Life is Random, Time is Not:
Markov Decision Processes with Window Objectives[*]

Thomas Brihaye[1], Florent Delgrange[1,3], Youssouf Oualhadj[4], and Mickael Randour[1,2]

[1] UMONS – Université de Mons, Belgium
[2] F.R.S.-FNRS, Belgium
[3] RWTH Aachen, Germany
[4] LACL – UPEC, France

Abstract. The window mechanism was introduced by Chatterjee et al. [18] to strengthen classical game objectives with time bounds. It permits to synthesize system controllers that exhibit acceptable behaviors within a configurable time frame, all along their infinite execution, in contrast to the traditional objectives that only require correctness of behaviors in the limit. The window concept has proved its interest in a variety of two-player zero-sum games, thanks to the ability to reason about such time bounds in system specifications, but also the increased tractability that it usually yields.

In this work, we extend the window framework to stochastic environments by considering the fundamental *threshold probability problem* in Markov decision processes for window objectives. That is, given such an objective, we want to synthesize strategies that guarantee satisfying runs with a given probability. We solve this problem for the usual variants of window objectives, where either the time frame is set as a parameter, or we ask if such a time frame exists. We develop a generic approach for window-based objectives and instantiate it for the classical mean-payoff and parity objectives, already considered in games. Our work paves the way to a wide use of the window mechanism in stochastic models.

1   Introduction

Game-based models for controller synthesis. *Two-player zero-sum games* [30,37] and *Markov decision processes* (MDPs) [28,4,38] are two popular frameworks to model decision making in adversarial and uncertain environments respectively. In the former, a system controller and its environment compete antagonistically, and synthesis aims at building strategies for the controller that ensure a specified behavior *against all possible strategies of the environment*. In the latter, the system is faced with a given stochastic model of its environment, and the focus is on satisfying a given level of expected performance, or a *specified behavior with a sufficient probability*. Classical objectives studied in both settings notably include *parity*, a canonical way of encoding ω-regular specifications, and *mean-payoff*, which evaluates the average payoff per transition in the limit of an infinite run in a weighted graph.

Window objectives in games. The traditional parity and mean-payoff objectives share two shortcomings. First, they both reason about infinite runs *in their limit*. While this elegant abstraction yields interesting theoretical properties and makes for robust interpretation, it is often beneficial in practical applications to be able to specify a *parameterized time frame* in which an acceptable behavior should be witnessed. Second, both parity and mean-payoff games belong to UP ∩ coUP [36,29], but despite recent breakthroughs [17,23], they are still not known to be in P. Furthermore, the latest results [22,27] indicate that all existing algorithmic approaches share inherent limitations that prevent inclusion in P.

Window objectives address the time frame issue as follows. In their *fixed* variant, they consider a window of size bounded by $\lambda \in \mathbb{N}_0$ (given as a parameter) sliding over an infinite run and declare this run to be winning if, in all positions, the window is such that the (mean-payoff or parity) objective is locally satisfied. In their *bounded* variant, the window size is not fixed a priori, but a run is winning if there exists a bound

λ for which the condition holds. Window objectives have been considered both in *direct* versions, where the window property must hold from the start of the run, and *prefix-independent* versions, where it must hold from some point on. Window games were initially studied for mean-payoff [18] and parity [15]. They have since seen diverse extensions and applications: e.g., [5,3,12,16,34,40].

Window objectives in MDPs. Our goal is to lift the theory of window games to the stochastic context. With that in mind, we consider the canonical *threshold probability problem*: given an MDP, a window objective defining a set of acceptable runs E, and a probability threshold α, we want to decide if there exists a controller *strategy* (also called *policy*) to achieve E with probability at least α. It is well-known that many problems in MDPs can be reduced to threshold problems for appropriate objectives: e.g., maximizing the *expectation* of a prefix-independent function boils down to maximizing the probability to reach the best end-components for that function (see examples in [4,14,39]).

Example. Before going further, let us consider an example. Take the MDP depicted in Fig. 1(a): circles depict states and dots depict actions, labeled by letters. Each action yields a probability distribution over successor states: for example, action b leads to s_2 with probability 0.5 and s_3 with the same probability. This MDP is actually a *Markov chain* (MC) as the controller has only one action available in each state: this process is purely stochastic.

Fig. 1. Simple Markov chain where parity is surely satisfied but all window parity objectives have probability zero.

We consider the *parity* objective here: we associate an non-negative integer priority with each state, and a run is winning if the minimum one amongst those seen infinitely often is even. Clearly, any run in this MC is winning: either it goes through s_3 infinitely often and the minimum priority is 0, or it does not, and the minimum priority seen infinitely often is 2. Hence the controller not only wins *almost-surely* (with probability one), but even *surely* (on all runs).

