2008.

3 R. Berthon, S. Guha, and J.-F. Raskin. Mixing Probabilistic and non-Probabilistic Objectives in Markov Decision Processes. In *LICS*, pages 195–208. ACM, 2020. doi:10.1145/3373718.3394805.

4 R. Berthon, J-P. Katoen, and T. Winkler. Markov Decision Processes with Sure Parity and Multiple Reachability Objectives. In *RP*, volume 15050 of *Lecture Notes in Computer Science*, pages 203–220. Springer, 2024. doi:10.1007/978-3-031-72621-7_14.

5 R. Berthon, M. Randour, and J.-F. Raskin. Threshold Constraints with Guarantees for Parity Objectives in Markov Decision Processes. In *ICALP*, volume 80 of *LIPIcs*, pages 121:1–121:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.121.

6 B. Bordais, S. Guha, and J-F. Raskin. Expected Window Mean-Payoff. In *FSTTCS*, volume 150 of *LIPIcs*, pages 32:1–32:15, 2019. doi:10.4230/LIPIcs.FSTTCS.2019.32.

7 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, December 2020. doi:10.23638/LMCS-16(4:13)2020.

8 V. Bruyère, E. Filiot, M. Randour, and J-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Information and Computation*, 254:259–295, 2017. doi:10.1016/j.ic.2016.10.011.

9 K. Chatterjee, L. de Alfaro, and T. A. Henzinger. The Complexity of Stochastic Rabin and Streett Games. In *ICALP*, pages 878–890. Springer, 2005. doi:10.1007/11523468_71.

10 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.

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

12 K. Chatterjee, Z. Křetínská, and J. Křetínský. Unifying Two Views on Multiple Mean-Payoff Objectives in Markov Decision Processes. *Logical Methods in Computer Science*, Volume 13, Issue 2, July 2017. doi:10.23638/LMCS-13(2:15)2017.

13 K. Chatterjee and N. Piterman. Combinations of Qualitative Winning for Stochastic Parity Games. In *CONCUR*, volume 140 of *LIPIcs*, pages 6:1–6:17, 2019. doi:10.4230/LIPIcs.CONCUR.2019.6.

14 L. Clemente and J.-F. Raskin. Multidimensional beyond Worst-Case and Almost-Sure Problems for Mean-Payoff Objectives. In *LICS*, pages 257–268. IEEE, 2015. doi:10.1109/LICS.2015.33.

15 L. Doyen, P. Gaba, and S. Guha. Expectation in Stochastic Games with Prefix-Independent Objectives. In *CONCUR*, volume 348 of *LIPIcs*, pages 16:1–16:19, 2025. doi:10.4230/LIPIcs.CONCUR.2025.16.

16 L. Doyen, P. Gaba, and S. Guha. Stochastic Window Mean-Payoff Games. *Logical Methods in Computer Science*, Volume 21, Issue 2, June 2025. doi:10.46298/lmcs-21(2:19)2025.

17 L. Doyen and S. Guha. Algorithm and Strategy Construction for Sure-Almost-Sure Stochastic Parity Games. In *STACS*, pages 34:1–34:20, 2026. doi:10.4230/LIPIcs.STACS.2026.34.

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

19 P. Gaba and S. Guha. Optimising Expectation with Guarantees for Window Mean Payoff in Markov Decision Processes. In *AAMAS*, pages 820–828, 2025. doi:10.5555/3709347.3743600.

20 P. Gaba and S. Guha. Sure-almost-sure and Sure-limit-sure Window Mean Payoff in Markov Decision Processes, 2026. doi:10.48550/arXiv.2605.12191.

21 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.

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

23 M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

24 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.

25 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.

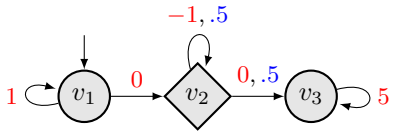

Figure 6 The winning region for sure-FWMP$(2,1)$ is $\{v_1, v_3\}$ and the winning region for limit-sure-FWMP$(2,5)$ is $\{v_1, v_2, v_3\}$. However, the winning region for sure-FWMP$(2,1)$-limit-sure-FWMP$(2,5)$ is $\{v_3\}$, a strict subset of the intersection $\{v_1, v_3\}$ of the winning regions for sure-FWMP$(2,1)$ and for limit-sure-FWMP$(2,5)$.

