

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 quantitative 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 the average payoff of a finite window that slides over an infinite run be greater than a given threshold. 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 P for the fixed variant (if ℓ is given in unary) and are both in $\text{NP} \cap \text{coNP}$ 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 Probability and statistics $\rightarrow$ Stochastic processes

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

Funding We acknowledge support of the Department of Atomic Energy, Government of India, under project no. RTI4014.

Shibashis Guha: Partially supported by CEFIPRA project no. 7302-I.

1 Introduction

Beyond worst-case synthesis. Classical two-player zero-sum games [1, 22] 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) [24] 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 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 [9] 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 [20, 25], a play may satisfy the objective while still exhibiting undesired behaviours over arbitrarily long finite segments [11, 13]. 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* [11], 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 $\text{FWMP}(\ell, \lambda)$ 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 $\text{BWMP}(\lambda)$ holds for a play if there exists some $\ell \geq 1$ for which $\text{FWMP}(\ell, \lambda)$ holds for the play.

Related work. The beyond worst-case framework under expectation and worst-case semantics was introduced in [9] for quantitative objectives. The problem where the players are restricted to finite-memory strategies was shown to be in $\text{NP} \cap \text{coNP}$ for the mean-payoff objective. The case of infinite memory strategy for the BWC synthesis problem was left open in [9] and was solved in [16]. Further, in [16], 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 [14]. 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 $\text{NP} \cap \text{coNP}$; in [15, 19], 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 without stochasticity [20, 25], and finitary versions were introduced as window mean-payoff objectives [11] and have been studied extensively since the last decade. For finitary mean-payoff objectives, the satisfaction problem [8] and the expectation problem [7] were studied in MDPs. Both problems were shown to be in PTIME for the $\text{FWMP}(\ell)$ objective and in $\text{NP} \cap \text{coNP}$ for the BWMP objective. The satisfaction problem and the expectation problem for window mean-payoff objectives have recently been studied in [18] and [17] respectively for the more general setting that are stochastic games. In [21], BWC synthesis with the expectation semantics has been studied

■ **Table 1** Previous and our results (shaded) for window mean-payoff satisfaction in MDPs

Winning condition		Complexity	Memory bound	
			lower	upper
FWMP(ℓ)	sure [11]	PTIME	$\ell - 1$	ℓ
	almost-sure [8]	PTIME	$\ell - 1$	ℓ
	SAS (Theorem 19)	PTIME	$\Omega(\max\{ V , \ell\})$	$\mathcal{O}(V \cdot \ell)$
	SLS (Theorem 29)	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 [11]	$\text{NP} \cap \text{coNP}$	memoryless	memoryless
	almost-sure [8]	$\text{NP} \cap \text{coNP}$	memoryless	memoryless
	SAS (Theorem 27)	$\text{NP} \cap \text{coNP}$	$\Omega(V \cdot w_{\max})$	$\mathcal{O}(V ^3 \cdot w_{\max})$
	SLS (Theorem 30)	$\text{NP} \cap \text{coNP}$	$\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{ V }}\right)$	$\mathcal{O}\left(\frac{ V \cdot \log(1/\varepsilon)}{(\mathbb{P}_{\min})^{ V }}\right)$

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) a window mean-payoff at least α surely, i.e. against all strategies of an adversarial environment, and (ii) a window mean-payoff at least 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) a window mean-payoff at least α surely, and (ii) a window mean-payoff at least β with probability $1 - \varepsilon$.

While almost-sure satisfaction and limit-sure satisfaction are equivalent in MDPs [10], we show that sure-almost-sure satisfaction is strictly stronger than sure-limit-sure satisfaction (Example 3). We show that for both FWMP(ℓ) and BWMP, the SAS and SLS problems are no more complex than solving sure satisfaction [11] or almost-sure satisfaction [8] of the same objective separately. We show in Theorems 19 and 29 that the above problems are in PTIME for FWMP(ℓ) if ℓ is given in unary, and in Theorems 27 and 30 that the problems are in $\text{NP} \cap \text{coNP}$ 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 18 and 26). Our results are summarized in Table 1.

For the sure-almost-sure satisfaction of quantitative objectives such as mean payoff [9] and parity [5], the following observations are useful: there is a winning strategy for sure satisfaction (that may invalidate almost-sure satisfaction) and a winning strategy for almost-sure satisfaction (that may invalidate sure satisfaction), and in order to satisfy sure-almost-sure satisfaction, a strategy may need to switch between the two infinitely often, ensuring in the long run both sure and almost-sure satisfaction. This however does not apply to sure-almost-sure synthesis for finitary objectives, as we have a bounded amount of turns to satisfy both objectives. We instead require different techniques and a more careful analysis.

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. Finally, we conclude with a brief discussion in Section 6.

2 Preliminaries

Probability distributions. A *probability distribution* over a finite set A is a function $\Pr: A \rightarrow [0, 1]$ such that $\sum_{a \in A} \Pr(a) = 1$. We denote by $\mathcal{D}(A)$ the set of all probability distributions over A . The *support* of the probability distribution $\Pr$ on A is $\text{Supp}(\Pr) = \{a \in A \mid \Pr(a) > 0\}$. For algorithm and complexity reasons, we assume that probability distributions take rational values. Probability distributions may also be defined over an infinite set A , see e.g. [6].

Markov decision processes. A finite *Markov decision process* (MDP) is a tuple $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, 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.
- $(V_\circ, V_\diamond)$ is a partition of V . The vertices in $V_\circ$ belong to the system or Player $\circ$ (denoted $\mathcal{P}_\circ$), and the vertices in $V_\diamond$ belong to the environment and are called *probabilistic vertices*.
- $\mathbb{P}: V_\diamond \rightarrow \mathcal{D}(V)$ 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, $\min\{\mathbb{P}(v)(v') \mid v, v' \in V, v' \in N^+(v)\}$.
- $w: E \rightarrow \mathbb{Q}$ 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 $v_i \in V_\circ$, then $\mathcal{P}_\circ$ chooses an out-neighbour $v_{i+1} \in N^+(v_i)$; otherwise $v_i \in V_\diamond$, and an out-neighbour $v_{i+1} \in N^+(v_i)$ is chosen with probability $\mathbb{P}(v_i)(v_{i+1})$. Player $\mathcal{P}_\circ$ 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* $\pi = v_0 v_1 v_2 \dots \in V^\omega$ 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* $v_i v_{i+1} \dots v_j$ of π . Its length is $|\pi(i, j)| = j - i$, the number of edges. We denote by $\pi(0, j)$ the finite *prefix* $v_0 v_1 \dots v_j$ of π , and by $\pi(i, \infty)$ the infinite *suffix* $v_i v_{i+1} \dots$ of π . We denote by $\text{Plays}_{\mathcal{M}}$ and $\text{Pref}_{\mathcal{M}}$ the set of all plays and the set of all finite prefixes in $\mathcal{M}$ respectively. We denote by $\text{Last}(\rho)$ the last vertex of the prefix $\rho \in \text{Pref}_{\mathcal{M}}$. We denote by $\text{Pref}_{\mathcal{M}}^i$ ($i \in \{\circ, \diamond\}$) the set of all prefixes ρ such that $\text{Last}(\rho) \in V_\circ$. Figure 1 shows an example of an MDP; vertices of $\mathcal{P}_\circ$ are shown as circles and probabilistic vertices as diamonds. The payoff for each edge is shown in **red** and probability distribution over out-neighbours of probabilistic vertices is shown in **blue**.

Objectives. An objective φ is a Borel-measurable subset of $\text{Plays}_{\mathcal{M}}$ [23]. A play $\pi \in \text{Plays}_{\mathcal{M}}$ *satisfies* an objective φ if $\pi \in \varphi$. For a target set $T \subseteq V$, we define some objectives:

- *reachability*, denoted $\text{Reach}(T) = \{\pi = v_0 v_1 v_2 \dots \mid \exists i \geq 0, v_i \in T\}$, is the set of all plays that visit a vertex in T ; we also define some variants of $\text{Reach}(T)$:

- For a subset $F \subseteq E$ of edges, *edge-restricted reachability*, denoted $\text{Reach}_F(T) = \{\pi = v_0v_1v_2 \cdots \mid \exists i \geq 0, \forall j \geq 0, v_i \in T \wedge (j < i \implies (v_j, v_{j+1}) \in F)\}$, is the set of all plays that visit a vertex in T and use only edges in F until the first visit to T ;
- For an integer $k \geq 0$, *bounded reachability*, denoted $\text{Reach}_{\leq k}(T) = \{\pi = v_0v_1v_2 \cdots \mid \exists i \geq 0, i \leq k, v_i \in T\}$, is the set of all plays that visit a vertex in T in at most k steps from the start of the play;
- *safety*, denoted $\text{Safe}(T) = \{\pi = v_0v_1v_2 \cdots \mid \forall i \geq 0, v_i \in T\}$, is the set of all plays that never visit a vertex not in T ;
- *Büchi*, denoted $\text{Büchi}(T) = \{\pi = v_0v_1v_2 \cdots \mid \forall i \geq 0, \exists j \geq i, v_j \in T\}$, is the set of all plays that visit T infinitely often; and
- *coBüchi*, denoted $\text{coBüchi}(T) = \{\pi = v_0v_1v_2 \mid \exists i \geq 0, \forall j \geq i, v_j \in T\}$, the set of all plays that eventually visit only vertices in T .

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

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

- *total payoff*, denoted $\text{TP}(\lambda) = \{\pi = v_0v_1v_2 \cdots \mid \limsup_{n \rightarrow \infty} \sum_{i=0}^{n-1} w(v_i, v_{i+1}) \geq \lambda\}$, is the set of all plays with (supremum) total payoff at least λ ; and
- *mean payoff*, denoted $\text{MP}(\lambda) = \{\pi = v_0v_1v_2 \cdots \mid \limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{i=0}^{n-1} w(v_i, v_{i+1}) \geq \lambda\}$, 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 $\rho \cdot \pi \in \text{Plays}_{\mathcal{M}}$), we have that $\rho \cdot \pi$ satisfies φ . An objective φ is *prefix-independent* if it is closed under both prefixes and suffixes. 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 objective $\text{Safe}(T)$ is closed under suffixes, $\text{Reach}(T)$ is closed under prefixes, and $\text{Büchi}(T)$, $\text{coBüchi}(T)$, and $\text{MP}(\lambda)$ are prefix-independent objectives (closed under both suffixes and prefixes). $\text{Reach}_F(T)$, $\text{Reach}_{\leq k}(T)$, and $\text{TP}(\lambda)$ are closed under neither suffixes nor prefixes.

Strategies. A (deterministic) *strategy* for $\mathcal{P}_{\bigcirc}$ in an MDP $\mathcal{M}$ is a function $\sigma: \text{Pref}_{\mathcal{M}}^{\bigcirc} \rightarrow V$ that maps prefixes ρ with $\text{Last}(\rho) \in V_{\bigcirc}$ to an out-neighbour of $\text{Last}(\rho)$. 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. If $v \in V_{\bigcirc}$, then the Mealy machine outputs an out-neighbour of v that $\mathcal{P}_{\bigcirc}$ must choose, and otherwise if $v \in V_{\diamond}$, then the Mealy machine outputs an empty string.

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 $\rho, \rho' \in \text{Pref}_{\mathcal{M}}^{\bigcirc}$ such that $\text{Last}(\rho) = \text{Last}(\rho')$, we have $\sigma(\rho) = \sigma(\rho')$. Memoryless strategies can be defined by Mealy

machines with only one state. A play $\pi = v_0 v_1 \dots$ is an *outcome* of a strategy σ if for all $j \geq 0$ with $v_j \in V_\circ$, 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 $\Pr_{\mathcal{M},v}^\sigma(\cdot)$ over plays in $\mathcal{M}$ (see e.g. [2]). For an objective φ , we have that $\Pr_{\mathcal{M},v}^\sigma(\varphi)$ is the probability that an outcome of σ from v in $\mathcal{M}$ satisfies φ .

Winning conditions. A strategy σ of $\mathcal{P}_\circ$ from v in $\mathcal{M}$ is *winning for positive- φ* if $\Pr_{\mathcal{M},v}^\sigma(\varphi) > 0$, *winning for probability(p)- φ* if $\Pr_{\mathcal{M},v}^\sigma(\varphi) \geq p$, *winning for almost-sure- φ* if $\Pr_{\mathcal{M},v}^\sigma(\varphi) = 1$, and *winning for sure- φ* 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 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-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 $\text{SureAttr}(T)$.

Two-player games. When computing the winning region for sure- φ in an MDP $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, it is useful to view $\mathcal{M}$ as a *two-player game* played by $\mathcal{P}_\circ$ 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 $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, a subset $V' \subseteq V$ of vertices induces a *subMDP* of $\mathcal{M}$ if for all vertices $v' \in V'$, player $\mathcal{P}_\circ$ 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-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 $N^+(v') \cap V' \neq \emptyset$, 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, $\mathcal{M}[V'] = ((V', E'), (V_\circ \cap V', V_\diamond \cap V'), \mathbb{P}', w')$, 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 region for sure- φ_{suf} in $\mathcal{M}$ induces a subMDP of $\mathcal{M}$.*

Proof. We show that the winning region for sure- φ_{suf} in $\mathcal{M}$ induces a subMDP in $\mathcal{M}$. Let W_S denote this winning region. We show that both Conditions (i) and (ii) hold for W_S .

- *Condition (i):* Suppose towards a contradiction that Condition (i) does not hold for W_S , that is, there exists a vertex $v \in W_\text{S}$ such that v does not have an out-neighbour in W_S . Since v is winning for sure- φ_{suf} , there exists a strategy σ from v that is winning for sure- φ_{suf} . In particular, if $v \in V_\circ$, then $\mathcal{P}_\circ$ moves the token from v to an out-neighbour u of v that is $\sigma(v)$, and if $v \in V_\diamond$, then the token may move to any out-neighbour u of v . In both cases, player $\mathcal{P}_\circ$ can continue following σ from u and ensure that all outcomes of σ starting with the prefix vu satisfy φ_{suf} , and thus, we have that vu is a winning prefix for sure- φ_{suf} . Since φ_{suf} is closed under suffixes, we have that for every play π starting with vu that satisfies φ_{suf} , the play $\pi(1, \infty)$ obtained by dropping the initial vertex v

from π also satisfies φ_{suf} . Thus, if the play begins from u and $\mathcal{P}_\circ$ plays as if the play started from vu , that is, if $\mathcal{P}_\circ$ follows the strategy σ_u given by $\sigma_u(u \cdot \rho) = \sigma(v \cdot u \cdot \rho)$ for all prefixes ρ , then we have that σ_u is a winning strategy from u for $\text{sure-}\varphi_{\text{suf}}$. We get that u is winning for $\text{sure-}\varphi_{\text{suf}}$ and thus belongs to W_Σ . This is a contradiction since we had assumed that none of the out-neighbours of v belong to W_Σ .

- *Condition (ii):* Suppose towards a contradiction that there exists a probabilistic vertex $v \in W_\Sigma \cap V_\diamond$ such that v has an out-neighbour u that does not belong to W_Σ . Using similar arguments as the previous case, we have that vu is a winning prefix for $\text{sure-}\varphi_{\text{suf}}$, and since φ_{suf} is closed under suffixes, we have that u is winning for $\text{sure-}\varphi_{\text{suf}}$. Thus, the vertex u belongs to W_Σ , which is a contradiction. ◀

A similar result holds for the winning region for $\text{almost-sure-}\varphi_{\text{suf}}$ in $\mathcal{M}$.

► **Proposition 2.** *If φ_{suf} is an objective that is closed under suffixes, then the winning region for $\text{almost-sure-}\varphi_{\text{suf}}$ in $\mathcal{M}$ induces a subMDP 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 $\mathcal{M}[V']$. 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' so as to obtain a probability distribution over the out-neighbours of v' that belong to V' . Formally, $\mathcal{M}[V'] = ((V', E'), (V_\circ \cap V', V_\diamond \cap V'), \mathbb{P}', w')$, 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.

Maximal end components. An *end component* (EC) in an MDP $\mathcal{M}$ is a subset $T \subseteq V$ of vertices such that T is strongly connected and T induces a subMDP $\mathcal{M}[T]$. Thus, for every pair of vertices v, v' in the T , player $\mathcal{P}_\circ$ has a strategy to reach v' from v with probability 1 and $\mathcal{P}_\circ$ can ensure with probability 1 that the token never leaves T . A *maximal end component* (MEC) is an EC that is not contained in any other EC. The MECs in an MDP are disjoint, and thus, each vertex in an MDP belongs to either no MEC or exactly one MEC. The MEC decomposition can be computed in polynomial time [12].