Now, consider the *window parity* objective that informally asks for the minimum priority inside a window of size bounded by λ to be even, with this window sliding all along the infinite run. Fix any $\lambda \in \mathbb{N}_0$. It is clear that every time s_1 is visited, there will be a fixed strictly positive probability $\varepsilon > 0$ of not seeing 0 before λ steps: this probability is $1/2^{\lambda-1}$. Let us call this seeing a *bad window*. Since we are in a *bottom strongly connected component* of the MC, we will almost-surely visit s_1 infinitely often [4]. Using classical probability arguments (Borel-Cantelli), one can easily be convinced that the probability to see bad windows infinitely often is one. Hence the probability to win the window parity objective is zero. This canonical example illustrates the difference between traditional parity and window parity: the latter is more restrictive as it asks for a strict bound on the time frame in which each odd priority should be answered by a smaller even priority.

Note that in practice, such a behavior is often wished for. For example, consider a computer server having to grant requests to clients. A classical parity objective can encode that requests should eventually be granted. However, it is clear that in a desired controller, requests should not be placed on hold for an arbitrarily long time. The existence of a finite bound on this holding time can be modeled with a *bounded* window parity objective, while a specific bound can also be set as a parameter using a *fixed* window parity objective.

Our contributions. We study the *threshold probability problem* in MDPs for window objectives based on *parity* and *mean-payoff*, two prominent formalisms in qualitative and quantitative (resp.) analysis of systems. We consider the different variants of window objectives mentioned above: fixed vs. bounded, direct vs. prefix-independent. A nice feature of our approach is that we provide a *unified view* of parity and mean-payoff window objectives: *our algorithm can actually be adapted for any window-based objective* if an appropriate black-box is provided for a restricted sub-problem. This has two advantages: (i) conceptually, our approach permits a *deeper understanding of the essence of the window mechanism*, not biased by technicalities of the specific underlying objective; (ii) our framework can *easily* be *extended* to other objectives for which a window version could be defined. This point is of great practical interest too, as it opens the perspective of a modular, generic software tool suite for window objectives.
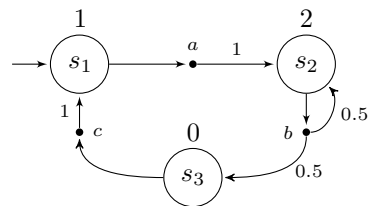
	parity		mean-payoff	
	complexity	memory	complexity	memory
direct fixed window (DFW)	P-c.	polynomial	EXPTIME/PSPACE-h.	pseudo-polynomial
fixed window (FW)			P-c.	polynomial
bounded window (BW)		memoryless	NP ∩ coNP	memoryless

Table 1. Complexity of the threshold probability problem for window objectives in MDPs and memory requirements. All memory bounds are tight and *pure* strategies always suffice. For mean-payoff, the PSPACE-hardness holds even for acyclic MDPs, and the bounded case is as hard as mean-payoff games. All results are new.

We give an overview of our results in Table 1. For the sake of space, we use acronyms below: DFW for direct fixed window, FW for (prefix-independent) fixed window, DBW for direct bounded window, and BW for (prefix-independent) bounded window. Our main contributions are as follows.

1. We solve DFW MDPs through reductions to safety MDPs over *well-chosen unfoldings*. This results in polynomial-time and pseudo-polynomial-time algorithms for the parity and mean-payoff variants respectively (Thm. 9). We prove these complexities to be almost tight (Thm. 10), the most interesting case being the PSPACE-hardness of DFW mean-payoff objectives, even in the case of *acyclic* MDPs.
   We also show that no upper bound can be established on the window size needed to win in general (Sect. 3.3), *in stark contrast to the two-player games situation* (Sect. 3.2).
2. We use similar reductions to prove that *finite memory* suffices in the prefix-independent case (Thm. 8). Yet, in this case, we can do better than using the unfoldings to solve the problem. We start by studying *end-components* (ECs), the crux for all prefix-independent objectives in MDPs: we show that ECs can be classified based on their *two-player zero-sum game interpretation* (Sect. 5). Using the result on finite memory, we prove that in ECs classified as *goods*, almost-sure satisfaction of window objectives can be ensured, whereas it is impossible to satisfy them with non-zero probability in the other ECs (Lem. 18). We also establish tight complexity bounds for this classification problem (Thm. 23). This *EC classification* is – both conceptually and complexity-wise – the cornerstone to deal with general MDPs.
3. Our general algorithm is developed in Sect. 6: we prove P-completeness for all prefix-independent variants but for the BW mean-payoff one (Thm. 25 and Thm. 26), where we show that the problem is in NP∩coNP and as hard as mean-payoff games, a canonical "hard" problem for that complexity class.
4. For all variants, we prove *tight memory bounds*: see Thm. 9 and Thm. 10 for the direct fixed variants, Thm. 25 and Thm. 26 for the prefix-independent fixed and bounded ones. In all cases, *pure* strategies (i.e., without randomness) suffice.
5. We leave out DBW objectives from our analysis as we show they are not well-behaved. We illustrate their behavior in Sect. 3.3 and discuss their pitfalls in Sect. 7.

Along the way, we develop several side results that help drawing a *line between MDPs and games* w.r.t. window objectives: e.g., whether or not a uniform bound exists in the bounded case (Sect. 3.3). As stated above, our approach is also generic and may be easily extended to other window-based objectives.