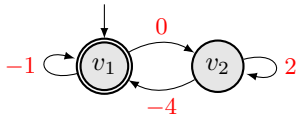

Figure 7 Both vertices are winning for sure-DirFWMP$(2,0)$ and for almost-sure-Büchi$(\{v_1\})$ (and even sure-Büchi$(\{v_1\})$), but neither vertex is winning for sure-DirFWMP$(2,0)$-almost-sure-Büchi$(\{v_1\})$.

A Additional remarks

▶ Remark 17 (SAS and SLS satisfaction of direct objectives). We only consider sure-almost-sure and sure-limit-sure satisfaction of the *prefix-independent* versions of the window mean-payoff objectives. It is shown in [7, Remark 4.10] that almost-sure satisfaction of DirFWMP(ℓ,α) (and thus also limit-sure satisfaction of DirFWMP(ℓ,α)) is equivalent to the sure satisfaction of DirFWMP(ℓ,α). Thus, sure-almost-sure satisfaction (and sure-limit-sure satisfaction) of DirFWMP(ℓ) reduces to just the sure satisfaction of DirFWMP(ℓ,β). It is also shown in [7, Example 7.1] that almost-sure satisfaction of DirBWMP is not well-behaved. Thus, sure-almost-sure and sure-limit-sure satisfaction of DirBWMP are also not well-behaved and we do not study these problems in this paper.

▶ Remark 18. We show using the MDP in Figure 6 that the winning region for sure-FWMP(ℓ,α)-limit-sure-FWMP(ℓ,β) is in general not equal to the intersection of winning region for sure-FWMP(ℓ,α) and the winning region for limit-sure-FWMP(ℓ,β).

Observe that the winning region for sure-FWMP$(2,1)$ is equal to $\{v_1, v_3\}$, since the strategy that always takes the self-loops on v_1 and v_3 is winning for sure-FWMP$(2,1)$ from v_1 and v_3, and there exists an outcome starting from v_2 that loops on v_2 forever giving a play that is losing for FWMP$(2,1)$. On the other hand, all vertices in this MDP are winning for almost-sure-FWMP$(2,5)$ (and thus also for limit-sure-FWMP$(2,5)$) as witnessed by the strategy that moves the token from v_1 to v_2 and loops on v_3. However, only the vertex v_3 is winning for sure-FWMP$(2,1)$-limit-sure-FWMP$(2,5)$. There does not exist a strategy that simultaneously satisfies both sure-FWMP$(2,1)$ and limit-sure-FWMP$(2,5)$ from v_1 as the former cannot be satisfied by moving to v_2 and the latter cannot be satisfied by staying at v_1.

However, we showed in Section 5 that the winning region for sure-FWMP(ℓ,α)-limit-sure-FWMP(ℓ,β) is the composition of the winning region for limit-sure-FWMP(ℓ,β) (equivalently almost-sure-FWMP(ℓ,β)) and for sure-FWMP(ℓ,α). That is, the winning region for sure-FWMP(ℓ,α)-limit-sure-FWMP(ℓ,β) is equal to the winning region for limit-sure-FWMP(ℓ,β) (equivalently almost-sure-FWMP(ℓ,β)) in the subMDP induced by the winning region for sure-FWMP(ℓ,α) in the original MDP. ⌟

B Full version of Algorithm 3

Algorithm 5 `SureDirFWMPPosReach`$(\mathcal{M}, \ell, \alpha, T)$ (full version).