3 Window mean-payoff objectives

In this section, we recall window mean-payoff objectives introduced in [11] in the context of two-player games. We see 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 [11] for arbitrary thresholds $\lambda \in \mathbb{Q}$. Given a play $\pi = v_0 v_1 \dots$, 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 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 $i < j$. 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 $\pi = v_0 v_1 \dots$ in $\mathcal{M}$ satisfies the *good window* objective $\text{GW}_{\mathcal{M}}(\ell, \lambda)$ if the λ -window that opened at v_0 in π closes in at most ℓ steps.

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

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

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

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

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

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

Bounded window length. We also consider bounded versions of $\text{DirFWMP}(\ell, \lambda)$ and $\text{FWMP}(\ell, \lambda)$. A play π satisfies the *direct bounded window mean-payoff* objective if there exists a window length $\ell \geq 1$ such that π satisfies $\text{DirFWMP}_{\mathcal{M}}(\ell, \lambda)$.

$$\text{DirBWMP}_{\mathcal{M}}(\lambda) = \{\pi \in \text{Plays}_{\mathcal{M}} \mid \exists \ell \geq 1, \pi \in \text{DirFWMP}_{\mathcal{M}}(\ell, \lambda)\}$$

Similarly, a play π satisfies the bounded window mean-payoff objective $\text{BWMP}(\lambda)$ if there exists a window length $\ell \geq 1$ such that π satisfies $\text{FWMP}_{\mathcal{M}}(\ell, \lambda)$. The $\text{BWMP}(\lambda)$ objective is also the prefix-independent version of the $\text{DirBWMP}(\lambda)$ objective.

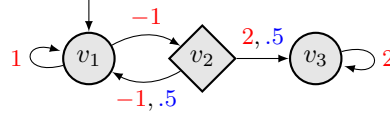
$$\begin{aligned} \text{BWMP}_{\mathcal{M}}(\lambda) &= \{\pi \in \text{Plays}_{\mathcal{M}} \mid \exists \ell \geq 1, \pi \in \text{FWMP}_{\mathcal{M}}(\ell, \lambda)\} \\ &= \{\pi \in \text{Plays}_{\mathcal{M}} \mid \exists k \geq 0, \pi(k, \infty) \in \text{DirBWMP}_{\mathcal{M}}(\lambda)\} \end{aligned}$$

Note that $\text{DirFWMP}(\ell, \lambda)$ and $\text{DirBWMP}(\lambda)$ are closed under suffixes, and $\text{FWMP}(\ell, \lambda)$ and $\text{BWMP}(\lambda)$ are prefix-independent objectives. We also note that all of these objectives strengthen the standard mean-payoff objective $\text{MP}(\lambda)$ [20]. In particular, we have the following relation between the objectives

$$\text{DirFWMP}(\ell, \lambda) \subseteq \text{FWMP}(\ell, \lambda) \subseteq \text{BWMP}(\lambda) \subseteq \text{MP}(\lambda),$$

$$\text{DirFWMP}(\ell, \lambda) \subseteq \text{DirBWMP}(\lambda) \subseteq \text{BWMP}(\lambda) \subseteq \text{MP}(\lambda),$$

and all of these inclusions are strict in general.



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

Decision problems. We consider the following decision problems that are a combination of sure and almost-sure (and limit-sure) satisfaction of FWMP(ℓ) objectives. We also study analogous decision problems for BWMP.

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 $\mathcal{P}_\circ$ such that (i) all outcomes of σ from v satisfy FWMP(ℓ, α), and (ii) with probability 1, the outcome of σ from v satisfies FWMP(ℓ, β) from v (i.e., $\Pr_{\mathcal{M},v}^\sigma(\text{FWMP}(\ell, \beta)) = 1$)? 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 $\mathcal{P}_\circ$ such that (i) all outcomes of σ_ε from v satisfy FWMP(ℓ, α), and (ii) with probability at least $1 - \varepsilon$, the outcome of σ_ε from v satisfies FWMP(ℓ, β) from v (i.e., $\Pr_{\mathcal{M},v}^{\sigma_\varepsilon}(\text{FWMP}(\ell, \beta)) \geq 1 - \varepsilon$)? 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-FWMP(ℓ, α).

► **Example 3.** While almost-sure satisfaction and limit-sure satisfaction of FWMP(ℓ, α) are the same in MDPs [10], we illustrate using the MDP in Figure 1 that sure-almost-sure satisfaction is strictly stronger than sure-limit-sure satisfaction for FWMP(ℓ). We have an MDP where the vertex v_1 is winning for sure-FWMP(2, 1)-limit-sure-FWMP(2, 2) but not for sure-FWMP(2, 1)-almost-sure-FWMP(2, 2).

To see that sure-FWMP(2, 1) and almost-sure-FWMP(2, 2) cannot be satisfied simultaneously from v_1 : in order to satisfy almost-sure-FWMP(2, 2), 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 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 $v_1 v_2 v_1$ appears infinitely often, and each such occurrence $v_1 v_2 v_1$ 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, 2). $\lrcorner$

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 $\mathcal{P}_\circ$ can satisfy sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β). We

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

In: An MDP $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, 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:  $W_{S_\alpha} \leftarrow \text{SureFWMP}(\mathcal{M}, \ell, \alpha)$ 
2:  $W_{S_\beta} \leftarrow \text{SureFWMP}(\mathcal{M}[W_{S_\alpha}], \ell, \beta)$ 
3:  $W \leftarrow W_{S_\beta}$ 
4: repeat
5:    $P \leftarrow X_{>0}^{\leq 1}(\mathcal{M}[W_{S_\alpha}], W)$ 
6:    $W_{\text{sdab}} \leftarrow \text{SureDirFWMPASBüchi}(\mathcal{M}[W_{S_\alpha} \setminus W], \ell, \alpha, P)$ 
7:    $W \leftarrow W \cup W_{\text{sdab}}$ 
8:    $W \leftarrow \text{SureAttr}(\mathcal{M}[W_{S_\alpha}], W)$ 
9: until  $W_{\text{sdab}} = \emptyset$ 
10: return  $W$ 

```

note that the winning region for the combination of winning conditions is not merely the intersection of the winning regions for sure-FWMP(ℓ, α) and for almost-sure-FWMP(ℓ, β): in the MDP in Example 3, we have that every vertex is winning for sure-FWMP(ℓ, α) and for almost-sure-FWMP(ℓ, β) separately, and yet there exist vertices v_1 and v_2 that are not winning for sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β). 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.

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_{S_α} that is the winning region for $\mathcal{P}_\circ$ for sure-FWMP(ℓ, α) in $\mathcal{M}$ by viewing the probabilistic vertices of $\mathcal{M}$ as adversarial, and then using [11, Algorithm 1] that finds the winning region for $\mathcal{P}_\circ$ for FWMP(ℓ, α) in two-player games. If $\mathcal{P}_\circ$ cannot satisfy sure-FWMP(ℓ, α) from a vertex v in $\mathcal{M}$, then $\mathcal{P}_\circ$ 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 $V \setminus W_{S_\alpha}$. Moreover, since FWMP(ℓ, α) is prefix-independent (and thus also closed under suffixes), it follows from Proposition 1 that W_{S_α} induces a subMDP $\mathcal{M}[W_{S_\alpha}]$ in which $\mathcal{P}_\circ$ 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_{S_β} that is the winning region for sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S_\alpha}]$, and in Line 3, we grow W by including in W all vertices that belong to W_{S_β} . 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 $\mathcal{M}[W_{S_\alpha}]$ from which the token reaches W in one step with positive probability but not surely. In other words, P is the set of all vertices in $(W_{S_\alpha} \cap V_\diamond) \setminus W$ with at least one out-neighbour in W and at least one out-neighbour not in W . We denote this by $X_{>0}^{\leq 1}$ in the algorithm.
- We note that every vertex v in $W_{S_\alpha} \setminus W$ has at least one out-neighbour in $W_{S_\alpha} \setminus W$ because otherwise $\mathcal{P}_\circ$ could ensure that the token reaches W surely from v , in which case v should be included in W . Thus, the set $W_{S_\alpha} \setminus W$ induces a subMDP closure of $\mathcal{M}[W_{S_\alpha}]$. Since this is the same as the subMDP closure of $\mathcal{M}$, we denote this by $\mathcal{M}[W_{S_\alpha} \setminus W]$. Suppose there exists a vertex v in $\mathcal{M}[W_{S_\alpha} \setminus W]$ from which $\mathcal{P}_\circ$ 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 $\mathcal{M}[W_{S_\alpha}]$, 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 $\mathcal{M}[W_{S_\alpha} \setminus W]$, 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)).

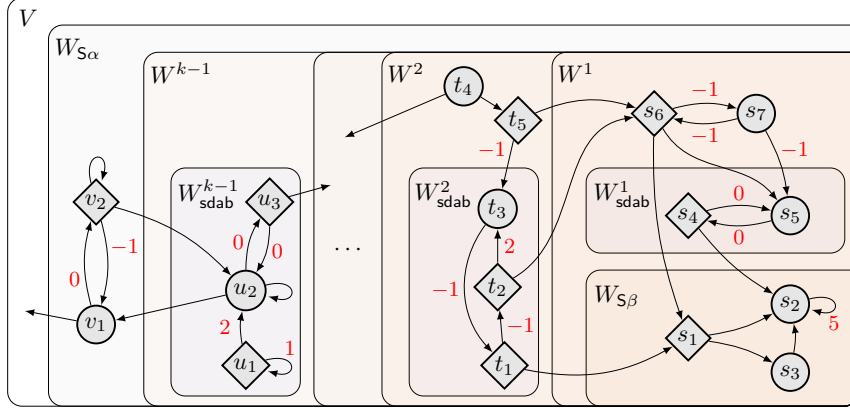
- Finally, we compute **SureAttr**(W), the sure-attractor of W , that is the set of all vertices in W_{S_α} from which $\mathcal{P}_\circ$ can surely reach W . We note that if $\mathcal{P}_\circ$ can satisfy sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) from all vertices in W , then $\mathcal{P}_\circ$ can 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 $\mathcal{P}_\circ$ can ensure that every outcome from v reaches a vertex in W , following which $\mathcal{P}_\circ$ 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).

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_{sdab}^i , 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_{S_β} 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_{sdab}^i are disjoint, since W_{sdab}^i is the winning region for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(P^i) in $\mathcal{M}[W_{S_\alpha} \setminus W^{i-1}]$.
- For $1 \leq i \leq k$, if W_{sdab}^i is non-empty, then P^i and W_{sdab}^i 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_{sdab}^i also belongs to W_{sdab}^i .
- For $1 \leq i \leq k$, we have that $W^{i-1} \cup W_{\text{sdab}}^i \subseteq W^i$ since W^i is obtained by computing the sure-attractor of $W^{i-1} \cup W_{\text{sdab}}^i$. Let us denote by A^i the set $W^i \setminus (W^{i-1} \cup W_{\text{sdab}}^i)$, that is the set of all vertices in W^i but not in $W^{i-1} \cup W_{\text{sdab}}^i$ from which $\mathcal{P}_\circ$ has a sure-attractor strategy to reach $W^{i-1} \cup W_{\text{sdab}}^i$. Thus, we have for all $1 \leq i \leq k$ that W^{i-1} , W_{sdab}^i , and A^i form a partition of W^i . Moreover, the entire set W can be divided into pairwise disjoint sets W^0 , W_{sdab}^1 , A^1 , W_{sdab}^2 , A^2 , $\dots$, W_{sdab}^k , and A^k .
- Since the algorithm terminates in the k^{th} iteration, we have that $W_{\text{sdab}}^k = \emptyset$ and thus also $A^k = \emptyset$. 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 $W^{j-1} \setminus W^{i-1}$. 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 $V \setminus W^{j-1}$.

► **Example 4.** We show some features of Algorithm 1 in Figure 2 for window length $\ell = 3$ and for thresholds $\alpha = 0$ and $\beta = 5$. The set W_{S_β} (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



■ **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.

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 $W_{S\alpha} \setminus W^0$, we compute W_{sdab}^1 , 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 $W_{S\alpha} \setminus (W_{sdab}^1 \cup W^0)$ from which $\mathcal{P}_\circ$ can ensure that the token reaches $W_{sdab}^1 \cup W^0$ surely is $\{s_6, s_7\}$. We come to the end of the first iteration of the **repeat** loop and we have that $W^1 = A^1 \cup W_{sdab}^1 \cup W^0 = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$. Since W_{sdab}^1 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 $\mathcal{M}[W_{S\alpha} \setminus W^1]$ is $W_{sdab}^2 = \{t_1, t_2, t_3\}$, 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 $W_{sdab}^k = \emptyset$. Thus, the algorithm terminates here and returns W^k . $\square$

Proof of correctness of Algorithm 1. We prove some intermediate lemmas before we prove the correctness of Algorithm 1 in Theorem 9. In Lemma 5 and Lemma 6, we show that starting from every vertex in $W_{S\alpha} \setminus W^k$ (where W^k is the set W at the end of the algorithm), player $\mathcal{P}_\circ$ cannot satisfy sure-DirFWMP(ℓ, α)-almost-sure-Reach(W^k) or sure-FWMP(ℓ, α)-almost-sure-Reach(W^k) respectively. Thus, starting from $W_{S\alpha} \setminus W^k$, if $\mathcal{P}_\circ$ wants to satisfy sure-DirFWMP(ℓ, α) or sure-FWMP(ℓ, α), then the token remains in $W_{S\alpha} \setminus W^k$ with some positive probability. Then, Lemma 7 and Lemma 8 together imply that if the token remains in $W_{S\alpha} \setminus W^k$, then sure-FWMP(ℓ, α)-almost-sure-FWMP(ℓ, β) cannot be satisfied.

► **Lemma 5.** *For all vertices $v \in W_{S\alpha} \setminus W^k$ in the MDP $\mathcal{M}$, the vertex v is not winning for sure-DirFWMP(ℓ, α)-almost-sure-Reach(W) in $\mathcal{M}$.*

Proof. For each vertex $v \in (W_{S\alpha} \setminus W^k) \cap V_\circ$ belonging to $\mathcal{P}_\circ$, there is no out-neighbour of v in W^k , because otherwise v would belong to the sure-attractor of W^k , and hence, also to W^k (Line 8), which is a contradiction since we started with $v \in W_{S\alpha} \setminus W^k$. A similar argument gives us that for all probabilistic vertices $v \in (W_{S\alpha} \setminus W^k) \cap V_\diamond$, it is not the case that all out-neighbours of v belong to W^k , because that would imply that v belongs to the sure-attractor of W^k , and hence, also to W^k , which is a contradiction. Thus, if v is a vertex in $W_{S\alpha} \setminus W^k$ with an out-neighbour in W^k , then it follows that v is a probabilistic vertex and that v has at least one out-neighbour in $W_{S\alpha} \setminus W^k$. Note that the set of all such vertices

is precisely the set P^k computed on Line 5.

Thus, starting from a vertex $v \in W_{S_\alpha} \setminus W^k$, if the token reaches W^k , then token reaches W^k through a vertex in P^k . Moreover, starting from v , the token can almost-surely reach W^k in $\mathcal{M}[W_{S_\alpha}]$ if and only if starting from v , the token can almost-surely visit P^k infinitely often in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$. Further, the token can almost-surely reach W^k from v while also satisfying $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}[W_{S_\alpha}]$ if and only if starting from v , the token can almost-surely visit P^k infinitely often while also satisfying $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$. Since at the end of Algorithm 1, the set W_{sdab}^k that is the winning region for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(P^k) is empty, we get that no vertex in $W_{S_\alpha} \setminus W^k$ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Reach(W^k) in $\mathcal{M}[W_{S_\alpha}]$. ◀

► **Lemma 6.** *For all vertices $v \in W_{S_\alpha} \setminus W^k$ in the MDP $\mathcal{M}$, the vertex v is not winning for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-Reach(W^k) in $\mathcal{M}$.*

Proof. Observe that a vertex $v \in W_{S_\alpha} \setminus W^k$ is winning for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-Reach(W^k) in $\mathcal{M}[W_{S_\alpha}]$ if and only if v is winning for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-Büchi(P^k) in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$. From Lemma 5, we have that no vertex is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(P^k) in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$. That is, for every vertex $v \in W_{S_\alpha} \setminus W^k$, for every strategy σ of $\mathcal{P}_\circ$ from v that is winning for almost-sure-Büchi(P^k), there exists an outcome of this strategy that contains an open α -window of length ℓ . Since this is true for every vertex in $W_{S_\alpha} \setminus W^k$, it is also the case that there exists an outcome of σ with infinitely many open α -windows of length ℓ . Thus, no vertex in $W_{S_\alpha} \setminus W^k$ is winning for $\mathcal{P}_\circ$ for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-Reach(W^k) in $\mathcal{M}[W_{S_\alpha}]$. ◀

► **Lemma 7.** *The winning region for $\text{sure-FWMP}(\ell, \beta)$ in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$ is empty.*

Proof. Suppose towards a contradiction that there exists a vertex $v \in W_{S_\alpha} \setminus W^k$ that is winning for $\text{sure-FWMP}(\ell, \beta)$ in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$. Let σ be a strategy of $\mathcal{P}_\circ$ that is winning for $\text{sure-FWMP}(\ell, \beta)$ from v in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$. The key observation is that σ is also a winning strategy for $\text{sure-FWMP}(\ell, \beta)$ from v in the larger MDP $\mathcal{M}[W_{S_\alpha}]$, and thus v belongs to W_{S_β} , and also to W^k . This is a contradiction since we started with $v \in W_{S_\alpha} \setminus W^k$. ◀

► **Lemma 8.** *The winning region for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$ is empty.*

Proof. From Lemma 7, we have that the winning region for $\text{sure-FWMP}(\ell, \beta)$ in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$ is empty. From [18, Lemma 6.1], we have that if the winning region for $\text{sure-FWMP}(\ell, \beta)$ in an MDP is empty, then the winning region for almost-sure-FWMP(ℓ, β) in the MDP is also empty. Thus, we have that the winning region for almost-sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$ is empty. It follows that $\mathcal{P}_\circ$ cannot satisfy $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β) from any vertex in $\mathcal{M}[W_{S_\alpha} \setminus W^k]$. ◀

The following theorem shows the correctness of Algorithm 1.

► **Theorem 9.** *Algorithm 1 computes the winning region for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β) in $\mathcal{M}$.*

Proof. ($\Rightarrow$) We show that starting from an arbitrary vertex $v \in W^k$, player $\mathcal{P}_\circ$ has a winning strategy for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β). We consider the cases $v \in W^0$ and $v \in W^k \setminus W^0$ separately:

- Suppose the token is on a vertex v in W^0 (that is, $W_{S\beta}$, the winning region for $\mathcal{P}_\circ$ for $\text{sure-FWMP}(\ell, \beta)$). Since $\text{FWMP}(\ell, \beta)$ is a prefix-independent objective, we have from Proposition 1 that W^0 induces a subMDP of $\mathcal{M}$. Thus, starting from v , player $\mathcal{P}_\circ$ has a strategy that is winning for $\text{sure-FWMP}(\ell, \beta)$ and also keeps the token in W^0 for all outcomes of the strategy. Since $\alpha < \beta$, we have that satisfaction of $\text{sure-FWMP}(\ell, \beta)$ implies satisfaction of $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β), and thus v is winning for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β).
- Otherwise, suppose the token is on a vertex v in $W^k \setminus W^0$. In particular, there exists $1 \leq i < k$ such that $v \in W^i \setminus W^{i-1}$. Since each vertex in $W^i \setminus W^{i-1}$ belongs to either A^i and W_{sdab}^i , we have the following cases:
 - The token is on a vertex v in A^i . Then $\mathcal{P}_\circ$ follows a sure-attractor strategy from v to surely reach a vertex in W_{sdab}^i or a vertex v' in W^{i-1} .
 - The token is on a vertex v in W_{sdab}^i . Then, player $\mathcal{P}_\circ$ follows a strategy from v that is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(P^i) to keep visiting vertices in P^i while surely always closing all α -windows in at most ℓ steps. The token either eventually reaches a vertex v' in W^{i-1} (which happens with probability 1) or remains in W_{sdab}^i forever (which happens with probability 0).

If the token reaches a vertex v' in W^{i-1} , then the vertex v' is either in W^0 or in $W^{i'} \setminus W^{i'-1}$ for some $1 \leq i' \leq i-1$, and we can repeat the above argument once again.

Repeating this sufficiently many times (at most k times), we get that from every vertex in W^k , player $\mathcal{P}_\circ$ has a strategy such that with probability 1 the token eventually reaches W^0 where the $\text{FWMP}(\ell, \beta)$ objective is surely satisfied, and with probability 0 the token is stuck in W_{sdab}^i for some $1 \leq i \leq k$ where surely all α -windows close in at most ℓ steps. Thus, from all vertices in W^k , player $\mathcal{P}_\circ$ has a strategy to satisfy $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β).

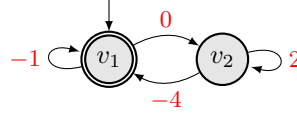
($\Leftarrow$) Now, we show the converse, that is, if a vertex v does not belong to W^k , then v is not winning for $\mathcal{P}_\circ$ for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β). If $v \notin W^k$, then we have two cases: either $v \in V \setminus W_{S\alpha}$ or $v \in W_{S\alpha} \setminus W^k$.

- If $v \in V \setminus W_{S\alpha}$, then v is not winning for $\text{sure-FWMP}(\ell, \alpha)$, and thus v is also not winning for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β).
- Otherwise, we have that $v \in W_{S\alpha} \setminus W^k$. From Lemma 6, we have that v is not winning for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-Reach(W^k). Thus, for all strategies σ of $\mathcal{P}_\circ$ that are winning for $\text{sure-FWMP}(\ell, \alpha)$, there exists $0 < p < 1$ such that with probability at least p , the outcome of σ from v remains in $W_{S\alpha} \setminus W^k$ forever. Further, from Lemma 8, it follows that with some positive probability, the outcome does not satisfy $\text{FWMP}(\ell, \beta)$. Thus, $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β) cannot be satisfied by $\mathcal{P}_\circ$ from v .

We have shown the converse direction as well, that is, if $v \notin W^k$, then v is not winning for $\mathcal{P}_\circ$ for $\text{sure-FWMP}(\ell, \alpha)$ -almost-sure-FWMP(ℓ, β). $\blacktriangleleft$

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 $\text{sure-DirFWMP}(\ell, \alpha)$ -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 $\text{sure-DirFWMP}(\ell, \alpha)$ and almost-sure-Büchi(T) separately, and yet no vertex in the MDP is winning for the combination $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(T). We show such an MDP in Figure 3.



■ **Figure 3** Both vertices are winning for $\text{sure-DirFWMP}(2, 0)$ and for $\text{almost-sure-Büchi}(\{v_1\})$ (and even $\text{sure-Büchi}(\{v_1\})$), but neither vertex is winning for $\text{sure-DirFWMP}(2, 0)\text{-almost-sure-Büchi}(\{v_1\})$.

■ **Algorithm 2** $\text{SureDirFWMPASBüchi}(\mathcal{M}, \ell, \alpha, T)$

In: An MDP $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, window length ℓ , threshold α , and target set T

Out: The winning region for $\text{sure-DirFWMP}(\ell, \alpha)\text{-almost-sure-Büchi}(T)$ in $\mathcal{M}$