Comments on our results. In the game setting, window objectives are all in polynomial time, except for the BW mean-payoff variant, in NP ∩ coNP. Despite clear differences in behaviors, the situation is almost the same here. The only outlier case is the DFW mean-payoff one, whose complexity rises significantly. As we show in Thm. 10, the loss of prefix-independence permits to emulate shortest path problems on MDPs that are famously hard to solve efficiently (e.g., [32,39,9,33]). In the almost-sure case, however, DFW mean-payoff MDPs collapse to P (Rmk. 12).

In games, window objectives permit to avoid long-standing NP ∩ coNP complexity barriers for parity [15] and mean-payoff [18]. Since both are known to be in P for the threshold probability problem in MDPs [21,39], the main interest of window objectives resides in their *modeling power*. Still, they may turn out to be

more efficient in practice too, as polynomial-time algorithms for parity and mean-payoff, based on linear programming, are often forsaken in favor of exponential-time value or strategy iteration ones (e.g., [2]).

Related work. We already mentioned many related articles, hence we only briefly discuss some remaining models here. Window parity games are strongly linked to the concept of *finitary ω-regular games*: see, e.g., [20], or [15] for a complete list of references. The window mechanism can be used to ensure a certain form of (local) guarantee over runs: different techniques have been considered in MDPs, notably *variance-based* [11] or *worst-case-based* [14,7] methods.

Finally, let us mention the very recent work of Bordais et al. [8], which considers a *seemingly* related question: the authors define a *value function* based on the window *mean-payoff* mechanism and consider maximizing its expected value (which is different from the expected window size we discuss in Sect. 7). While there are definite similarities in our works w.r.t. technical tools, the two approaches are quite different and have their own strengths: we focus on deep understanding of the window mechanism through a *generic approach* for the canonical threshold probability problem for all window-based objectives, here instantiated as mean-payoff *and* parity; whereas Bordais et al. focus on a particular *optimization problem* for a function relying on this mechanism.

In addition to having *different philosophies*, our divergent approaches yield *interesting differences*. We mention three examples illustrating the conceptual gap. First, in [8], the studied function takes the same value for direct and prefix-independent bounded window mean-payoff objectives, whereas we show in Sect. 3.3 that the classical definitions of window objectives induce a striking difference between both (in the MDP of Ex. 2, the prefix-independent version is satisfied for window size one, whereas no uniform bound on all runs can be defined for the direct case). Second, we are able to prove PSPACE-hardness for the DFW mean-payoff case, whereas the best lower bound known for the related problem in [8] is PP. Lastly, let us recall that our work also deals with window *parity* objectives while the function of [8] is strictly built on mean-payoff.

Our paper presents in full details, with additional proofs and examples, the contributions published in a preceding conference version [13].

Outline. Sect. 2 defines the model and problem under study. In Sect. 3, we introduce window objectives, discuss their status in games, and illustrate their behavior in MDPs. Sect. 4 is devoted to the fixed variants and the aforementioned reductions. In Sect. 5, we analyze the case of ECs and develop the classification procedure. We build on it in Sect. 6 to solve the general case. Finally, in Sect. 7, we discuss the limitations of our work, as well as interesting extensions within arm's reach (e.g., multi-objective threshold problem, expected value problem).

2 Preliminaries

Probability distributions. Given a set S, let $\mathcal{D}(S)$ denote the set of rational probability distributions over S. Given a distribution $\iota \in \mathcal{D}(S)$, let $\mathsf{Supp}(\iota) = \{s \in S \mid \iota(s) > 0\}$ denote its support.

Markov decision processes. A finite *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, A, \delta)$ where S is a finite set of *states*, A is a finite set of *actions* and $\delta \colon S \times A \to \mathcal{D}(S)$ is a partial function called the *probabilistic transition function*. The set of actions that are available in a state $s \in S$ (i.e., for which $\delta(s, a)$ is defined) is denoted by $A(s)$. We use $\delta(s, a, s')$ as a shorthand for $\delta(s, a)(s')$. We assume w.l.o.g. that MDPs are *deadlock-free*: for all $s \in S$, $A(s) \neq \emptyset$. An MDP where for all $s \in S$, $|A(s)| = 1$ is a fully-stochastic process called a *Markov chain* (MC).

A *run* of $\mathcal{M}$ is an infinite sequence $\rho = s_0 a_0 \ldots a_{n-1} s_n \ldots$ of states and actions such that $\delta(s_i, a_i, s_{i+1}) > 0$ for all $i \geq 0$. The *prefix* up to the n-th state of ρ is the finite sequence $\rho[0, n] = s_0 a_0 \ldots a_{n-1} s_n$. The *suffix* of ρ starting from the n-th state of ρ is the run $\rho[n, \infty] = s_n a_n s_{n+1} a_{n+1} \ldots$. Moreover, we denote by $\rho[n]$ the n-th state s_n of ρ. Finite prefixes of runs of the form $h = s_0 a_0 \ldots a_{n-1} s_n$ are called *histories*. We sometimes denote the last state of history h by $\mathsf{Last}(h)$. We resp. denote the sets of runs and histories of an MDP $\mathcal{M}$ by $\mathsf{Runs}(\mathcal{M})$ and $\mathsf{Hists}(\mathcal{M})$.