In: An MDP $\mathcal{M} = ((V, E), (V_\bigcirc, V_\Diamond), \mathbb{P}, w)$, window length ℓ, threshold α, and target set T
Out: The winning region for sure-DirFWMP(ℓ, α)-positive-Reach(T) in $\mathcal{M}$

$W_{\mathsf{SD}\alpha} \leftarrow$ `SureDirFWMP`$(\mathcal{M}, \ell, \alpha)$
$\mathcal{M} \leftarrow \mathcal{M}[W_{\mathsf{SD}\alpha}]$, $T \leftarrow T \cap W_{\mathsf{SD}\alpha}$
$R \leftarrow T$, $E_g, E_b \leftarrow \varnothing$
while $\{e = (s,t) \in E \mid s \in V \setminus T, t \in R, \text{ and } e \in E \setminus (E_g \cup E_b)\} \neq \varnothing$ **do**
    Choose such an edge e arbitrarily.
    if `IsGoodEdge`(e, E_g) **then**
        $E_g \leftarrow E_g \cup \{e\}$, $R \leftarrow R \cup \{s\}$, $E_b \leftarrow \varnothing$
    else
        $E_b \leftarrow E_b \cup \{e\}$
return R

procedure `IsGoodEdge`$(e = (s,t), E_g)$
    Construct MDP $\mathcal{M}'_e$ in the following way:
        $V \leftarrow$ the set of vertices in $\mathcal{M}$
        $\widetilde{R} \leftarrow \{\tilde{v} \mid v \in R\}$, where $\tilde{v}$ belongs to $\mathcal{P}_\bigcirc$ if and only if v belongs to $\mathcal{P}_\bigcirc$.
        The set of vertices of $\mathcal{M}'_e$ is $V \cup \widetilde{R} \cup \{\hat{s}\}$.
        The vertex $\hat{s}$ belongs to $\mathcal{P}_\bigcirc$ if and only if s belongs to $\mathcal{P}_\bigcirc$.
        $E \leftarrow$ the set of edges in $\mathcal{M}$
        $\widetilde{E}_g \leftarrow \{(\tilde{u}, \tilde{v}) \mid u, v \in R, (u,v) \in E_g)\}$
        $\widetilde{E}_\Diamond \leftarrow \{(\tilde{u}, v) \mid u \in (R \setminus T) \cap V_\Diamond, v \in V, (u,v) \in E \setminus E_g)\}$
        $\widehat{E}_\Diamond \leftarrow$ **if** $s \in V_\bigcirc$ **then** $\varnothing$ **else** $\{(\hat{s}, v) \mid (s,v) \in E \text{ and } v \neq t\}$
        The set of edges of $\mathcal{M}'_e$ is $E \cup \widetilde{E}_g \cup \widetilde{E}_\Diamond \cup \{(\hat{s}, \tilde{t})\} \cup \widehat{E}_\Diamond$.
        The probability and payoff of each edge is preserved from $\mathcal{M}$.
    $\mathcal{M}_e \leftarrow$ For all $v \in T$, identify $\tilde{v}$ with v in $\mathcal{M}'_e$
    return true iff $\hat{s}$ is winning for sure-DirFWMP(ℓ, α) in $\mathcal{M}_e$

C Discussion

In the context of reactive synthesis, beyond worst-case synthesis has received considerable attention since the last decade. In this paper, we continued this study for window mean-payoff objectives which strengthen the classical mean-payoff objective by removing the drawback that there may exist arbitrarily long windows with suboptimal mean payoff. Our algorithms use techniques that are significantly different from existing studies on beyond worst-case synthesis and our results are positive in the sense that the complexities of solving these problems are no more than that of solving the corresponding sure satisfaction problem and the almost-sure satisfaction problem separately.