```

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

```

Description of Algorithm 2. We begin by calling Algorithm 3 ($\text{SureDirFWMPPosReach}$) in Line 1 to compute W_{sdpr} , the winning region for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$. (The subscript sdpr is short for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$.) 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 $\text{sure-Safe}(W_{\text{sdpr}})$, that is the set of all vertices in $\mathcal{M}$ from which $\mathcal{P}_\circ$ can ensure for all outcomes that the token visits only vertices in W_{sdpr} . Then, in Line 6 where we recursively call Algorithm 2 ($\text{SureDirFWMPASBüchi}$) on the smaller MDP $\mathcal{M}[S]$.

Proof of correctness of Algorithm 2. We prove the correctness of Algorithm 2 in Theorem 11. Before that, we prove an intermediate result in Lemma 10.

► **Lemma 10.** *For all vertices v in an MDP $\mathcal{M}$, if $\mathcal{P}_\circ$ has a strategy to reach a target $T \subseteq V$ from v with positive probability while simultaneously satisfying $\text{sure-DirFWMP}(\ell, \alpha)$, then $\mathcal{P}_\circ$ also has a strategy to reach T from v with positive probability in at most $|V| \cdot \ell$ steps while simultaneously satisfying $\text{sure-DirFWMP}(\ell, \alpha)$.*

Proof. Since the $\text{DirFWMP}(\ell, \alpha)$ objective is closed under suffixes, we have from Proposition 1 that $\mathcal{P}_\circ$ can ensure that the token never leaves the winning region of $\text{sure-DirFWMP}(\ell, \alpha)$, and moreover, it is also necessary for $\mathcal{P}_\circ$ to keep the token within this winning region in order to satisfy $\text{sure-DirFWMP}(\ell, \alpha)$. Thus, without loss of generality, let us suppose that all vertices in the MDP are winning for $\text{sure-DirFWMP}(\ell, \alpha)$, and we only consider strategies of $\mathcal{P}_\circ$ that are winning for $\text{sure-DirFWMP}(\ell, \alpha)$.

Now, suppose for a vertex v in $\mathcal{M}$, there exists a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)$ that is also winning for $\text{positive-Reach}(T)$. Then, we want to show that there exists a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)$ following which $\mathcal{P}_\circ$ can, starting from v , with positive probability, reach T from v in at most $|V| \cdot \ell$ steps.

Suppose that this is not the case, that is, suppose that for all strategies of $\mathcal{P}_\circ$ that are winning for $\text{sure-DirFWMP}(\ell, \alpha)$ and for $\text{positive-Reach}(T)$ from v , the probability that the token reaches T in at most $|V| \cdot \ell$ steps starting from v is zero. Let $x > |V| \cdot \ell$ be the

minimum number of steps that $\mathcal{P}_\circ$ can ensure that the token reaches T from v in with positive probability. Consider one such outcome where the token reaches T from v in x steps. We also have that this outcome satisfies $\text{DirFWMP}(\ell, \alpha)$, and thus, all α -windows in this outcome close in at most ℓ steps. Moreover, from the inductive property of windows, we have that there exist infinitely many positions in this outcome where there are no open α -windows pending to be closed.

Consider the prefix $v_0 v_1 \cdots v_x$ of this outcome of length x , that is, the prefix that ends with the first visit to a vertex in T . We are interested in the positions in this prefix where there are no open α -windows. At the start of the play (that is, at v_0) there are no open α -windows. The α -window starting at v_0 closes in at most ℓ steps, say at v_{i_1} , and when this α -window closes, there are once again no open α -windows at v_{i_1} by the inductive property of windows. When the α -window starting at v_{i_1} closes, say at v_{i_2} , this is once again a position where there are no open α -windows. This way, the distance between every two consecutive positions v_{i_j} and $v_{i_{j+1}}$ with no open α -windows is at most ℓ . Since the prefix $v_0 v_1 \cdots v_x$ has at least $|V| \cdot \ell + 1$ vertices and v_0 is a position with no open α -windows, there exist at least $|V| + 1$ positions in the prefix where there are no open α -windows. Since there are only $|V|$ vertices in the game, by the pigeonhole principle, two of these positions (say v_j and v_k) have the same vertex u of the game. The following two cases are possible:

- If $u \in V_\circ$, then $\mathcal{P}_\circ$ could skip the cycle $v_j \cdots v_k$, resulting in the sequence of vertices $v_0 v_1 \cdots v_j v_{k+1} \cdots v_x$. Since $\mathcal{P}_\circ$ can ensure with positive probability that the token reaches v_x from v_k in at most $x - k$ steps, player $\mathcal{P}_\circ$ can also ensure with positive probability that the token reaches v_x from v_j in at most $x - k$ steps as both v_j and v_k are the same vertex u and at both positions, all α -windows have been closed. This gives a way for $\mathcal{P}_\circ$ to ensure that the token reaches v_x from v_0 in less than x steps, which is a contradiction, since x was assumed to be the minimum number of steps in which the token reaches T .
- If $u \in V_\diamond$, then again, consider the prefix $v_0 v_1 \cdots v_j v_{k+1} \cdots v_x$ obtained by skipping the cycle $v_j \cdots v_k$. With positive probability, the sequence $v_{k+1} \cdots v_x$ is seen immediately after v_j . Thus, using the same argument as in the previous case, player $\mathcal{P}_\circ$ can ensure with positive probability that the token reaches v_x from v_0 in less than x steps, which is a contradiction.

In both cases we get a contradiction, and thus, we have that $\mathcal{P}_\circ$ can reach T from v with positive probability in at most $|V| \cdot \ell$ steps while also satisfying $\text{sure-DirFWMP}(\ell, \alpha)$. ◀

► **Theorem 11.** *Algorithm 2 computes the winning region for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(T).*

Proof. ($\Rightarrow$) First, we show that from every vertex v returned by Algorithm 2, player $\mathcal{P}_\circ$ has a strategy that is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(T). Recall that the algorithm returns all vertices in $\mathcal{M}$ if all vertices in $\mathcal{M}$ are winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive-Reach(T), and otherwise the algorithm is called recursively on the subMDP $\mathcal{M}[S]$ where S is the winning region for $\text{sure-Safe}(W_{\text{sdp}})$ in $\mathcal{M}$, and W_{sdp} is the winning region for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive-Reach(T) in $\mathcal{M}$. Thus, if the algorithm eventually returns a non-empty set, then in the final recursive call which is called on the subMDP say $\mathcal{M}'$ of $\mathcal{M}$, we have that all vertices of $\mathcal{M}'$ belong to $W'_{\text{sdp}} = \text{SureDirFWMPPosReach}(\mathcal{M}', \ell, \alpha, T)$. For every vertex v in W'_{sdp} , since v is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive-Reach(T) in $\mathcal{M}'$, from Lemma 10, there exists a strategy σ_{sdp}^v of $\mathcal{P}_\circ$ that is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive-Reach $_{\leq |V| \cdot \ell}(T)$ from v in $\mathcal{M}'$. Using these strategies, we construct a strategy σ_{sdpab} that is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(T) from all vertices in $\mathcal{M}'$.

Description of strategy σ_{sdab} . If the play begins from a vertex v in $\mathcal{M}'$, then σ_{sdab} follows σ_{sdpr}^v for the first $|V| \cdot \ell$ steps and further continues to follow σ_{sdpr}^v until all open α -windows are closed for the first time after the $|V| \cdot \ell$ steps. At this point, if the token is at a vertex v' , then the strategy σ_{sdab} resets to $\sigma_{\text{sdpr}}^{v'}$, that is, σ_{sdab} forgets the history and follows $\sigma_{\text{sdpr}}^{v'}$ from v' for $|V| \cdot \ell$ steps before resetting again when all α -windows are closed the next time.

We claim that this strategy σ_{sdab} is winning for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T) in $\mathcal{M}$ from all vertices in $\mathcal{M}'$, and we prove this claim now. Since, for every v in $\mathcal{M}'$, the strategy σ_{sdpr}^v is winning for sure-DirFWMP(ℓ, α), we have that for every outcome of σ_{sdpr}^v , every α -window closes in at most ℓ steps. Moreover, from the inductive property of windows, for every outcome of σ_{sdpr}^v , the number of steps between every two consecutive points of the outcome where all α -windows are closed is at most ℓ . It follows that for every outcome of σ_{sdab} , it takes at most ℓ steps for all α -windows to be closed after the first $|V| \cdot \ell$ steps, at which point σ_{sdab} resets. Therefore, we have that the number of steps between consecutive resets of σ_{sdab} is greater than $|V| \cdot \ell$ and is at most $|V| \cdot \ell + \ell$. Further, since σ_{sdpr}^v is winning for positive-Reach $_{\leq |V| \cdot \ell}(T)$ in $\mathcal{M}$ for all v in $\mathcal{M}'$, we have that after every reset of σ_{sdab} , with positive probability (at least $(\mathbb{P}_{\min})^{|V| \cdot \ell}$) the token reaches T before the strategy resets again. Since the strategy resets at most every $|V| \cdot \ell + \ell$ steps and there is at least a fixed positive probability of reaching T after every reset, the probability that the token reaches T infinitely often is 1. Since σ_{sdab} only resets when all α -windows are closed, and each σ_{sdpr}^v is winning for sure-DirFWMP(ℓ, α), we have that surely all α -windows are always closed in at most ℓ steps. Thus, σ_{sdab} is a winning strategy for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T) in $\mathcal{M}$ from every vertex v in $\mathcal{M}'$.

($\Leftarrow$) Now, we prove the converse direction, that is, if a vertex v is not returned by the algorithm, then $\mathcal{P}_{\bigcirc}$ cannot simultaneously satisfy sure-DirFWMP(ℓ, α) and almost-sure-Büchi(T) from v . A vertex v is not included in the region returned by the algorithm if in some recursive call of the algorithm, the vertex v does not belong to the winning region S for sure-Safe(W_{sdpr}) (Line 6). If v does not belong to S , then for all strategies of $\mathcal{P}_{\bigcirc}$, starting from v , with positive probability the token eventually visits a vertex not in W_{sdpr} . From a vertex not in W_{sdpr} , for all strategies of $\mathcal{P}_{\bigcirc}$ that are winning for sure-DirFWMP(ℓ, α), the token does not reach the target set T with any positive probability, and thus, we also have that the token does not visit T infinitely often with any positive probability. Hence, from every vertex v not in S , if $\mathcal{P}_{\bigcirc}$ follows a strategy that is winning for sure-DirFWMP(ℓ, α), then almost-sure-Büchi(T) is not satisfied, and otherwise if $\mathcal{P}_{\bigcirc}$ follows a strategy that is not winning for sure-DirFWMP(ℓ, α), then sure-DirFWMP(ℓ, α) is not satisfied. Thus, we have that every vertex v that is not returned by the algorithm is not winning for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T) in $\mathcal{M}$. $\blacktriangleleft$

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). We note that the MDP shown in Figure 3 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 $W_{\text{SD}\alpha}$ 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 $W_{\text{SD}\alpha}$ induces a subMDP $\mathcal{M}[W_{\text{SD}\alpha}]$ of $\mathcal{M}$. We are not

■ **Algorithm 3** SureDirFWMPPosReach($\mathcal{M}, \ell, \alpha, T$)

In: An MDP $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, 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:  $W_{\text{SD}\alpha} \leftarrow \text{SureDirFWMP}(\mathcal{M}, \ell, \alpha)$ 
2:  $\mathcal{M} \leftarrow \mathcal{M}[W_{\text{SD}\alpha}]$ ,  $T \leftarrow T \cap W_{\text{SD}\alpha}$ 
3:  $R \leftarrow T$ ,  $E_g, E_b \leftarrow \emptyset$ 
4: while  $\{e = (s, t) \in E \mid s \in V \setminus T, t \in R, \text{ and } e \in E \setminus (E_g \cup E_b)\} \neq \emptyset$  do
5:   if IsGoodEdge( $e, E_g$ ) then
6:      $E_g \leftarrow E_g \cup \{e\}$ ,  $R \leftarrow R \cup \{s\}$ ,  $E_b \leftarrow \emptyset$ 
7:   else
8:      $E_b \leftarrow E_b \cup \{e\}$ 
9: return  $R$ 

10: procedure IsGoodEdge( $e = (s, t), E_g$ )
11:   Construct MDP  $\mathcal{M}'_e$  in the following way:
12:    $V \leftarrow$  the set of vertices in  $\mathcal{M}$ 
13:    $\tilde{R} \leftarrow \{\tilde{v} \mid v \in R\}$ 
14:   The set of vertices of  $\mathcal{M}'_e$  is  $V \cup \tilde{R} \cup \{\hat{s}\}$ .
   The kind of each vertex is preserved from  $\mathcal{M}$ .

15:    $E \leftarrow$  the set of edges in  $\mathcal{M}$ 
16:    $\tilde{E}_g \leftarrow \{(\tilde{u}, \tilde{v}) \mid u, v \in R, (u, v) \in E_g\}$ 
17:    $\tilde{E}_\diamond \leftarrow \{(\tilde{u}, v) \mid u \in (R \setminus T) \cap V_\diamond, v \in V, (u, v) \in E \setminus E_g\}$ 
18:    $\hat{E}_\diamond \leftarrow$  if  $s \in V_\circ$  then  $\emptyset$  else  $\{(\hat{s}, v) \mid (s, v) \in E \text{ and } v \neq t\}$ 
19:   The set of edges of  $\mathcal{M}'_e$  is  $E \cup \tilde{E}_g \cup \tilde{E}_\diamond \cup \{(\hat{s}, \tilde{t})\} \cup \hat{E}_\diamond$ .
   The probability and payoff of each edge is preserved from  $\mathcal{M}$ .

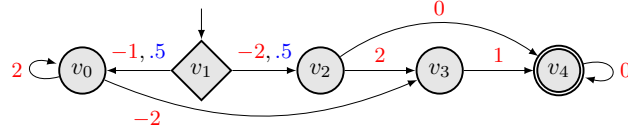
20:    $\mathcal{M}_e \leftarrow$  For all  $v \in T$ , identify  $\tilde{v}$  with  $v$  in  $\mathcal{M}'_e$ 
21:   return true iff  $\hat{s}$  is winning for sure-DirFWMP( $\ell, \alpha$ ) in  $\mathcal{M}_e$ 