End-components. Fix an MDP $\mathcal{M} = (S, A, \delta)$. A *sub-MDP* of $\mathcal{M}$ is an MDP $\mathcal{M}' = (S', A', \delta')$ with $S' \subseteq S$, $\emptyset \neq A'(s) \subseteq A(s)$ for all $s \in S'$, $\mathsf{Supp}(\delta(s, a)) \subseteq S'$ for all $s \in S', a \in A'(s)$, $\delta' = \delta|_{S' \times A'}$. Such a sub-MDP

$\mathcal{M}'$ is an *end-component* (EC) of $\mathcal{M}$ if and only if the underlying graph of $\mathcal{M}'$ is *strongly connected*, i.e., there is a run between any pair of states in S'. Given an EC $\mathcal{M}' = (S', A', \delta')$ of $\mathcal{M}$, we say that its sub-MDP $\mathcal{M}'' = (S'', A'', \delta'')$, $S'' \subseteq S'$, $A'' \subseteq A'$, is a *sub-EC* of $\mathcal{M}'$ if $\mathcal{M}''$ is also an EC. We let $\mathsf{EC}(\mathcal{M})$ denote the set of ECs of $\mathcal{M}$, which may be of exponential size as ECs need not be disjoint.

The union of two ECs with non-empty intersection is itself an EC: hence we can define the *maximal* ECs (MECs) of an MDP, i.e., the ECs that cannot be extended. We let $\mathsf{MEC}(\mathcal{M})$ denote the set of MECs of $\mathcal{M}$, of polynomial size (because MECs are pair-wise disjoints) and computable in polynomial time [19].

Strategies. A *strategy* σ is a function $\mathsf{Hists}(\mathcal{M}) \to \mathcal{D}(A)$ such that for all $h \in \mathsf{Hists}(\mathcal{M})$ ending in s, we have $\mathsf{Supp}(\sigma(h)) \subseteq A(s)$. The set of all strategies is Σ. A strategy is *pure* if all histories are mapped to *Dirac distributions*, i.e., the support is a singleton. A strategy σ can be encoded by a *Mealy machine* $(Q, \sigma_a, \sigma_u, \iota)$ where Q is a finite or infinite set of memory states, ι the *initial distribution on* Q, σ_a the *next action function* $\sigma_a \colon S \times Q \to \mathcal{D}(A)$ where $\mathsf{Supp}(\sigma_a(s, q)) \subseteq A(s)$ for any $s \in S$ and $q \in Q$, and σ_u the *memory update function* $\sigma_u \colon A \times S \times Q \to Q$. We say that σ is *finite-memory* if $|Q| < \infty$, and K-*memory* if $|Q| = K$; it is memoryless if $K = 1$, thus only depends on the last state of the history. We see such strategies as functions $s \mapsto \mathcal{D}(A(s))$ for $s \in S$. A strategy is *infinite-memory* if Q is infinite. The entity choosing the strategy is often called the *controller*.

Induced MC. An MDP $\mathcal{M}$, a strategy σ encoded by $(Q, \sigma_a, \sigma_u, \iota)$, and a state s determine a Markov chain $\mathcal{M}_s^\sigma$ defined on the state space $S \times Q$ as follows. The initial distribution is such that for any $q \in Q$, state (s, q) has probability $\iota(q)$, and 0 for other states. For any pair of states (s, q) and (s', q'), the probability of transition $(s, q) \xrightarrow{a} (s', q')$ is equal to $\sigma_a(s, q)(a) \cdot \delta(s, a, s')$ if $q' = \sigma_u(s, q, a)$, and to 0 otherwise. A *run* of $\mathcal{M}_s^\sigma$ is an infinite sequence of the form $(s_0, q_0)a_0(s_1, q_1)a_1 \ldots$, where each $(s_i, q_i) \xrightarrow{a_i} (s_{i+1}, q_{i+1})$ is a transition with non-zero probability in $\mathcal{M}_s^\sigma$, and $s_0 = s$. When considering the probabilities of events in $\mathcal{M}_s^\sigma$, we will often consider sets of runs of $\mathcal{M}$. Thus, given $E \subseteq (SA)^\omega$, we denote by $\mathbb{P}_{\mathcal{M}, s}^\sigma[E]$ the probability of the runs of $\mathcal{M}_s^\sigma$ whose projection[5] to $\mathcal{M}$ is in E, i.e., the probability of event E when $\mathcal{M}$ is executed with initial state s and strategy σ. Note that every measurable set (*event*) has a uniquely defined probability [43] (Carathéodory's extension theorem induces a unique probability measure on the Borel σ-algebra over cylinders of $(SA)^\omega$). We may drop some subscripts of $\mathbb{P}_{\mathcal{M}, s}^\sigma$ when the context is clear.

Bottom strongly-connected components. The counterparts of ECs in MCs are *bottom strongly-connected components* (BSCCs). In our formalism, where an MC is simply an MDP $\mathcal{M} = (S, A, \delta)$ with $|A(s)| = 1$ for all $s \in S$, BSCCs are exactly the ECs of such an MDP $\mathcal{M}$.