Novelty. The results presented in this paper are novel in the following ways:

- In [10], it is shown that the winning region for sure-$\mathsf{FWMP}(\ell, \alpha)$ can be computed by repeatedly finding winning regions for sure-$\mathsf{DirFWMP}(\ell, \alpha)$ and sure-attractors to sets. We carefully extend this approach in Algorithm 1, where we find the winning region for sure-$\mathsf{FWMP}(\ell, \alpha)$-almost-sure-$\mathsf{FWMP}(\ell, \beta)$ by computing the winning regions for sure-$\mathsf{DirFWMP}(\ell, \beta)$, for sure-$\mathsf{DirFWMP}(\ell, \alpha)$-almost-sure-$\mathsf{Büchi}(P)$, and sure-attractors to various sets.
- In Algorithm 2, we show that the techniques that allow us to reduce satisfaction of almost-sure-$\mathsf{Büchi}(T)$ to satisfaction of positive-$\mathsf{Reach}(T)$ can be lifted to give a reduction of sure-$\mathsf{DirFWMP}(\ell, \alpha)$-almost-sure-$\mathsf{Büchi}(T)$ to sure-$\mathsf{DirFWMP}(\ell, \alpha)$-positive-$\mathsf{Reach}(T)$.
- Algorithm 3 uses the `IsGoodEdge` procedure to reduce the problem of finding the winning region for sure-$\mathsf{DirFWMP}(\ell, \alpha)$-positive-$\mathsf{Reach}(T)$ in an MDP $\mathcal{M}$ to finding the winning region for sure-$\mathsf{DirFWMP}(\ell, \alpha)$ in MDPs $\mathcal{M}_e$ intricately created from $\mathcal{M}$ for edges e in $\mathcal{M}$. The MDPs $\mathcal{M}_e$ are designed such that $\mathcal{P}_{\bigcirc}$ is forced to choose edges that with positive probability take the token closer to the target T, allowing us to drop the reachability requirement. This technique has not appeared elsewhere in the literature to the best of our knowledge.

Randomized strategies. We have only considered deterministic strategies for $\mathcal{P}_{\bigcirc}$ in this work since for both sure-almost-sure satisfaction and sure-limit-sure satisfaction, randomized strategies neither give $\mathcal{P}_{\bigcirc}$ any additional power nor do they allow for winning strategies of smaller memory size.

For the sure-almost-sure satisfaction problem, Examples 9 and 12 show that, in general, the memory requirement does not decrease even if we allow randomization for $\mathsf{FWMP}(\ell)$ and BWMP objectives respectively. This is because the token must eventually always take the correct sequence of $|V|$ vertices for sure-almost-sure satisfaction of $\mathsf{FWMP}(\ell)$ and the correct sequence of $|V| \cdot w_{\max}$ edges for the sure-almost-sure satisfaction of BWMP.

Randomization does not help in the case of sure-limit-sure satisfaction either. A winning strategy for sure-limit-sure satisfaction must surely eventually switch from the memoryless almost-sure attractor strategy to $W_{\mathsf{S}\beta}$ to a strategy that is winning for $\mathsf{FWMP}(\ell, \beta)$ (if $W_{\mathsf{S}\beta}$ has been reached in the first N steps) or for $\mathsf{FWMP}(\ell, \alpha)$ (if $W_{\mathsf{S}\beta}$ has not been reached after N steps). The winning strategy must follow the almost-sure attractor strategy to $W_{\mathsf{S}\beta}$ for sufficiently many turns to ensure that $\mathsf{FWMP}(\ell, \beta)$ is satisfied with a high probability, and this requires a counter of the given size.

Future work. Some interesting problems to study include the combination of satisfying a window mean payoff α surely (resp. almost surely) and a window mean payoff β with some threshold probability p, or a combination of sure, almost sure, and threshold probability satisfaction, or even a combination of satisfaction with different threshold probabilities, that is, α_1 with probability p_1 and α_2 with probability p_2. Another avenue to extend this work is study combinations of winning conditions with different window lengths, for instance, satisfaction of sure-$\mathsf{FWMP}(\ell_1, \alpha)$-almost-sure-$\mathsf{FWMP}(\ell_2, \beta)$ for $\ell_1 \neq \ell_2$. It would also be interesting to see if some of these techniques can be used for the study of other finitary objectives like finitary parity or Streett conditions. Finally, various combinations of sure, almost-sure, and threshold probability satisfaction can also be studied for classical parity (some of which exist in [5] and [3]) and classical mean-payoff objectives.