```

interested in vertices in $V \setminus W_{\text{SD}\alpha}$, and thus we restrict the MDP $\mathcal{M}$ to $\mathcal{M}[W_{\text{SD}\alpha}]$ (that is, we refer to the subMDP $\mathcal{M}[W_{\text{SD}\alpha}]$ as $\mathcal{M}$) where all vertices are winning for sure-DirFWMP(ℓ, α), and we refer to $T \cap W_{\text{SD}\alpha}$ 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 . That is, we will show that a vertex v is winning for sure-DirFWMP(ℓ, α)-positive-Reach(T) if and only if v is added to R at some point in the execution of the algorithm.

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 8) 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 (Lines 10 to 21)



■ **Figure 4** An MDP where we want to determine the winning region for $\text{sure-DirFWMP}(\ell = 3, \alpha = 0)\text{-positive-Reach}(T = \{v_4\})$. The winning region is $\{v_1, v_2, v_3, v_4\}$.

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 $\text{sure-DirFWMP}(\ell, \alpha)$. Formally, an edge e in $\mathcal{M}$ is *good* with respect to E_g if there exists a winning strategy σ_e for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E_g}(T)$ from edge e in $\mathcal{M}$. That is, all outcomes of σ_e with initial edge e satisfy $\text{DirFWMP}(\ell, \alpha)$, 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.

If `IsGoodEdge` returns false, then the edge is added to E_b (Line 8). 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 $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$ if s is not already in R (Line 6). This is because the winning strategy σ_e for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E_g}(T)$ from edge (s, t) immediately gives a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ 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 $E'_g \supsetneq E_g$, since a strategy σ_e that is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E_g}(T)$ from e is also winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E'_g}(T)$ from e , since satisfaction of $\text{Reach}_{\{e\} \cup E_g}(T)$ implies satisfaction of $\text{Reach}_{\{e\} \cup E'_g}(T)$. 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 $E'_g \supsetneq E_g$, as non-existence of a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E_g}(T)$ does not imply the non-existence of a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E'_g}(T)$. Thus, when an edge e is added to E_g , the edges present in E_b may or may not be bad with respect to $E_g \cup \{e\}$. Therefore, we reset E_b to the empty set each time an edge is added to E_g (Line 6), 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. The **while** loop ends 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 $\mathcal{M}$, eventually all such edges are added to either E_g or E_b , and thus, the **while** loop eventually terminates. 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. When the **while** loop terminates in Line 9, we return the set R of vertices.

► **Example 12.** Consider the MDP in Figure 4. We explain how Algorithm 3 works on this MDP. We consider the winning condition $\text{sure-DirFWMP}(\ell = 3, \alpha = 0)\text{-positive-Reach}(T = \{v_4\})$. Initially, we have $T = R = \{v_4\}$ and $E_g = E_b = \emptyset$. 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 $E_g = \emptyset$ 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 $\text{sure-DirFWMP}(\ell, \alpha)$. 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 $\text{sure-DirFWMP}(3, 0)\text{-positive-Reach}_{\{(v_1, v_2), (v_2, v_4)\}}(T)$. 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 $\text{sure-DirFWMP}(3, 0)$ 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 $\text{sure-DirFWMP}(3, 0)$ 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 $\emptyset$, 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 $\emptyset$ 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 $v_1 v_2 v_3 v_4$ 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 $\text{sure-DirFWMP}(3, 0)$. 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 that is returned by Algorithm 3 is $\{v_1, v_2, v_3, v_4\}$ which is exactly the set of vertices from which $\text{sure-DirFWMP}(\ell = 3, \alpha = 0)\text{-positive-Reach}(T = \{v_4\})$ can be satisfied. $\perp$

Description of IsGoodEdge procedure. Given an MDP $\mathcal{M}$, the procedure $\text{IsGoodEdge}(e = (s, t), E_g)$ determines if an edge e is good with respect to the set E_g of edges, that is, the procedure checks if the edge e is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E_g}(T)$ in $\mathcal{M}$. In other words, IsGoodEdge checks if, starting from e , it is possible with positive probability to reach T through only edges in $\{e\} \cup E_g$ and also simultaneously satisfy $\text{sure-DirFWMP}(\ell, \alpha)$. The procedure constructs an MDP $\mathcal{M}_e$ by augmenting some vertices and edges to $\mathcal{M}$, and returns true if and only if a designated vertex $\hat{s}$ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}_e$. This reduces the problem of satisfaction of $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$ to that of satisfaction of $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}_e$, which we know how to solve from [11, Algorithm 2]. The correctness of this reduction follows from Lemma 13. We describe the construction of $\mathcal{M}_e$ in two steps.

In the first step, we construct an intermediate MDP $\mathcal{M}'_e$. The set of vertices of the MDP $\mathcal{M}'_e$ is $V \cup \tilde{R} \cup \{\hat{s}\}$, where

- the set V is the set of all the vertices of $\mathcal{M}$;
- the set $\tilde{R}$ contains a copy $\tilde{v}$ of every vertex v in R
(we denote by $\tilde{T}$ the set $\{\tilde{v} \in \tilde{R} \mid v \in T\}$, that is, the set of all vertices in $\tilde{R}$ in $\mathcal{M}'_e$ that are copies of vertices in T , and we have that $\tilde{T} \subseteq \tilde{R}$);
- the set $\{\hat{s}\}$ contains a copy of the vertex s .

We note that the copy of a vertex that is controlled by $\mathcal{P}_\circ$ is also controlled by $\mathcal{P}_\circ$, and the copy of a probabilistic vertex is a probabilistic vertex.

The set of edges of the MDP $\mathcal{M}'_e$ is $E \cup \tilde{E}_g \cup \tilde{E}_\diamond \cup \{(\hat{s}, \tilde{t})\} \cup \hat{E}_\diamond$, where

- the set E is the set of all the edges of $\mathcal{M}$;

- the set $\tilde{E}_g$ contains a copy $(\tilde{u}, \tilde{v})$ of every edge (u, v) in E_g (recall that for every edge (u, v) belonging to E_g , we have that $u \in R \setminus T$ and $v \in R$, and thus, $(\tilde{u}, \tilde{v}) \in (\tilde{R} \setminus \tilde{T}) \times \tilde{R}$);
- the set $\tilde{E}_\diamond$ contains, for every probabilistic vertex u in $R \setminus T$, for every out-edge (u, v) of u that is not in E_g , a copy $(\tilde{u}, v)$ of (u, v) ;
- the set $\{(\hat{s}, \tilde{t})\}$ contains a copy of the edge $e = (s, t)$,
- if s (and hence also $\hat{s}$) is a vertex belonging to $\mathcal{P}_\circ$, then $\hat{E}_\diamond$ is empty, and if s is a probabilistic vertex, then $\hat{E}_\diamond$ contains a copy $(\hat{s}, v)$ of every out-edge (s, v) of s such that $v \neq t$.

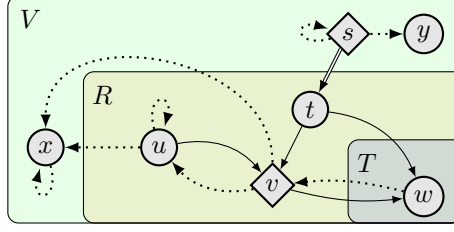
When we make a copy of an edge, we preserve its payoff and probability. Briefly, the set E contains edges that start and end in V . The set $\tilde{E}_g \cup \tilde{E}_\diamond$ contains edges starting in $\tilde{R} \setminus \tilde{T}$. The set $\{(\hat{s}, \tilde{t})\} \cup \hat{E}_\diamond$ contains edges starting at $\hat{s}$. The set $\tilde{E}_\diamond \cup \hat{E}_\diamond$ contains edges starting in probabilistic vertices in $(\tilde{R} \setminus \tilde{T}) \cup \{\hat{s}\}$ and ending in V . We note that if s belongs to R , then in addition to s and $\hat{s}$, the MDP $\mathcal{M}'_e$ also contains a copy $\tilde{s}$ of s .

In the second step, we obtain $\mathcal{M}_e$ from $\mathcal{M}'_e$ by identifying, for all $v \in T$ in $\mathcal{M}'_e$, the vertex $\tilde{v}$ with v . That is, we merge $\tilde{v}$ and v into one vertex, and the set of in-edges (resp., out-edges) of v in $\mathcal{M}_e$ is the union of the in-edges (resp., out-edges) of v and in-edges (resp., out-edges) of $\tilde{v}$ in $\mathcal{M}'_e$. We have that $\tilde{T}$ and T become identical in $\mathcal{M}_e$, and belong to both V and $\tilde{R}$ in $\mathcal{M}_e$. In other words, vertices in $\tilde{T}$ in $\mathcal{M}'_e$ are merged into vertices in T , and thus, each vertex in $\mathcal{M}_e$ belongs to exactly one of the three sets: V , $\tilde{R} \setminus T$, or $\{\hat{s}\}$. We show in Figure 5 an MDP $\mathcal{M}$, in Figure 6 the intermediate MDP $\mathcal{M}'_e$, and in Figure 7 the MDP $\mathcal{M}_e$. We describe in further detail the out-edges of vertices in $\mathcal{M}_e$.

- If s is a vertex belonging to $\mathcal{P}_\circ$, then $\hat{s}$ has exactly one out-edge $(\hat{s}, \tilde{t})$, and $\tilde{t}$ belongs to $\tilde{R}$. This simulates forcing $\mathcal{P}_\circ$ to choose the edge (s, t) from s in $\mathcal{M}$.
- If s is a probabilistic vertex, then in addition to $(\hat{s}, \tilde{t})$, the vertex $\hat{s}$ also has the set $\hat{E}_\diamond$ of out-edges, that is, an out-edge $(\hat{s}, v)$ for every out-edge (s, v) of s such that $v \neq t$. Thus, with positive probability, the token moves along the edge $(\hat{s}, \tilde{t})$. If the token does not choose the edge $(\hat{s}, \tilde{t})$, then the behaviour of the token from $\hat{s}$ is the same as the behaviour of the token from s if the edge (s, t) is not chosen.
- For each vertex $\tilde{u}$ in $\tilde{R} \setminus T$ belonging to $\mathcal{P}_\circ$, the vertex only has out-edges from $\tilde{E}_g$, that is, the set of out-edges of $\tilde{u}$ consists of only copies $(\tilde{u}, \tilde{v})$ of edges (u, v) that belong to E_g . This simulates forcing $\mathcal{P}_\circ$ to choose an out-edge from E_g when the token reaches vertex u in $\mathcal{M}$.
- For each probabilistic vertex $\tilde{u}$ in $\tilde{R} \setminus T$, in addition to the copy $(\tilde{u}, \tilde{v})$ of every out-edge (u, v) of u belonging to E_g , the vertex $\tilde{u}$ also has the set $\tilde{E}_\diamond$ of out-edges, that is an out-edge $(\tilde{u}, v)$ for every out-edge of (u, v) of u that does not belong to E_g . Since $u \in R \setminus T$ and every vertex in $R \setminus T$ has at least one out-edge in E_g , it follows that $\tilde{u}$ has at least one out-edge in $\tilde{E}_g$. Thus, with positive probability, the token chooses an edge in $\tilde{E}_g$ from $\tilde{u}$. If the token does not choose an edge in $\tilde{E}_g$ from $\tilde{u}$, then the behaviour of the token from $\tilde{u}$ is the same as the behaviour of the token from u if an edge in E_g is not chosen from u .
- For all vertices v in V in $\mathcal{M}_e$, the set of out-edges of v is the same as the set of out-edges of v in $\mathcal{M}$.

We are interested in plays in $\mathcal{M}_e$ that start from $\hat{s}$. The following observations give more insight on plays in $\mathcal{M}_e$.

1. The vertex $\hat{s}$ in $\mathcal{M}_e$ has no in-edges. Thus, starting from $\hat{s}$, the token moves to $(\tilde{R} \setminus T) \cup V$ in the next step and never returns to $\hat{s}$. All edges that start from a vertex in V end in



■ **Figure 5** An MDP $\mathcal{M}$ with $T = \{w\}$, $R = \{t, u, v, w\}$, and $E_g = \{(u, v), (t, v), (t, w), (v, w)\}$. We want to determine if the edge (s, t) is good with respect to E_g . Edges in E_g are solid and edges not in E_g are dotted.