Sure and almost sure events. Let $\mathcal{M} = (S, A, \delta)$, $\sigma \in \Sigma$, and $E \subseteq (SA)^\omega$ be an event. We say that E is *sure*, written $\mathsf{S}_{\mathcal{M}, s}^\sigma[E]$, if and only if $\mathsf{Runs}(\mathcal{M}_s^\sigma) \subseteq E$ (again abusing our notation to consider projections on $(SA)^\omega$); and that E is *almost-sure*, written $\mathsf{AS}_{\mathcal{M}, s}^\sigma[E]$, if and only if $\mathbb{P}_{\mathcal{M}, s}^\sigma[E] = 1$.

Almost-sure reachability of ECs. Given a run $\rho = s_0 a_0 s_1 a_1 \ldots \in \mathsf{Runs}(\mathcal{M})$, let

$$\mathsf{inf}(\rho) = \{s \in S \mid \forall i \geq 0, \exists j > i, s_j = s\}$$

denote the set of states visited infinitely-often along ρ, and let

$$\mathsf{infAct}(\rho) = \{a \in A \mid \forall i \geq 0, \exists j > i, a_j = a\}$$

similarly denote the actions taken infinitely-often along ρ.

Let $\mathsf{limitSet}(\rho)$ denote the pair $(\mathsf{inf}(\rho), \mathsf{infAct}(\rho))$. Note that this pair may induce a well-defined sub-MDP $\mathcal{M}' = (\mathsf{inf}(\rho), \mathsf{infAct}(\rho), \delta|_{\mathsf{inf}(\rho) \times \mathsf{infAct}(\rho)})$, but in general it needs not be the case. A folklore result in MDPs (e.g., [4]) is the following: for any state s of MDP $\mathcal{M}$, for any strategy $\sigma \in \Sigma$, we have that

$$\mathsf{AS}_{\mathcal{M}, s}^\sigma[\{\rho \in \mathsf{Runs}(\mathcal{M}_s^\sigma) \mid \mathsf{limitSet}(\rho) \in \mathsf{EC}(\mathcal{M})\}],$$

---

[5] The projection of a run $(s_0, q_0)a_0(s_1, q_1)a_1 \ldots$ in $\mathcal{M}_s^\sigma$ to $\mathcal{M}$ is simply the run $s_0 a_0 s_1 a_1 \ldots$ in $\mathcal{M}$. For the sake of readability, we make similar abuse of notation – identifying runs in the induced MC with their projections in the MDP – throughout our paper.

that is, under any strategy, the limit behavior of the MDP almost-surely coincides with an EC. This property is a key tool in the analysis of MDPs with prefix-independent objectives, as it essentially says that we only need to identify the "best" ECs and maximize the probability to reach them.

Decision problem. An *objective* for an MDP $\mathcal{M} = (S, A, \delta)$ is a measurable set of runs $E \subseteq (SA)^\omega$. Given an MDP $\mathcal{M} = (S, A, \delta)$, an initial state s, a threshold $\alpha \in [0, 1] \cap \mathbb{Q}$, and such an objective E, the *threshold probability problem* is to decide whether there exists a strategy $\sigma \in \Sigma$ such that $\mathbb{P}^\sigma_{\mathcal{M}, s}[E] \geq \alpha$ or not.

Furthermore, if it exists, we want to build such a strategy.

Weights and priorities. In this paper, we always assume an MDP $\mathcal{M} = (S, A, \delta)$ with either (i) a *weight* function $w \colon A \to \mathbb{Z}$ of largest absolute weight W, or (ii) a *priority* function $p \colon S \to \{0, 1, \ldots, d\}$, with $d \leq |S| + 1$ (w.l.o.g.). This choice is left implicit when the context is clear, to offer a unified view of mean-payoff and parity variants of window objectives.

Complexity. When studying the complexity of decision problems, we make the classical assumptions of the field: we consider the model size $|\mathcal{M}|$ to be polynomial in $|S|$ and the *binary encoding* of weights and probabilities (e.g., $V = \log_2 W$, with W the largest absolute weight), whereas we consider the largest priority d, as well as the upcoming window size λ, to be encoded in unary. When a problem is polynomial in W, we say that it is *pseudo*-polynomial: it would be polynomial if weights would be given in unary.

Mean-payoff and parity objectives. We consider window objectives based on *mean-payoff* and *parity* objectives. Let us discuss those classical objectives.

The first one is a *quantitative* objective, for which we consider *weighted* MDPs. Let $\rho \in \mathsf{Runs}(\mathcal{M})$ be a run of such an MDP. The *mean-payoff* of prefix $\rho[0, n]$ is $\mathsf{MP}(\rho[0, n]) = \frac{1}{n} \sum_{i=0}^{n-1} w(a_i)$, for $n > 0$. This is naturally extended to runs by considering the limit behavior. The *mean-payoff* of ρ is $\mathsf{MP}(\rho) = \liminf_{n \to \infty} \mathsf{MP}(\rho[0, n])$. Given a threshold $\nu \in \mathbb{Q}$, the mean-payoff objective accepts all runs whose mean-payoff is above the threshold, i.e., $\mathsf{MeanPayoff}(\nu) = \{\rho \in \mathsf{Runs}(\mathcal{M}) \mid \mathsf{MP}(\rho) \geq \nu\}$. The corresponding threshold probability problem is in P using linear programming, and pure memoryless strategies suffice (see, e.g., [39]). Note that ν can be taken equal to zero w.l.o.g., and the mean-payoff function can be equivalently (in the classical one-dimension setting) defined using $\limsup$.