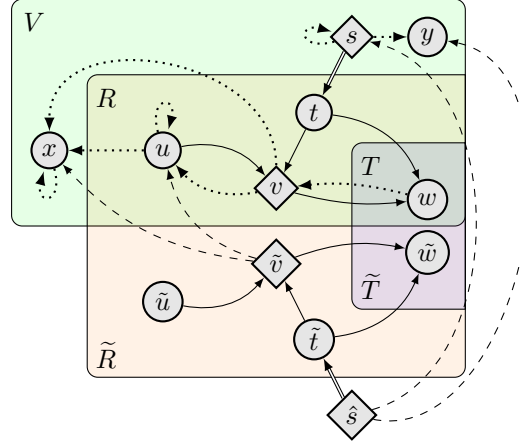
- a vertex in V . Therefore, if the token reaches V , then the token remains in V forever. Thus, every play in $\mathcal{M}_e$ starting from $\hat{s}$ is of the form $\{\hat{s}\} (\tilde{R} \setminus T)^\omega$ or $\{\hat{s}\} (\tilde{R} \setminus T)^* V^\omega$.
2. We can project vertices in $\mathcal{M}_e$ to vertices in $\mathcal{M}$ by ignoring the hats and the tildes on the vertex names. Formally, we define a projection $\text{Proj}: V \cup \tilde{R} \cup \{\hat{s}\} \rightarrow V$ as $\text{Proj}(v) = v$ for all $v \in V$, $\text{Proj}(\tilde{v}) = v$ for all $v \in \tilde{R}$, and $\text{Proj}(\hat{s}) = s$. Projection of vertices in $\mathcal{M}_e$ naturally extends to projection of prefixes $\text{Proj}: \text{Pref}_{\mathcal{M}_e} \rightarrow \text{Pref}_{\mathcal{M}}$ and projection of plays $\text{Proj}: \text{Plays}_{\mathcal{M}_e} \rightarrow \text{Plays}_{\mathcal{M}}$ defined as $\text{Proj}(v_0 v_1 v_2 \dots) = \text{Proj}(v_0) \text{Proj}(v_1) \text{Proj}(v_2) \dots$. It can be seen for all $\pi_e \in \text{Plays}_{\mathcal{M}_e}$ that the projection $\text{Proj}(\pi_e)$ is a valid play in $\mathcal{M}$, that is, if v_{i+1} is an out-neighbour of v_i in $\mathcal{M}_e$, then $\text{Proj}(v_{i+1})$ is an out-neighbour of $\text{Proj}(v_i)$ in $\mathcal{M}$, and further, the sequence of edge payoffs in π_e is the same as the sequence of edge payoffs in $\text{Proj}(\pi_e)$.
 3. For every strategy σ_e in $\mathcal{M}_e$, we can construct a strategy in σ in $\mathcal{M}$ (using additional memory) such that the behaviour of the two strategies is the same, that is, the same edge payoffs are seen with the same probabilities in outcomes of σ_e in $\mathcal{M}_e$ and σ in $\mathcal{M}$. Formally, we have that for all prefixes ρ in $\mathcal{M}_e$ ending in a vertex belonging to $\mathcal{P}_\circ$, we have $\text{Proj}(\sigma_e(\rho)) = \sigma(\text{Proj}(\rho))$. Given a prefix ρ in $\mathcal{M}_e$, the strategy σ mimics on the projected prefix $\text{Proj}(\rho)$ the projection of the behaviour of σ_e on ρ . Since σ_e may have different behaviour depending on whether the token is on $\tilde{u}$ or u in $\mathcal{M}_e$ for a vertex $u \in R \setminus T$, or different behaviour for the token at s , $\tilde{s}$, and $\hat{s}$ for the vertex s , the strategy σ requires at most three times as much memory as σ_e to distinguish among these cases.

Proof of correctness of Algorithm 3. We prove the correctness of Algorithm 3 in Theorem 17. Before that, we prove the correctness of `IsGoodEdge` in Lemma 13, and prove some intermediate results in Propositions 14 and 15 and Lemma 16.

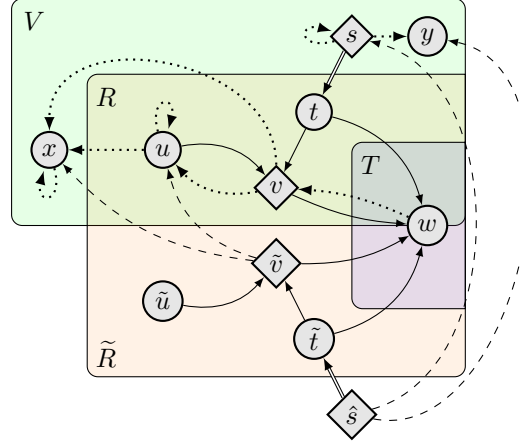
► **Lemma 13.** *Given E_g and an edge $e = (s, t)$ with $s \in V \setminus T$ and $t \in R$, the procedure `IsGoodEdge`(e, E_g) correctly computes whether e is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\} \cup E_g}(T)$ in $\mathcal{M}$.*

Proof. Recall that E_g is empty at the start of the algorithm and that edges are added to E_g over the course of the algorithm. We prove the statement of the lemma by induction on the size of E_g .

Base case: For the base case, we suppose that $E_g = \emptyset$. The procedure `IsGoodEdge` is called on an edge $e = (s, t)$ with $s \in V \setminus T$ and $t \in R$. Since E_g is empty, we have that $R = T$, and thus $t \in T$. We show that `IsGoodEdge`($e, \emptyset$) returns true if and only if the edge (s, t) is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\}}(T)$ in $\mathcal{M}$.



■ **Figure 6** Construction of $\mathcal{M}'_e$. The vertices and edges present in $\mathcal{M}$ remain unchanged. A copy $\{\tilde{t}, \tilde{u}, \tilde{v}, \tilde{w}\}$ of every vertex in R is added. A copy $\hat{s}$ of s is also added. Edges in $\tilde{E}_\diamond \cup \hat{E}_\diamond$ are dashed.



■ **Figure 7** Obtaining $\mathcal{M}_e$ from $\mathcal{M}'_e$ by merging vertex $\tilde{w}$ in $\tilde{T}$ with w in T .

($\Rightarrow$) Suppose $\text{IsGoodEdge}(e, \emptyset)$ returns true, that is, the vertex $\hat{s}$ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}_e$. If s is a vertex belonging to $\mathcal{P}_\circ$, then $\hat{s}$ has only one out-edge $(\hat{s}, \tilde{t})$ where $\tilde{t}$ is the same as t since $t \in T$. Since $\hat{s}$ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$, it follows that there exists a strategy σ_e of $\mathcal{P}_\circ$ that moves the token along the edge $(\hat{s}, \tilde{t})$ and then continues the play from t in a way that satisfies $\text{sure-DirFWMP}(\ell, \alpha)$. On the other hand, if s is a probabilistic vertex, then $\hat{s}$ has an out-edge $(\hat{s}, t)$ and may have other out-edges. Since $\hat{s}$ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$, we have that no matter what edge is chosen from $\hat{s}$, player $\mathcal{P}_\circ$ has a strategy that continues the play such that it satisfies $\text{sure-DirFWMP}(\ell, \alpha)$. In particular, there exists a strategy σ_e of $\mathcal{P}_\circ$ that continues the play from $(\hat{s}, t)$ such that it satisfies $\text{sure-DirFWMP}(\ell, \alpha)$. In both cases, the strategy σ that mimics the behaviour of σ_e (that is $\sigma(s \cdot t \cdot \rho) = \sigma_e(\hat{s} \cdot t \cdot \rho)$ for all prefixes ρ) is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ from e in $\mathcal{M}$. Since the token reaches T immediately after the initial edge e , we have that σ is also winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\}}(T)$ from e in $\mathcal{M}$.

($\Leftarrow$) Conversely, suppose that the edge (s, t) is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\}}(T)$ in $\mathcal{M}$. Then, there exists a strategy σ such that every outcome of σ in $\mathcal{M}$

starting from e satisfies $\text{DirFWMP}(\ell, \alpha)$. We use σ to construct a strategy σ_e that is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ from $\hat{s}$ in $\mathcal{M}_e$. If s belongs to $\mathcal{P}_\circ$, then so does $\hat{s}$, and let σ_e be defined as $\sigma_e(\hat{s}) = t$, and $\sigma_e(\hat{s} \cdot t \cdot \rho) = \sigma(s \cdot t \cdot \rho)$ for all prefixes ρ . On the other hand, if s is a probabilistic vertex, in which case so is $\hat{s}$, then let σ_e be defined as $\sigma_e(\hat{s} \cdot \rho) = \sigma(s \cdot \rho)$ for all ρ . In both cases, an outcome of σ visits the same edge payoffs with the same probability as an outcome of σ_e . Since σ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ from s , we also have that σ_e is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ from $\hat{s}$.

Inductive step: For the inductive step, suppose $|E_g| = k > 0$, and we label the edges of E_g as $e_1, e_2, \dots, e_k$ in the order they were added to E_g . We want to verify if an edge $e = (s, t)$ for $s \in V \setminus T$ and $t \in R$ is good with respect to E_g .

($\Rightarrow$) Suppose the procedure $\text{IsGoodEdge}(e, E_g)$ returns true, that is, the vertex $\hat{s}$ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}_e$, and thus, there exists a strategy σ_e of $\mathcal{P}_\circ$ that is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}_e$ from $\hat{s}$. We construct a strategy from the edge e that is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\} \cup E_g}(T)$ in $\mathcal{M}$.

Suppose $\mathcal{P}_\circ$ follows the strategy σ_e from $\hat{s}$ until the first time in the play where all α -windows are closed. For all outcomes of σ_e , this happens in at most ℓ steps from the start of the play since σ_e is winning for $\text{sure-DirFWMP}(\ell, \alpha)$. Moreover, since every vertex in $\tilde{R} \setminus T$ belonging to $\mathcal{P}_\circ$ contains has all out-edges in $\tilde{E}_g$ and every probabilistic vertex in $\tilde{R} \setminus T$ has at least one out-edge in $\tilde{E}_g$, starting from $\hat{s}$ we have that at least one of the following hold: (i) with positive probability the token reaches T within at most ℓ steps while taking only edges in $\{(\hat{s}, \tilde{t})\} \cup \tilde{E}_g$, or (ii) the token remains in $\tilde{R} \setminus T$ until all α -windows are closed while taking only edges in $\{(\hat{s}, \tilde{t})\} \cup \tilde{E}_g$.

Suppose the token reaches T while visiting only edges in $\{(\hat{s}, \tilde{t})\} \cup \tilde{E}_g$, that is, we have that $\text{Reach}_{\{(\hat{s}, \tilde{t})\} \cup \tilde{E}_g}(T)$ is satisfied and the token has entered V . From Observation 1, the token now behaves the same as it would in $\mathcal{M}$. If $\mathcal{P}_\circ$ continues to follow the strategy σ_e for the rest of the play, then since σ_e is winning for $\text{sure-DirFWMP}(\ell, \alpha)$, we have that all α -windows are closed in at most ℓ steps. Thus, in this case, $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{(\hat{s}, \tilde{t})\} \cup \tilde{E}_g}(T)$ is satisfied from $\hat{s}$ in $\mathcal{M}_e$. If $\mathcal{P}_\circ$ follows the projection σ (as described in Observation 3) in $\mathcal{M}$ of the strategy σ_e , then we see that σ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\} \cup E_g}(T)$ in $\mathcal{M}$ from (s, t) .

Otherwise, suppose that the token remains in $\tilde{R} \setminus T$ until all α -windows are closed for the first time while choosing only edges in $\{(\hat{s}, \tilde{t})\} \cup \tilde{E}_g$. Suppose the token is on a vertex $\tilde{s}_1 \in \tilde{R} \setminus T$ for some $s_1 \in R \setminus T$ when all α -windows are closed for the first time. Since s_1 belongs to $R \setminus T$, we have that s_1 has at least one out-edge e_i in E_g for $1 \leq i \leq k$, and let this edge be $e_i = (s_1, t_1)$ for some $t_1 \in \tilde{R} \setminus T$. The edge e_i belongs to E_g because the procedure $\text{IsGoodEdge}(e_i, E'_g)$ returned true for $E'_g = \{e_1, e_2, \dots, e_{i-1}\}$. By the induction hypothesis, we have that the edge (s_1, t_1) is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{E'_g}(T)$ in $\mathcal{M}$. Using Observation 3, we have that starting from (s, t) , player $\mathcal{P}_\circ$ has a strategy that reaches s_1 with positive probability while only visiting edges in $\{e\} \cup E_g$ while also surely closing all α -windows in at most ℓ steps. Then, from the induction hypothesis, from the edge (s_1, t_1) , player $\mathcal{P}_\circ$ has a strategy to satisfy $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{(s_1, t_1)\} \cup E'_g}(T)$ in $\mathcal{M}$. Thus, the edge e is winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\} \cup E_g}(T)$ in $\mathcal{M}$.

($\Leftarrow$) Suppose $\text{IsGoodEdge}(e, E_g)$ returns false, that is, the vertex $\hat{s}$ is not winning for $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}_e$. Since the $\tilde{R} \cup \{\hat{s}\}$ part of $\mathcal{M}_e$ simulates forcing $\mathcal{P}_\circ$ to only choose edges in $\{e\} \cup E_g$, we get that starting from s in $\mathcal{M}$, if $\mathcal{P}_\circ$ only chooses edges in $\{e\} \cup E_g$ from vertices belonging to $\mathcal{P}_\circ$, then $\mathcal{P}_\circ$ cannot satisfy $\text{sure-DirFWMP}(\ell, \alpha)$. Hence, starting from s in $\mathcal{M}$, it is not possible for $\mathcal{P}_\circ$ to satisfy $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\} \cup E_g}(T)$. Thus, the edge (s, t) is not winning for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{\{e\} \cup E_g}(T)$ in $\mathcal{M}$. $\blacktriangleleft$

► **Proposition 14.** *At the end of Algorithm 3, if v is a vertex with an out-edge (v, u) that is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$, then v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$.*

Proof. Since v has an out-edge (v, u) that is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$, we have that after taking the edge (v, u) , there exists a strategy σ continuing the play from u that is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$. There are two cases depending on whether the vertex v belongs to $V_\circ$ or $V_\diamond$.

- If $v \in V_\circ$, then $\mathcal{P}_\circ$ can move the token to u , and then follow the strategy σ from u to satisfy $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$.
- If $v \in V_\diamond$, then the token moves from v to u with positive probability, from which $\mathcal{P}_\circ$ can follow the strategy σ from u to satisfy $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$. If token moves from v to a vertex other than u , then since v is winning for $\text{sure-DirFWMP}(\ell, \alpha)$, we have that $\text{sure-DirFWMP}(\ell, \alpha)$ is still satisfied.

In both cases, we have that the vertex v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$. ◀

► **Proposition 15.** *At the end of Algorithm 3, for all edges (u, v) such that $u \notin R$ and $v \in R$, the edge (u, v) is not winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$.*

Proof. Suppose towards a contradiction that there exists an edge (u, v) such that $u \notin R$ and $v \in R$ and (u, v) is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$. Then, from Line 6 of the algorithm, we have that u must be included in R , which is a contradiction. ◀

► **Lemma 16.** *At the end of Algorithm 3, for all vertices v in MDP $\mathcal{M}$, the vertex v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$ if and only if v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$.*

Proof. Since satisfaction of $\text{Reach}_{E_g}(T)$ implies satisfaction of $\text{Reach}(T)$, the forward direction is immediate. We prove the converse, that is, if v is not winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$, then v is not winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$.

Suppose towards a contradiction that starting from v , there exists a strategy σ that is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$. Consider an outcome π of this strategy that reaches T and also satisfies $\text{DirFWMP}(\ell, \alpha)$. Since v is not winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$, we have that there exists at least one edge in this outcome before reaching T that does not belong to E_g . Consider the last edge in this outcome before reaching T that does not belong to E_g , say (x, y) . Every edge in the outcome after (x, y) and before the token reaches T belongs to E_g . From the proof of Lemma 13, we have that since (x, y) does not belong to E_g , there does not exist a strategy that is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{(x, y)\} \cup E_g}(T)$ from x . This is, however, a contradiction, since the path from x to T in π that does not visit any edge not in $\{(x, y)\} \cup E_g$ occurs with positive probability. ◀

► **Theorem 17.** *Algorithm 3 computes the winning region for $\mathcal{P}_\circ$ for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$.*

Proof. Recall that if a vertex v does not belong to $W_{\text{SD}\alpha}$, then v is not winning for $\text{sure-DirFWMP}(\ell, \alpha)$ in $\mathcal{M}$, and therefore also not winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$. The algorithm correctly removes the vertices not in $W_{\text{SD}\alpha}$ in Line 2. In the rest of the algorithm and the rest of this proof, we work with the subMDP $\mathcal{M}[W_{\text{SD}\alpha}]$ and refer to this subMDP as $\mathcal{M}$, so we have that all vertices in $\mathcal{M}$ are winning for $\text{sure-DirFWMP}(\ell, \alpha)$.

Recall that the algorithm returns the set $R \subseteq V$. We show for all v in $\mathcal{M}$ that v belongs to the set R if and only if v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$. We consider two cases separately. First, we show this result holds for all $v \in T$, and then, for all $v \in V \setminus T$.

Let $v \in T$. We have that every vertex in $\mathcal{M}$ is winning for $\text{sure-DirFWMP}(\ell, \alpha)$. Moreover, every play π starting from v satisfies $\text{Reach}(T)$ since v is in T , and thus, a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)$ from v is also a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$. Thus, v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$. Since $T \subseteq R$, we also have that v belongs to R . We have shown that for each vertex v in T , the vertex v is in R if and only if v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$.

Now we see the interesting case, that is when $v \in V \setminus T$.

($\Rightarrow$) Suppose v belongs to R . Since v also belongs to $V \setminus T$, we have that v belongs to $R \setminus T$. The vertex v belongs to $R \setminus T$ because it is added to the set R in Line 6 when an out-edge e of v is determined to be good by **IsGoodEdge** with respect to the set E_g at that point. From Lemma 13, we have that this edge e is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E_g}(T)$ in $\mathcal{M}$. From Proposition 14, it follows that v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{E_g}(T)$ in $\mathcal{M}$ for the set E_g at the end of Algorithm 3. By Lemma 16, it implies that v is winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$.

($\Leftarrow$) Suppose v does not belong to R . Since $T \subseteq R$ and $v \notin R$, in order to satisfy $\text{positive-Reach}(T)$ from v , the token needs to reach R with positive probability. From Proposition 15, we have that starting from v , if the token enters R , it enters R through an edge in E_b . However, from Lemma 13, it follows that once an edge e in E_b is taken, then $\mathcal{P}_\circ$ does not have a strategy to satisfy $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}_{\{e\} \cup E_g}(T)$, and from Lemma 16, player $\mathcal{P}_\circ$ does not have a strategy to satisfy $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ either. Thus, we have that v is not winning for $\text{sure-DirFWMP}(\ell, \alpha)\text{-positive-Reach}(T)$ in $\mathcal{M}$. $\blacktriangleleft$

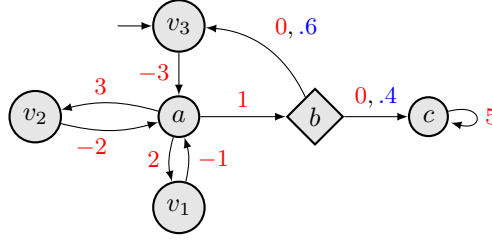
Memory requirement and complexity for sure-almost-sure FWMP(ℓ)

We conclude this section by discussing bounds on the memory requirement and the complexity for the sure-almost-sure satisfaction of $\text{FWMP}(\ell)$ objectives.

Memory requirement: lower bound. It is known that memory of size ℓ suffices for satisfying $\text{sure-FWMP}(\ell, \alpha)$ [18, Theorem 4.4] and for $\text{almost-sure-FWMP}(\ell, \alpha)$ [18, Theorem 6.5]. In Example 18, we show that $\Omega(\max\{|V|, \ell\})$ memory may be necessary for the sure-almost-sure satisfaction of the $\text{FWMP}(\ell)$ objective.

► **Example 18.** Consider the MDP in Figure 8. Note that the only vertex belonging to $\mathcal{P}_\circ$ with more than one out-edge is a . Thus, a strategy of $\mathcal{P}_\circ$ 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 $\text{sure-FWMP}(2, 0)$ and the memoryless strategy that always chooses vertex b from a satisfies $\text{almost-sure-FWMP}(2, 5)$. However, neither of these strategies are winning for $\text{sure-FWMP}(2, 0)\text{-almost-sure-FWMP}(2, 5)$.

In order to satisfy $\text{sure-FWMP}(2, 0)$, player $\mathcal{P}_\circ$ must eventually close all 0-windows in at most 2 steps, and in order to satisfy $\text{almost-sure-FWMP}(2, 5)$ 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, then token must take the path $v_3 \cdot a \cdot v_2 \cdot a \cdot v_1 \cdot a \cdot b$. 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



■ **Figure 8** 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 .

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 . When the token moves from v_1 to a , then the token can move from a to b while also ensuring that the 0-window starting at v_1 closes in 2 steps. Thus, in order to reach b from v_3 while closing all 0-windows in at most 2 steps, the token needs to visit both v_2 and v_1 and hence this requires a strategy of memory size 3.

This example can be generalized to an MDP $\mathcal{M}_n$ with $n + 3$ vertices $\{v_n, v_{n-1}, \dots, v_1, a, b, c\}$, 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_3) with payoff 0, an edge (b, c) with payoff 0, and a self-loop on c with payoff 5. It can be seen that memory of size $n - 1$ 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 ℓ memory, we have that $\mathcal{P}_\circ$ needs at least $\Omega(\max\{|V|, \ell\})$ memory to satisfy sure-FWMP(2, 0)-almost-sure-FWMP(2, 5). $\lrcorner$

Memory requirement: upper bound. We give an upper bound for the memory requirement for winning strategies for sure-almost-sure satisfaction of FWMP(ℓ) by explicitly constructing a winning strategy. We first show a winning strategy σ_{sdpr} for sure-DirFWMP(ℓ, α)-positive-Reach(T). Then, we use this strategy to construct a winning strategy σ_{sdab} for sure-DirFWMP(ℓ, α)-almost-sure-Büchi(T), which is used in turn to describe a winning strategy σ_{sas} for sure-almost-sure satisfaction of FWMP(ℓ).

Winning strategy for sure-DirFWMP(ℓ, α)-positive-Reach(T). Recall that we have a set E_g of edges and a winning region R for sure-DirFWMP(ℓ, α)-positive-Reach(T) at the end of the execution of Algorithm 3. Let $u_1, u_2, \dots, u_n$ denote the set of vertices in $R \setminus T$ at the end of the algorithm in the order in which they were added to $R \setminus T$ where $|R \setminus T| = n$. For $1 \leq i \leq n$, let e_i denote the first out-edge of u_i added to E_g during the algorithm, and let E_g^i denote the set of all edges present in E_g immediately after the edge e_i is added to E_g .

For an arbitrary vertex v belonging to R , we describe a winning strategy σ_{sdpr} from v . If v belongs to T , then Reach(T) is already satisfied at the start of the play, and it suffices for σ_{sdpr} to mimic a winning strategy for sure-DirFWMP(ℓ, α) from v throughout the rest of the play. Otherwise, if $v \notin T$, then $v \in R \setminus T$, and v is u_i for some $1 \leq i \leq n$. The construction of $\mathcal{M}_{e_i}$ ensures that there exists a winning strategy σ_{e_i} for sure-DirFWMP(ℓ, α) from $\hat{u}_i$ in $\mathcal{M}_{e_i}$, and from Lemma 13 it follows that projecting this strategy to $\mathcal{M}$ gives a strategy σ_i that is winning for sure-DirFWMP(ℓ, α)-positive-Reach $_{E_g^i}(T)$ from u_i . The strategy σ_{sdpr} mimics σ_i from u_i , as long as only edges in E_g^i are chosen, until either all α -windows are closed for the first time (which happens in at most ℓ steps) or the token reaches T , whichever occurs first. If the token ever takes an edge not in E_g^i , then σ_{sdpr} continues to follow a winning strategy

for $\text{sure-DirFWMP}(\ell, \alpha)$ for the rest of the play, ensuring that all α -windows close in at most ℓ steps. Otherwise, with probability at least $(\mathbb{P}_{\min})^\ell$, the token only chooses edges in E_g^i until all α -windows are closed for the first time or the token reaches T .

If the token reaches T , then σ_{sdpr} continues to follow a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)$ for the rest of the play. Otherwise, the token reaches a vertex u_j with all α -windows closed for some $1 \leq j < i$, and σ_{sdpr} switches to mimicking a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}_{E_g^j}(T)$ from u_j . Since n is finite and $E_g^j \subsetneq E_g^i$ for $1 \leq j < i \leq n$, we can repeat the same procedure from u_j to get that, with positive probability, the token reaches T eventually (in fact, in at most $n \cdot \ell$ steps).

Each strategy σ_{e_i} is a winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)$ and requires at most ℓ memory [18, Lemma 4.1]. The strategy σ_i obtained after projecting σ_{e_i} to $\mathcal{M}$ requires at most three times the memory, that is, at most $3 \cdot \ell$. For each $1 \leq i \leq n$, there is a potentially different strategy σ_i that σ_{sdpr} mimics, and thus, the memory requirement σ_{sdpr} is at most $3 \cdot n \cdot \ell$, which is at most $3 \cdot |V| \cdot \ell$.

Winning strategy for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(T). For an arbitrary vertex v belonging to the winning region for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(T), we define a winning strategy σ_{sdab} . The strategy σ_{sdab} follows the winning strategy σ_{sdpr} for $\text{sure-DirFWMP}(\ell, \alpha)$ -positive- $\text{Reach}(T)$ as described above as long as there are no deviations from E_g and no vertex in T is reached. If there is a deviation or a vertex in T is reached, then the strategy σ_{sdab} continues to follow the strategy σ_{sdpr} for at most $\ell - 1$ more steps until all open α -windows are closed, following which the strategy σ_{sdab} resets, that is, it forgets the history and starts following σ_{sdpr} afresh. The memory requirement of σ_{sdab} is no greater than that of σ_{sdpr} , that is at most $3 \cdot |V| \cdot \ell$.

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

- If the token is in A^i for some $1 \leq i \leq k$, then σ_{sas} follows a sure-attractor strategy to reach $W_{\text{sdab}}^i \cup W^{i-1}$.
- If the token is in W_{sdab}^i for some $1 \leq i \leq k$, then σ_{sas} follows the winning strategy σ_{sdab} for $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(P^i). This ensures that $\text{DirFWMP}(\ell, \alpha)$ 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 $\text{sure-FWMP}(\ell, \beta)$.

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

► **Theorem 19.** *The sure-almost-sure problem for the $\text{FWMP}(\ell)$ objective is in PTIME when ℓ is given in unary. The memory required is at least $\Omega(\max\{|V|, \ell\})$ and at most $\mathcal{O}(|V| \cdot \ell)$.*

Proof. We have shown the bounds on the memory requirement above. We show that Algorithm 1 has a polynomial running time. First, we note that **SureDirFWMP** can be computed in time $\mathcal{O}(|V| \cdot |E| \cdot \ell \cdot \log(w_{\max}))$ [11, Lemma 5] and **SureFWMP** can be computed in time $\mathcal{O}(|V|^2 \cdot |E| \cdot \ell \cdot \log(w_{\max}))$ [11, Lemma 6], which are polynomial in the size of the input when ℓ is given in unary. We denote the running time of **SureDirFWMP** by $\mathbb{C}$.

The **repeat** loop from Line 4 to Line 9 of Algorithm 1 is dominated by the call to Algorithm 2, which in turn is a recursive algorithm that 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 this loop, we make a call to **IsGoodEdge** which involves calling **SureDirFWMP** on

$\mathcal{M}_e$. Thus, the running time of Algorithm 3 is $\mathcal{O}(|E|^2 \cdot \mathbb{C})$. Algorithm 2 has a recursion depth of at most $|V|$ since it is called on a smaller $\mathcal{M}$ each time, and hence its time complexity is $\mathcal{O}(|V| \cdot |E|^2 \cdot \mathbb{C})$. Finally, the **repeat** loop of Algorithm 1 runs for at most $|V|$ iterations, thus leading to a total time complexity of $\mathcal{O}(|V|^2 \cdot |E|^2 \cdot \mathbb{C})$, which is $\mathcal{O}(|V|^3 \cdot |E|^3 \cdot \ell \cdot \log(w_{\max}))$. This is polynomial in the size of the input when ℓ is given in unary. $\blacktriangleleft$

4.2 Sure-almost-sure satisfaction for BWMP

In this section, we present an algorithm to solve the sure-almost-sure satisfaction problem for BWMP, the bounded window mean-payoff objective. Recall that a play π satisfies $\text{BWMP}(\alpha)$ if there exists a window length $\ell \geq 1$ such that π satisfies $\text{FWMP}(\ell, \alpha)$.

The algorithm is similar to Algorithm 1. To solve the sure-almost-sure satisfaction problem for BWMP, we consider Algorithm 1 for the $\text{FWMP}(\ell)$ objective and replace every occurrence of $\text{FWMP}(\ell)$ with BWMP . Thus, the algorithm for sure-almost-sure BWMP has the same structure as that of Algorithm 1, i.e. it has subalgorithms for $\text{sure-DirBWMP}(\alpha)$ -almost-sure-Büchi(T) and $\text{sure-DirBWMP}(\alpha)$ -positive-Reach(T) (analogous to $\text{sure-DirFWMP}(\ell, \alpha)$ -almost-sure-Büchi(T) and $\text{sure-DirFWMP}(\ell, \alpha)$ -positive-Reach(T) respectively, obtained again by replacing all occurrences of $\text{FWMP}(\ell)$ with BWMP). We state the analogous algorithms for sure-almost-sure BWMP, $\text{sure-DirBWMP}(\alpha)$ -almost-sure-Büchi(T), and $\text{sure-DirBWMP}(\alpha)$ -positive-Reach(T) (Algorithm 7, Algorithm 8, and Algorithm 9, respectively) in the appendix for completeness. The proof of correctness for these algorithms also follows directly from the proof of correctness of the algorithms for the $\text{FWMP}(\ell)$ objectives (Theorem 9, Theorem 11, Theorem 17) by replacing $\text{FWMP}(\ell)$ with BWMP . We state in Lemma 20 and Lemma 21 results that relate the winning region for $\text{sure-DirBWMP}(\alpha)$ with the winning region for $\text{sure-DirFWMP}(\ell', \alpha)$ where $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$.

► **Lemma 20.** *All vertices in $\mathcal{M}$ are winning for $\text{sure-DirBWMP}(0)$ if and only if all vertices in $\mathcal{M}$ are winning for $\text{sure-DirFWMP}(\ell', 0)$, where $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$.*

Proof. We list some observations that are useful in the proof.

1. If a play π satisfies $\text{DirBWMP}(0)$, then π also satisfies the total-payoff objective $\text{TP}(0)$ in $\mathcal{M}$. This is because it happens infinitely often in π that all 0-windows are closed, and each time all 0-windows are closed, it is the case that the total payoff is non-negative. Thus, if a vertex v is winning for $\text{sure-DirBWMP}(0)$ in $\mathcal{M}$, then v is also winning for $\text{sure-TP}(0)$ in $\mathcal{M}$.
2. We recall [11, Lemma 10] which states that if a vertex v is winning for $\text{sure-TP}(0)$ in $\mathcal{M}$ then v is winning for $\text{sure-GW}(\ell', 0)$ in $\mathcal{M}$ for $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$, that is, $\mathcal{P}_{\bigcirc}$ can ensure for all outcomes that the 0-window starting at the v closes in at most ℓ' steps.
3. We recall from [11, Lemma 5] that if all vertices in $\mathcal{M}$ are winning for $\text{sure-GW}(\ell', 0)$, then all vertices in $\mathcal{M}$ are winning for $\text{sure-DirFWMP}(\ell', 0)$ since each time a 0-window closes, satisfaction of $\text{sure-GW}(\ell', 0)$ ensures that the next 0-window gets closed in at most ℓ' steps.
4. If a vertex v is winning for $\text{sure-DirFWMP}(\ell', 0)$ in $\mathcal{M}$, then v is also winning for $\text{sure-DirBWMP}(0)$ in $\mathcal{M}$, since ℓ' is a witness window length for the satisfaction of $\text{DirBWMP}(0)$.

Now, suppose all vertices in $\mathcal{M}$ are winning for $\text{sure-DirBWMP}(0)$. Then, we have that all vertices in $\mathcal{M}$ are winning for $\text{sure-TP}(0)$ (Observation 1), which implies that all vertices in $\mathcal{M}$ are winning for $\text{sure-GW}(\ell', 0)$ (Observation 2), which implies that all vertices in $\mathcal{M}$ are winning for $\text{sure-DirFWMP}(\ell', 0)$ (Observation 3), which in turn implies that all vertices are winning for $\text{sure-DirBWMP}(0)$ (Observation 4). Thus, all four of the preceding statements are

equivalent, and in particular, we have that all vertices in $\mathcal{M}$ are winning for $\text{sure-DirBWMP}(0)$ if and only if all vertices in $\mathcal{M}$ are winning for $\text{sure-DirFWMP}(\ell', 0)$. $\blacktriangleleft$

Lemma 21 lets us use $\text{DirFWMP}(\ell', \alpha)$ as a drop-in replacement for $\text{DirBWMP}(\alpha)$ whenever necessary.

► **Lemma 21.** *For every vertex v in an MDP $\mathcal{M}$, the vertex v is winning for $\text{sure-DirBWMP}(\alpha)$ if and only if v is winning for $\text{sure-DirFWMP}(\ell', \alpha)$, where $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$.*

Proof. Let W_B denote the winning region for $\text{sure-DirBWMP}(\alpha)$ in $\mathcal{M}$, and let v be a vertex in W_B . Since $\text{DirBWMP}(\alpha)$ is an objective that is closed under suffixes, from Proposition 1 we have that W_B induces a subMDP $\mathcal{M}[W_B]$ of $\mathcal{M}$.