The second objective, parity, is a *qualitative* one, for which we consider MDPs with a priority function. The *parity objective* requires that the smallest priority seen infinitely often along a run be even, i.e., $\mathsf{Parity} = \{\rho \in \mathsf{Runs}(\mathcal{M}) \mid \min_{s \in \mathsf{inf}(\rho)} p(s) = 0 \pmod 2\}$. Again, the corresponding threshold probability problem is in P and pure memoryless strategies suffice [21].

3 Window objectives

3.1 Definitions

Good windows. Given a weighted MDP $\mathcal{M}$ and $\lambda > 0$, we define the *good window mean-payoff* objective

$$\mathsf{GW_{mp}}(\lambda) = \left\{\rho \in \mathsf{Runs}(\mathcal{M}) \mid \exists\, l < \lambda,\ \mathsf{MP}(\rho[0, l+1]) \geq 0\right\}$$

requiring the existence of a window of size bounded by λ and starting at the first position of the run, over which the mean-payoff is at least equal to zero (w.l.o.g.).

Similarly, given an MDP $\mathcal{M}$ with priority function p, we define the *good window parity* objective,

$$\mathsf{GW_{par}}(\lambda) = \left\{\rho \in \mathsf{Runs}(\mathcal{M}) \mid \exists\, l < \lambda,\ \big(p(\rho[l]) \bmod 2 = 0 \wedge \forall\, k < l,\ p(\rho[l]) < p(\rho[k])\big)\right\}$$

requiring the existence of a window of size bounded by λ and starting at the first position of the run, for which the last priority is even and is the smallest within the window.

To preserve our generic approach, we use subscripts mp and par for mean-payoff and parity variants respectively. So, given $\Omega = \{\mathsf{mp}, \mathsf{par}\}$ and a run $\rho \in \mathsf{Runs}(\mathcal{M})$, we say that *an Ω-window is closed* in at most λ steps from $\rho[i]$ if $\rho[i, \infty]$ is in $\mathsf{GW_\Omega}(\lambda)$. If a window is not yet closed, we call it *open*.

Fixed variants. Given $\lambda > 0$, we define the *direct fixed window* objective

$$\mathsf{DFW}_\Omega(\lambda) = \{\rho \in \mathsf{Runs}(\mathcal{M}) \mid \forall\, j \geq 0,\ \rho[j, \infty] \in \mathsf{GW}_\Omega(\lambda)\}$$

asking for all Ω-windows to be closed within λ steps along the run.

We also define the *fixed window* objective

$$\mathsf{FW}_\Omega(\lambda) = \{\rho \in \mathsf{Runs}(\mathcal{M}) \mid \exists\, i \geq 0,\ \rho[i, \infty] \in \mathsf{DFW}_\Omega(\lambda)\}$$

that is the *prefix-independent* version of the previous one: it requires it to be eventually satisfied.

Bounded variant. Finally, we define the *bounded window* objective

$$\mathsf{BW}_\Omega = \{\rho \in \mathsf{Runs}(\mathcal{M}) \mid \exists\, \lambda > 0,\ \rho \in \mathsf{FW}_\Omega(\lambda)\}$$

requiring the existence of a bound λ for which the fixed window objective is satisfied. Note that this bound need not be *uniform* along all runs in general. A *direct* variant may also be defined, but turns out to be ill-suited in the stochastic context: we illustrate it in Sect. 3.3 and discuss its pitfalls in Sect. 7. Hence we focus on the prefix-independent version in the following.

3.2 Overview in games.

Window mean-payoff and window parity objectives were considered in two-player zero-sum games [18,15]. The game setting is equivalent to deciding if there exists a controller strategy in an MDP such that the corresponding objective is *surely* satisfied, i.e., by all consistent runs. We quickly summarize the main results here. In both cases, window objectives establish *conservative approximations* of the classical ones, with *improved complexity*.

For mean-payoff [18], (direct and prefix-independent) fixed window objectives can be solved in polynomial time, both in the model and the window size. Memory is in general needed for both players, but polynomial memory suffices. Bounded versions belong to $\mathsf{NP} \cap \mathsf{coNP}$ and are as hard as mean-payoff games. Memoryless strategies suffice for the controller but not for its opponent (infinite memory is needed for the prefix-independent case, polynomial memory suffices in the direct case). Interestingly, if the controller can win the bounded window objective, then a *uniform bound* exists, i.e., there exists a window size λ sufficiently large such that the bounded version coincides with the fixed one. Recall that this is not granted by definition. We will see that *in MDPs, this does not hold* (Ex. 2). This uniform bound in games is however pseudo-polynomial.

For parity [15], similar results are obtained. The crucial difference is the uniform bound on λ, which also exists but in this case is equal to the number of states of the game. Thanks to that, all variants of window parity objectives belong to P. The memory requirements are the same as for window mean-payoff.