Every vertex in $\mathcal{M}[W_B]$ is winning for $\text{sure-DirBWMP}(\alpha)$. If we subtract α from the payoff of every edge in $\mathcal{M}[W_B]$, then we obtain an MDP $(\mathcal{M}[W_B])_{-\alpha}$ in which every vertex is winning for $\text{sure-DirBWMP}(0)$. Using Lemma 20, we get that every vertex in $(\mathcal{M}[W_B])_{-\alpha}$ is winning for $\text{sure-DirFWMP}(\ell', 0)$. Adding back α to every edge payoff in $(\mathcal{M}[W_B])_{-\alpha}$ gives us that every vertex in $\mathcal{M}[W_B]$ is winning for $\text{sure-DirFWMP}(\ell', \alpha)$. Since $\mathcal{M}[W_B]$ is a subMDP of $\mathcal{M}$, we get that every vertex in W_B is also winning for $\text{sure-DirFWMP}(\ell', \alpha)$ in the original MDP $\mathcal{M}$.

For the converse direction, since $\text{DirFWMP}(\ell', \alpha)$ is also an objective that is closed under suffixes, we can follow the same proof as above starting with the winning region W_F for $\text{sure-DirFWMP}(\ell', \alpha)$ in $\mathcal{M}$ to show that every vertex in $\mathcal{M}$ that is winning for $\text{sure-DirFWMP}(\ell', \alpha)$ is also winning for $\text{sure-DirBWMP}(\alpha)$. $\blacktriangleleft$

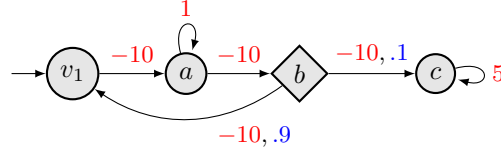
► **Lemma 22.** *For every vertex v in an MDP $\mathcal{M}$, the vertex v is winning for $\text{sure-DirBWMP}(\alpha)$ -positive-Reach(T) if and only if v is winning for $\text{sure-DirFWMP}(\ell', \alpha)$ -positive-Reach(T) for $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$.*

Proof. We recall the proof of correctness of Algorithm 3 for window length $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$, giving us an algorithm for $\text{sure-DirFWMP}(\ell', \alpha)$ -positive-Reach(T). We can use Lemma 21 in the proof of correctness of Algorithm 3 to replace every occurrence of “winning for $\text{sure-DirFWMP}(\ell', \alpha)$ ” with “winning for $\text{sure-DirBWMP}(\alpha)$ ”. This gives that the winning region for $\text{sure-DirBWMP}(\alpha)$ -positive-Reach(T) coincides with the winning region for $\text{sure-DirFWMP}(\ell', \alpha)$ -positive-Reach(T). $\blacktriangleleft$

We can follow the process described in the proof of Lemma 22 for the proofs of correctness of Algorithm 2 and Algorithm 1 to obtain Lemma 24 and Lemma 25 for $\text{sure-DirBWMP}(\alpha)$ -almost-sure-Büchi(T) and $\text{sure-BWMP}(\alpha)$ -almost-sure-BWMP(β) respectively. In order to show the correctness of the algorithm for $\text{sure-DirBWMP}(\alpha)$ -almost-sure-Büchi(T), we need to prove an analogous variant of Lemma 10 for BWMP, which we do in Lemma 23.

► **Lemma 23.** *For every vertex v in an MDP $\mathcal{M}$, if $\mathcal{P}_\circ$ has a strategy to reach a target $T \subseteq V$ from v with positive probability while simultaneously satisfying $\text{sure-DirBWMP}(\alpha)$, then $\mathcal{P}_\circ$ also has a strategy to reach T from v with positive probability in at most $|V|^2 \cdot (|V| \cdot w_{\max} + 1)$ steps while simultaneously satisfying $\text{sure-DirBWMP}(\alpha)$.*

Proof. From Lemma 21, we have that $\mathcal{P}_\circ$ can ensure that there exists a window length ℓ such that all α -windows in every outcome of the MDP close in at most ℓ steps, then there exists a window length $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$ such that $\mathcal{P}_\circ$ has a strategy such that all α -windows close in at most ℓ' steps in every outcome of the MDP. Then, we use Lemma 10 with window length ℓ' to get the desired result. $\blacktriangleleft$



■ **Figure 9** 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 30 is required for a strategy that is winning for sure-BWMP(0)-almost-sure-BWMP(5) from v_2 .

► **Lemma 24.** *For every vertex v in an MDP $\mathcal{M}$, the vertex v is winning for sure-DirBWMP(α)-almost-sure-Büchi(T) if and only if v is winning for sure-DirFWMP(ℓ' , α)-almost-sure-Büchi(T) for $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$.*

► **Lemma 25.** *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-DirFWMP(ℓ' , α)-almost-sure-DirFWMP(ℓ' , α) for $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$.*

Memory requirement and complexity for sure-almost-sure BWMP

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

Memory requirement: lower bound. It is known that memoryless strategies suffice for satisfying sure-BWMP(α) [11, Theorem 19] and for satisfying almost-sure-BWMP(α) [8, Theorem 6.2(b)]. In Example 26, 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 26.** Consider the MDP in Figure 9. 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 $\mathcal{P}_\circ$ 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 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 30 is winning for sure-BWMP(0)-almost-sure-BWMP(5).

This example can be generalized to an MDP $\mathcal{M}_n$ with $n + 3$ vertices $\{v_n, v_{n-1}, \dots, v_1, a, b, c\}$ 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 $-10 = -w_{\max}$, 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 $\mathcal{M}_n$ requires memory of size at least $(n + 2) \cdot w_{\max}$, which is $\Omega(|V| \cdot w_{\max})$. $\lrcorner$

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

In: An MDP $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, 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: $W_{S_\alpha} \leftarrow \text{SureFWMP}(\mathcal{M}, \ell, \alpha)$
 - 2: $W_{AS_\beta} \leftarrow \text{AlmostSureFWMP}(\mathcal{M}[W_{S_\alpha}], \ell, \beta)$
 - 3: **return** W_{AS_β}
-

Memory requirement: upper bound. By Lemma 25, 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 $\ell' = |V| \cdot (|V| \cdot w_{\max} + 1)$, and moreover, this strategy σ_F is also winning strategy for sure-almost-sure BWMP from v . From Theorem 9, 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 [11, Theorem 19], we have that satisfaction of sure-BWMP(α) and of sure-DirBWMP(α) are both in $\text{NP} \cap \text{coNP}$. 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 $\text{PTIME}^{\text{NP} \cap \text{coNP}}$, which is the same as $\text{NP} \cap \text{coNP}$.

► **Theorem 27.** *The sure-almost-sure problem for the BWMP objective is in $\text{NP} \cap \text{coNP}$. 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 objectives is given in the appendix. It is obtained by simply replacing SureFWMP and AlmostSureFWMP with SureBWMP and AlmostSureBWMP respectively.

Description of Algorithm 4. We first compute the set W_{S_α} of vertices that are winning for sure-FWMP(ℓ, α) in $\mathcal{M}$ using [11, Algorithm 1]. Then, in the subMDP $\mathcal{M}[W_{S_\alpha}]$ induced by W_{S_α} , we compute using [8, Algorithm 1] the set W_{AS_β} of vertices that are winning for almost-sure-FWMP(ℓ, β), and we return this set W_{AS_β} . We briefly describe [8, Algorithm 1]: it first computes the set W_{S_β} of all vertices in $\mathcal{M}[W_{S_\alpha}]$ that are winning for sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S_\alpha}]$ (once again using [11, Algorithm 1]). Then, the algorithm performs the MEC decomposition of $\mathcal{M}[W_{S_\alpha}]$, and labels all the MECs that have a non-empty intersection with W_{S_β} as *good MECs*. The algorithm finally returns all vertices in $\mathcal{M}[W_{S_\alpha}]$ from which $\mathcal{P}_\circ$ can reach the set of good MECs with probability 1.

Correctness of Algorithm 4. It is useful to note that since FWMP(ℓ, β) is prefix-independent, from Proposition 2, the set W_{AS_β} that is the winning region for almost-sure-FWMP(ℓ, β) in $\mathcal{M}$ induces a subMDP of $\mathcal{M}$. Thus, if the token reaches W_{AS_β} , then $\mathcal{P}_\circ$ has a strategy that keeps the token in W_{AS_β} . We also recall that we have $\alpha < \beta$.

► **Theorem 28.** *Algorithm 4 computes the winning region for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in $\mathcal{M}$.*

Proof. ($\Rightarrow$) We show that if a vertex v belongs to the set $W_{AS\beta}$ returned by Algorithm 4, then v is winning for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in $\mathcal{M}$. That is, we show that for all $\varepsilon > 0$, player $\mathcal{P}_\circ$ has a strategy σ_ε from v that satisfies both sure-FWMP(ℓ, α) and probability($1 - \varepsilon$)-FWMP(ℓ, β).

Recall that the set $W_{AS\beta}$ is the winning region for almost-sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S\alpha}]$. Before we describe σ_ε , we describe a strategy σ_{AS} that is winning for almost-sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S\alpha}]$ from all vertices in $W_{AS\beta}$. We recall the following from [8, Lemma 6.1]: if the winning region for almost-sure-FWMP(ℓ, β) in an MDP is non-empty, then the winning region for sure-FWMP(ℓ, β) in the MDP is also non-empty. Since we started with $W_{AS\beta}$ non-empty, the winning region for sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S\alpha}]$ is also non-empty (we call this region $W_{S\beta}$). Moreover, the set $W_{AS\beta}$ is precisely the set of all vertices in $\mathcal{M}[W_{S\alpha}]$ from where $\mathcal{P}_\circ$ has a strategy to reach $W_{S\beta}$ with probability 1. The strategy σ_{AS} is defined as follows: if the token is in $W_{AS\beta} \setminus W_{S\beta}$, then follow an almost-sure attractor strategy to $W_{S\beta}$, and if the token is in $W_{S\beta}$, then follow a winning strategy for sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S\beta}]$.

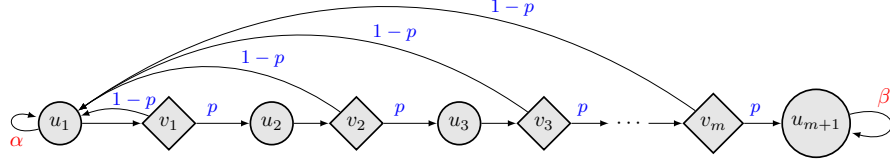
If $\mathcal{P}_\circ$ follows this strategy σ_{AS} , then with probability 1, the token eventually reaches $W_{S\beta}$ and stays there forever, and eventually all β -windows close in at most ℓ steps. Thus, this strategy is winning for almost-sure-FWMP(ℓ, β), and since $\alpha < \beta$, we have that this strategy is also winning for almost-sure-FWMP(ℓ, α). However, this strategy may not be winning for sure-FWMP(ℓ, α) as there may exist an outcome of σ_{AS} where the token gets stuck in $W_{AS\beta} \setminus W_{S\beta}$ forever and infinitely many open α -windows of length ℓ occur in the outcome. We modify the strategy σ_{AS} to construct a strategy σ_ε that is winning for both sure-FWMP(ℓ, α) and probability($1 - \varepsilon$)-FWMP(ℓ, β).

Let N be a sufficiently large integer that depends on $|W_{AS\beta}|$ (the number of vertices in $W_{AS\beta}$), $\mathbb{P}_{\min}$ (the minimum edge probability in $W_{AS\beta}$), and ε . The strategy σ_ε is as follows: for the first N steps of the play, the strategy σ_ε follows σ_{AS} , that is, if the token is in $W_{AS\beta} \setminus W_{S\beta}$, then σ_ε follows an almost-sure-attractor strategy to $W_{S\beta}$, and if the token is in $W_{S\beta}$, then σ_ε follows a winning strategy for sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S\beta}]$. After N steps of the play have elapsed, if the token is in $W_{S\beta}$, then σ_ε continues to follow the winning strategy for sure-FWMP(ℓ, β) in the $\mathcal{M}[W_{S\beta}]$ for the rest of the play, and otherwise the token is in $W_{AS\beta} \setminus W_{S\beta}$, in which case σ_ε switches to a strategy that is winning for sure-FWMP(ℓ, α) in $\mathcal{M}[W_{S\alpha}]$. Such a winning strategy exists from every vertex in $W_{S\alpha}$, and thus such a winning strategy also exists from all vertices in $W_{AS\beta} \setminus W_{S\beta}$.

If a large enough N is chosen, then with probability at least $1 - \varepsilon$, the token reaches $W_{S\beta}$ in the first N steps following which satisfaction of FWMP(ℓ, β) is guaranteed, and even if the token does not reach $W_{S\beta}$, all outcomes of this strategy satisfy FWMP(ℓ, α). Thus, we have that this strategy σ_ε is winning for sure-FWMP(ℓ, α) and probability($1 - \varepsilon$)-FWMP(ℓ, β). Since we can construct such a strategy σ_ε for every $\varepsilon > 0$, we have that v is winning for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in $\mathcal{M}$.

($\Leftarrow$) We show that if v does not belong to the set $W_{AS\beta}$ returned by Algorithm 4, then v is not winning for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in $\mathcal{M}$. If $v \notin W_{AS\beta}$, then exactly one of the following hold: either $v \in V \setminus W_{S\alpha}$ or $v \in W_{S\alpha} \setminus W_{AS\beta}$.

- Suppose $v \in V \setminus W_{S\alpha}$. Then, the vertex v is not winning for sure-FWMP(ℓ, α) in $\mathcal{M}$, and hence it is also not winning for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in $\mathcal{M}$.
- Suppose $v \in W_{S\alpha} \setminus W_{AS\beta}$. Starting from v , if $\mathcal{P}_\circ$ ever moves the token out of $W_{S\alpha}$ into $V \setminus W_{S\alpha}$, then $\mathcal{P}_\circ$ cannot ensure satisfaction of sure-FWMP(ℓ, α). On the other hand, if



■ **Figure 10** The self-loop on u_1 has payoff α and the self-loop on edge u_{m+1} has payoff β . Every other edge has payoff $\alpha - 1$.

$\mathcal{P}_\circ$ never moves the token out of W_{S_α} , then since v is not in the winning region W_{AS_β} for almost-sure-FWMP(ℓ, β) in $\mathcal{M}[W_{S_\alpha}]$, there exists some $0 < \varepsilon' \leq 1$ such that $\mathcal{P}_\circ$ cannot satisfy probability($1 - \varepsilon'$)-FWMP(ℓ, β) from v . For all $\varepsilon < \varepsilon'$, player $\mathcal{P}_\circ$ does not have a strategy to keep the token in W_{S_α} and also satisfy probability($1 - \varepsilon$)-FWMP(ℓ, β), in which case the limit-sure-FWMP(ℓ, β) is not satisfied from v . Thus, we have that v is not winning for sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) in $\mathcal{M}$.

This shows the correctness of Algorithm 4, and also shows how to construct strategies for the satisfaction of sure-FWMP(ℓ, α)-limit-sure-FWMP(ℓ, β) for $\mathcal{P}_\circ$ from vertices in W_{AS_β} . ◀

In the rest of this section, we discuss the memory requirement and complexity for the sure-limit-sure satisfaction problems for the FWMP(ℓ) and BWMP objectives.

Memory requirement. We consider a winning strategy σ_ε of $\mathcal{P}_\circ$ for sure-FWMP(ℓ, α)-probability($1 - \varepsilon$)-FWMP(ℓ, β) as described in the proof of Theorem 28. The strategy σ_ε is defined in two phases: in the first phase it follows a memoryless almost-sure attractor strategy to W_{S_β} until either the token reaches W_{S_β} 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_{S_β} or $W_{AS_\beta} \setminus W_{S_\beta}$ respectively at the end of the first phase). A counter of size N can count the length of the first phase of the strategy, and memory of size ℓ is sufficient for a winning strategy for sure-FWMP(ℓ, β) or sure-FWMP(ℓ, α) [18, Theorem 4.4] in the second phase. Thus, we get an upper bound of $N + \ell$ on the memory requirement of σ_ε .