3.3 Illustration

Example 1. We first go back to the example of Sect. 1, depicted in Fig. 1. Let $\mathcal{M}$ be this MDP. Fix run $\rho = (s_1\, a\, s_2\, b\, s_3\, c)^\omega$. We have that $\rho \notin \mathsf{FW}_{\mathsf{par}}(\lambda = 2)$ – a fortiori, $\rho \notin \mathsf{DFW}_{\mathsf{par}}(\lambda = 2)$ – as the window that opens in s_1 is not closed after two steps (because s_1 has odd priority 1, and 2 is not smaller than 1 so does not suffice to answer it). If we now set $\lambda = 3$, we see that this window closes on time, as 0 is encountered within three steps. As all other windows are immediately closed, we have $\rho \in \mathsf{DFW}_{\mathsf{par}}(\lambda = 3)$ – a fortiori, $\rho \in \mathsf{FW}_{\mathsf{par}}(\lambda = 3)$ and $\rho \in \mathsf{BW}_{\mathsf{par}}$.

Regarding the probability of these objectives, however, we have already argued that, for all $\lambda > 0$, $\mathbb{P}_{\mathcal{M}, s_1}[\mathsf{FW}_{\mathsf{par}}(\lambda)] = 0$, whereas $\mathbb{P}_{\mathcal{M}, s_1}[\mathsf{Parity}] = 1$ since s_3 is almost-surely visited infinitely often but any time bound is almost-surely exceeded infinitely often too. Observe that $\mathsf{BW}_{\mathsf{par}} = \bigcup_{\lambda > 0} \mathsf{FW}_{\mathsf{par}}(\lambda)$, hence we also have that $\mathbb{P}_{\mathcal{M}, s_1}[\mathsf{BW}_{\mathsf{par}}] = 0$ (by countable additivity).

Similar reasoning holds for window mean-payoff objectives, by taking the weight function $w = \{a \mapsto -1, b \mapsto 0, c \mapsto 1\}$. ◁

With the next example, we illustrate one of the main differences between games and MDPs w.r.t. window objectives. As discussed in Sect. 3.2, in two-player zero-sum games, both for parity and mean-payoff, there exists a *uniform bound* on the window size λ such that the fixed variants coincide with the bounded ones. We show that this property does not carry over to MDPs.
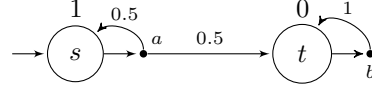

Fig. 2. On this MC, there is no uniform bound over all runs, in contrast to the game setting.

Example 2. Consider the MC $\mathcal{M}$ depicted in Fig. 2. It is clear that for any $\lambda > 0$, there is probability $1/2^{\lambda-1}$ that objective $\mathsf{DFW}_{\mathsf{par}}(\lambda)$ is not satisfied. Hence, for all $\lambda > 0$, $\mathbb{P}_{\mathcal{M},s}[\mathsf{DFW}_{\mathsf{par}}(\lambda)] < 1$.

Now let $\mathsf{DBW}_{\mathsf{par}}$ be the *direct* bounded window objective evoked in Sect. 3.1 and defined as $\mathsf{DBW}_{\mathsf{par}} = \bigcup_{\lambda>0} \mathsf{DFW}_{\mathsf{par}}(\lambda)$. We claim that $\mathbb{P}_{\mathcal{M},s}[\mathsf{DBW}_{\mathsf{par}}] = 1$. Indeed, any run ending in t belongs to $\mathsf{DBW}_{\mathsf{par}}$, as it belongs to $\mathsf{DFW}_{\mathsf{par}}(\lambda)$ for λ equal to the length of the prefix up to t. Since t is almost-surely reached (as it is the only BSCC of the MC), we conclude that $\mathsf{DBW}_{\mathsf{par}}$ is indeed satisfied almost-surely.

Essentially, the difference stems from the fairness of probabilities. In a game, the opponent would control the successor choice after action a and always go back to s, resulting in objective $\mathsf{DBW}_{\mathsf{par}}$ being lost. However, in this MC, s is almost-surely left eventually, but we cannot guarantee when: hence there exists a window bound for each run, but there is no uniform bound over all runs.

This simple MC also illustrates the classical difference between sure and almost-sure satisfaction: while we have $\mathsf{AS}_{\mathcal{M},s}[\mathsf{FW}_{\mathsf{par}}(\lambda = 1)]$, we do not have $\mathsf{S}_{\mathcal{M},s}[\mathsf{FW}_{\mathsf{par}}(\lambda = 1)]$, because of run $\rho = (s\,a)^{\omega}$.

Again the same reasoning holds for mean-payoff variants, for example with $w = \{a \mapsto -1, b \mapsto 1\}$.   ◁

We leave out the *direct* bounded objective DBW_{Ω} from now on. We will come back to it in Sect. 7 and motivate why this objective is not well-behaved. Hence, in the following, we focus on direct and prefix-independent fixed window objectives and prefix-independent bounded ones.