We show lower and upper bounds for N , respectively $\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{|V|}}\right)$ and $\mathcal{O}\left(\frac{|V| \cdot \log(1/\varepsilon)}{(\mathbb{P}_{\min})^{|V|}}\right)$ for the minimum number of steps that the almost-sure attractor strategy to W_{S_β} must be followed for in order to ensure that W_{S_β} is reached with probability at least $1 - \varepsilon$. We show the bounds for a family of conservative MDPs, i.e., given $|V|$ and $\mathbb{P}_{\min}$, the family consists of the MDP in which the probability x_N of reaching W_{S_β} from the initial vertex in at most N visits to probabilistic vertices is the minimum amongst all MDPs with $|V|$ vertices and minimum probability $\mathbb{P}_{\min}$. If the bounds hold for this family of MDPs, then they hold for all MDPs. An MDP $\mathcal{M}_{m,p}$ of this family is shown in Figure 10 with $|V| = 2m + 1$ vertices and $\mathbb{P}_{\min} = p$. The MDP $\mathcal{M}_{m,p}$ consists of vertices $u_1, u_2, \dots, u_{m+1}$ belonging to $\mathcal{P}_\circ$ and vertices $v_1, v_2, \dots, v_m$ that are probabilistic. For all $1 \leq i \leq m$, we have that vertex u_i has an out-neighbour of v_i , and vertex v_i has out-neighbours u_i and u_{i+1} with probabilities $1 - p$ and p respectively. Vertices u_1 and u_{m+1} have self-loops with payoffs α and β respectively. All other edges in $\mathcal{M}_{m,p}$ have payoff $\alpha - 1$.

For all $\ell \geq 1$, the set W_{S_β} consists of only u_{m+1} . In order to reach u_{m+1} starting from u_1 , the “right” edge (v_i, v_{i+1}) must be seen at m consecutive visits to the probabilistic vertices $v_1, v_2, \dots, v_m$. If at any probabilistic vertex along the way the token moves to u_1 , then the token loses all progress towards reaching u_{m+1} and has to start over again. We want to find

bounds on the number of times N the MDP $\mathcal{M}_{m,p}$ visits probabilistic vertices for to ensure that u_{m+1} is reached with probability at least $1 - \varepsilon$. Formally, we would like to find N in terms of $|V|$, $\mathbb{P}_{\min}$, and ε such that $x_N \geq 1 - \varepsilon$ holds. It can be seen that the probability x_N of reaching u_{m+1} starting from u_1 in $\mathcal{M}_{m,p}$ after visiting probabilistic vertices at most N times is equal to the following: given a coin that lands heads with probability p and tails with probability $1 - p$, the probability that a sequence of N tosses of this coin contains a streak of at least m consecutive heads. We can compute x_N using the recurrence relation

$$\begin{aligned} x_N &= p^m + \sum_{i=1}^m p^{i-1} \cdot (1-p) \cdot x_{N-i} \\ &= p^m + (1-p) \cdot x_{N-1} + p \cdot (1-p) \cdot x_{N-2} + \cdots + p^{m-1} \cdot (1-p) \cdot x_{N-m} \end{aligned}$$

for $N \geq m$, and with initial values $x_0 = x_1 = \cdots = x_{m-1} = 0$. Intuitively, a sequence of N coin tosses that contains a streak of m heads either does not contains a tail in the first m tosses (which happens with probability p^m), or the first tail appears at the i^{th} position (this happens with probability $p^{i-1} \cdot (1-p)$, following which there is a streak of m heads in the remaining $N-i$ tosses with probability x_{N-i}) for $1 \leq i \leq m$. This recurrence relation, however, does not have a nice closed form, so we find approximate lower and upper bounds for N such that $x_N \geq 1 - \varepsilon$.

- *Upper bound:* We divide the sequence of N coin tosses into blocks of length m each (we assume that m divides N). We define q to be probability that at least one of the N/m blocks consists of all heads. If one of the blocks is all heads, then this is a witness for the streak of at least m consecutive heads in the sequence of N coin tosses, and thus, we have that $x_N \geq q$. For each block, the probability that the block contains at least one tail is $(1 - p^m)$, independent of the other blocks. The probability q that at least one of the blocks is all heads is $1 - (1 - p^m)^{N/m}$. We get that $x_N \geq 1 - (1 - p^m)^{N/m}$. Further, if $1 - (1 - p^m)^{N/m} \geq 1 - \varepsilon$, then we have the following:

$$\begin{aligned} &1 - (1 - p^m)^{N/m} \geq 1 - \varepsilon \\ \iff &(1 - p^m)^{N/m} \leq \varepsilon \\ \iff &\frac{1}{(1 - p^m)^{N/m}} \geq \frac{1}{\varepsilon} \\ \iff &\frac{N}{m} \log(1/(1 - p^m)) \geq \log(1/\varepsilon) \\ \iff &N \geq \frac{m \cdot \log(1/\varepsilon)}{\log(1/(1 - p^m))} \end{aligned}$$

Moreover, for values of p not close to 1 (say $0 < p \leq 0.5$), we have that $\frac{1}{\log(1/(1 - p^m))}$ is closely bounded above by $\frac{2.4}{p^m}$. Thus, if $N \geq \frac{2.4 \cdot m \cdot \log(1/\varepsilon)}{p^m}$, then $N \geq \frac{m \cdot \log(1/\varepsilon)}{\log(1/(1 - p^m))}$, which in turn implies $1 - (1 - p^m)^{N/m} \geq 1 - \varepsilon$, and the desired inequality $x_N \geq 1 - \varepsilon$ holds since $x_N \geq 1 - (1 - p^m)^{N/m}$. Since $m = \frac{|V|-1}{2}$ and $p = \mathbb{P}_{\min}$ for $\mathcal{M}_{m,p}$, we have that for all MDPs with $|V|$ vertices and minimum probability $\mathbb{P}_{\min}$, choosing a value of N that is at least $\mathcal{O}\left(\frac{|V| \cdot \log(1/\varepsilon)}{(\mathbb{P}_{\min})^{|V|}}\right)$ is sufficient to ensure that $W_{S\beta}$ is reached with probability at least $1 - \varepsilon$ after N visits to probabilistic vertices, and this gives us an upper bound for N .

- *Lower bound:* If there is a streak of at least m consecutive heads in the sequence of N coin tosses, then the streak must begin at the i^{th} coin toss for some $1 \leq i \leq N - m + 1$. Moreover, the probability of starting a streak of m heads at the i^{th} coin toss is p^m , independent of i . Using the union bound, we get that x_N is bounded above by $(N - m + 1) \cdot p^m$, which

is bounded above by $N \cdot p^m$. That is, we have $N \cdot p^m \geq x_N$. Thus, in order to satisfy $x_N \geq 1 - \varepsilon$, it is necessary that $N \cdot p^m \geq 1 - \varepsilon$ holds, that is $N \geq \frac{1-\varepsilon}{p^m}$. Substituting $m = \frac{|V|-1}{2}$ and $p = \mathbb{P}_{\min}$ gives us a lower bound $\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{|V|}}\right)$ for N .

Complexity of SLS-FWMP(ℓ). Since sure satisfaction and almost-sure satisfaction of the FWMP(ℓ) objectives can both be checked in polynomial time [8, 11], we have that Algorithm 4 runs in polynomial time, and we have the following result.

► **Theorem 29.** *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 $\mathcal{O}\left(\frac{|V| \cdot \log(1/\varepsilon)}{(\mathbb{P}_{\min})^{|V|}} + \ell\right)$.*

Complexity of SLS-BWMP. Since sure satisfaction and almost-sure satisfaction of the BWMP objectives are in $\text{NP} \cap \text{coNP}$ [8, 11], and the algorithm for sure-limit-sure satisfaction for BWMP makes a polynomial number of calls to algorithms for sure satisfaction and almost-sure satisfaction of BWMP objectives, we have that sure-limit-sure satisfaction of BWMP objectives is in $\text{PTIME}^{\text{NP} \cap \text{coNP}}$, which is the same as $\text{NP} \cap \text{coNP}$. Moreover, an ε -optimal strategy for the sure-limit-sure satisfaction of BWMP objectives follows a similar two-phase structure as that of sure-limit-sure satisfaction of FWMP(ℓ) objectives. The first phase requires a counter of size N (with the same bounds as before), and memoryless strategies exist for the satisfaction of sure-BWMP(α) or sure-BWMP(β) [11, Theorem 19] in the second phase. Thus, we have that memory of size $N + 1$ suffices for an ε -optimal strategy.

► **Theorem 30.** *Sure-limit-sure satisfaction of BWMP objectives is in $\text{NP} \cap \text{coNP}$. The memory required is at least $\Omega\left(\frac{1-\varepsilon}{(\mathbb{P}_{\min})^{|V|}}\right)$ and at most $\mathcal{O}\left(\frac{|V| \cdot \log(1/\varepsilon)}{(\mathbb{P}_{\min})^{|V|}}\right)$.*

6 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.

As 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 . 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.

References

- 1 K. R. Apt and E. Grädel. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011. doi:10.1017/CB09780511973468.

- 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 P. Billingsley. *Probability and Measure*. John Wiley and Sons, second edition, 1986.
- 7 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.
- 8 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.
- 9 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.
- 10 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.
- 11 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.
- 12 K. Chatterjee and M. Henzinger. Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-Component Decomposition. *J. ACM*, 61(3):15:1–15:40, 2014. doi:10.1145/2597631.
- 13 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.
- 14 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, Jul 2017. doi:10.23638/LMCS-13(2:15)2017.
- 15 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.
- 16 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.
- 17 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.
- 18 L. Doyen, P. Gaba, and S. Guha. Stochastic Window Mean-Payoff Games. *Logical Methods in Computer Science*, Volume 21, Issue 2, Jun 2025. doi:10.46298/lmcs-21(2:19)2025.
- 19 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.
- 20 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.
- 21 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.
- 22 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.

- 23 D. A. Martin. The Determinacy of Blackwell Games. *The Journal of Symbolic Logic*, 63(4):1565–1581, 1998. doi:10.2307/2586667.
- 24 M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.
- 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.

A

 Appendix: Additional algorithms and results

Algorithm 5

 SureAttr($\mathcal{M}, T$)

In: An MDP $\mathcal{M} = ((V, E), (V_{\bigcirc}, V_{\bigtriangledown}), \mathbb{P}, w)$ and a target set $T \subseteq V$

Out: SureAttr(T), the sure-attractor of T

```

1:  $X \leftarrow T$ 
2: for all  $v \in V_{\bigcirc}$  do
3:   if  $N^+(v) \cap T \neq \emptyset$  then                                 $\triangleright$  at least one out-neighbour in  $T$ 
4:      $X \leftarrow X \cup \{v\}$ 
5: for all  $v \in V_{\bigtriangledown}$  do
6:   if  $N^+(v) \subseteq T$  then                                           $\triangleright$  all out-neighbours in  $T$ 
7:      $X \leftarrow X \cup \{v\}$ 
8: if  $X \setminus T = \emptyset$  then
9:   return  $T$ 
10: else
11:   return SureAttr( $\mathcal{M}, X$ )

```

Algorithm 6

 SureSafe($\mathcal{M}, T$)

In: An MDP $\mathcal{M} = ((V, E), (V_{\bigcirc}, V_{\bigtriangledown}), \mathbb{P}, w)$ and a safe set $T \subseteq V$

Out: The winning region for sure-Safe(T) in $\mathcal{M}$

```

1:  $X \leftarrow \emptyset$ 
2: for all  $v \in V_{\bigtriangledown}$  do
3:   if  $N^+(v) \cap (V \setminus T) \neq \emptyset$  then                         $\triangleright$  at least one out-neighbour in  $V \setminus T$ 
4:      $X \leftarrow X \cup \{v\}$ 
5: for all  $v \in V_{\bigcirc}$  do
6:   if  $N^+(v) \subseteq V \setminus T$  then                                     $\triangleright$  all out-neighbours in  $V \setminus T$ 
7:      $X \leftarrow X \cup \{v\}$ 
8: if  $X = \emptyset$  then
9:   return  $T$ 
10: else
11:   return SureSafe( $\mathcal{M}, T \setminus X$ )

```

► **Theorem 31.** Algorithm 7 computes the winning region for sure-BWMP(α)-almost-sure-BWMP(β) in $\mathcal{M}$.

► **Theorem 32.** Algorithm 8 computes the winning region for sure-DirBWMP(α)-almost-sure-Büchi(T).

► **Theorem 33.** Algorithm 9 computes the winning region for sure-DirBWMP(α)-positive-Reach(T) in $\mathcal{M}$.

Algorithm 7 SureAlmostSureBWMP($\mathcal{M}, \alpha, \beta$)

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

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

```

1:  $W_{S\alpha} \leftarrow \text{SureBWMP}(\mathcal{M}, \alpha)$ 
2:  $W_{S\beta} \leftarrow \text{SureBWMP}(\mathcal{M}[W_{S\alpha}], \beta)$ 
3:  $W \leftarrow W_{S\beta}$ 
4: repeat
5:    $P \leftarrow X_{>0}^{\leq 1}(\mathcal{M}[W_{S\alpha}], W)$ 
6:    $W_{\text{sdab}} \leftarrow \text{SureDirBWMPASBüchi}(\mathcal{M}[W_{S\alpha} \setminus W], \alpha, P)$ 
7:    $W \leftarrow W \cup W_{\text{sdab}}$ 
8:    $W \leftarrow \text{SureAttr}(\mathcal{M}[W_{S\alpha}], W)$ 
9: until  $W_{\text{sdab}} = \emptyset$ 
10: return  $W$ 

```

Algorithm 8 SureDirBWMPASBüchi($\mathcal{M}, \alpha, T$)

In: An MDP $\mathcal{M} = ((V, E), (V_{\bigcirc}, V_{\bigtriangleup}), \mathbb{P}, w)$, threshold α , and target set T

Out: The winning region for sure-DirBWMP(α)-almost-sure-Büchi(T) in $\mathcal{M}$

```

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

```

Algorithm 9 SureDirBWMPPosReach($\mathcal{M}, \alpha, T$)

In: An MDP $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, threshold α , and target set T

Out: The winning region for sure-DirBWMP(α)-positive-Reach(T) in $\mathcal{M}$

```

1:  $W_{\text{SD}\alpha} \leftarrow \text{SureDirBWMP}(\mathcal{M}, \alpha)$ 
2:  $\mathcal{M} \leftarrow \mathcal{M}[W_{\text{SD}\alpha}]$ ,  $T \leftarrow T \cap W_{\text{SD}\alpha}$ 
3:  $R \leftarrow T$ ,  $E_g, E_b \leftarrow \emptyset$ 
4: while  $\{e = (s, t) \in E \mid s \in V \setminus T, t \in R, \text{ and } e \in E \setminus (E_g \cup E_b)\} \neq \emptyset$  do
5:   if IsGoodEdge( $e, E_g$ ) then
6:      $E_g \leftarrow E_g \cup \{e\}$ ,  $R \leftarrow R \cup \{s\}$ ,  $E_b \leftarrow \emptyset$ 
7:   else
8:      $E_b \leftarrow E_b \cup \{e\}$ 
9: return  $R$ 

10: procedure IsGoodEdge( $e = (s, t), E_g$ )
11:   Construct MDP  $\mathcal{M}'_e$  in the following way:
12:    $V \leftarrow$  the set of vertices in  $\mathcal{M}$ 
13:    $\tilde{R} \leftarrow \{\tilde{v} \mid v \in R\}$ 
14:   The set of vertices of  $\mathcal{M}'_e$  is  $V \cup \tilde{R} \cup \{\hat{s}\}$ .
   The kind of each vertex is preserved from  $\mathcal{M}$ .

15:    $E \leftarrow$  the set of edges in  $\mathcal{M}$ 
16:    $\tilde{E}_g \leftarrow \{(\tilde{u}, \tilde{v}) \mid u, v \in R, (u, v) \in E_g\}$ 
17:    $\tilde{E}_\diamond \leftarrow \{(\tilde{u}, v) \mid u \in (R \setminus T) \cap V_\diamond, v \in V, (u, v) \in E \setminus E_g\}$ 
18:    $\hat{E}_\diamond \leftarrow$  if  $s \in V_\circ$  then  $\emptyset$  else  $\{(\hat{s}, v) \mid (s, v) \in E \text{ and } v \neq t\}$ 
19:   The set of edges of  $\mathcal{M}'_e$  is  $E \cup \tilde{E}_g \cup \tilde{E}_\diamond \cup \{(\hat{s}, \tilde{t})\} \cup \hat{E}_\diamond$ .
   The probability and payoff of each edge is preserved from  $\mathcal{M}$ .

20:    $\mathcal{M}_e \leftarrow$  For all  $v \in T$ , identify  $\tilde{v}$  with  $v$  in  $\mathcal{M}'_e$ 
21:   return true iff  $\hat{s}$  is winning for sure-DirBWMP( $\alpha$ ) in  $\mathcal{M}_e$ 

```

Algorithm 10 SureLimitSureBWMP($\mathcal{M}, \alpha, \beta$)

In: An MDP $\mathcal{M} = ((V, E), (V_\circ, V_\diamond), \mathbb{P}, w)$, sure threshold α , and limit-sure threshold β

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

```

1:  $W_{\text{S}\alpha} \leftarrow \text{SureBWMP}(\mathcal{M}, \alpha)$ 
2:  $W_{\text{AS}\beta} \leftarrow \text{AlmostSureBWMP}(\mathcal{M}[W_{\text{S}\alpha}], \beta)$ 
3: return  $W_{\text{AS}\beta}$ 

```