4 Fixed case: better safe than sorry

We start with the fixed variants of window objectives. Our main goal here is to establish that *pure finite-memory* strategies suffice in all cases. As a by-product, we also obtain algorithms to solve the corresponding decision problems. Still, for the prefix-independent variants, we will obtain better complexities using the upcoming generic approach (Sect. 6).

Our main tools are natural reductions from direct (resp. prefix-independent) window problems on MDPs to safety (resp. co-Büchi) problems on *unfoldings* based on the window size λ (i.e., larger arenas incorporating information on open windows). We use *identical unfoldings* for both direct and prefix-independent objectives, in order to obtain a unified proof. In both cases, the set of states to avoid in the unfolding corresponds to leaving a window open for λ steps.

4.1 Reductions

Mean-payoff. Let $\mathcal{M} = (S, A, \delta)$ be an MDP with weight function w (of maximal absolute weight W), and $\lambda > 0$ be the window size. We define the unfolding MDP $\mathcal{M}_{\lambda} = (\tilde{S}, A, \tilde{\delta})$ as follows:

- $\tilde{S} = S \times \{0, \ldots, \lambda\} \times \{-\lambda \cdot W, \ldots, 0\}$.
- $\tilde{\delta} \colon \tilde{S} \times A \to \mathcal{D}(\tilde{S})$ is defined as follows for all a in A:

$$\tilde{\delta}\left((s, l, z), a\right)(t, l+1, z+w(a)) = \nu \text{ if } \left(\delta(s,a)(t) = \nu\right) \,\wedge\, \left(l < \lambda\right) \,\wedge\, \left(z + w(a) < 0\right),$$

$$\tilde{\delta}\left((s, l, z), a\right)(t, 0, 0) = \nu \text{ if } \left(\delta(s,a)(t) = \nu\right) \,\wedge\, \left[\left(z + w(a) \geq 0\right) \vee \left(\left(l = \lambda\right) \,\wedge\, \left(z < 0\right)\right)\right].$$

- Once an initial state $s_{\mathrm{init}} \in S$ is fixed in $\mathcal{M}$, the associated initial state in $\mathcal{M}_{\lambda}$ is $\tilde{s}_{\mathrm{init}} = (s_{\mathrm{init}}, 0, 0) \in \tilde{S}$.

Note that $\mathcal{M}_\lambda$ is unweighted: it keeps track in each of its states of the current state of $\mathcal{M}$, the size of the current open window as well as the current sum of weights in the window: these two values are reset whenever a window is closed (left-hand side of the disjunction) or stays open for λ steps (right hand-side). A key underlying property used here is the so-called *inductive property of windows* [18,15].

Property 3 (Inductive property of windows). Consider a run $\rho = s_0 a_0 s_1 a_1 \ldots$ in an MDP. Fix a window starting in position $i \geq 0$. Let j be the position in which this window gets closed, assuming it does. Then, all windows in positions from i to j also close in j.

The validity of this property is easy to check by contradiction (if it was not the case, then the window in i would close before j). This property is fundamental in our reduction: without it we would have to keep track of all open windows in parallel, which would result in a blow-up exponential in λ.

In the upcoming reductions, the set of states to avoid will be $B = \{(s, l, z) \mid (l = \lambda) \wedge (z < 0)\}$. By construction of $\mathcal{M}_\lambda$, it exactly corresponds to windows staying open for λ steps.

Parity. Let $\mathcal{M} = (S, A, \delta)$ be an MDP with priority function p, and $\lambda > 0$ be the window size. We define the unfolding MDP $\mathcal{M}_\lambda = (\tilde{S}, A, \tilde{\delta})$ as follows:

- $\tilde{S} = S \times \{0, \ldots, \lambda\} \times \{0, 1, \ldots, d\}$.
- $\tilde{\delta} \colon \tilde{S} \times A \to \mathcal{D}(\tilde{S})$ is defined as follows for all a in A:

$$\tilde{\delta}\left((s, l, c), a\right)(t, l+1, \min(c, p(t))) = \nu \text{ if } \left(\delta(s, a)(t) = \nu\right) \wedge \left(l < \lambda - 1\right) \wedge \left(c \bmod 2 = 1\right),$$
$$\tilde{\delta}\left((s, l, c), a\right)(t, 0, p(t)) = \nu \text{ if } \left(\delta(s, a)(t) = \nu\right) \wedge \left(l = \lambda - 1\right) \wedge \left(c \bmod 2 = 1\right),$$
$$\tilde{\delta}\left((s, l, c), a\right)(t, 0, p(t)) = \nu \text{ if } \left(\delta(s, a)(t) = \nu\right) \wedge \left(c \bmod 2 = 0\right).$$

- Once an initial state $s_{\text{init}} \in S$ is fixed in $\mathcal{M}$, the associated one in $\mathcal{M}_\lambda$ is $\tilde{s}_{\text{init}} = (s_{\text{init}}, 0, p(s_{\text{init}})) \in \tilde{S}$.

This unfolding keeps track in each of its states of the current state of the original MDP, the size of the current window and the minimum priority in the window: again, these two values are reset whenever a window is closed or stays open for λ steps.

In this case, the set of states to avoid will be $B = \{(s, l, c) \mid (l = \lambda - 1) \wedge (c \bmod 2 = 1)\}$, with an equivalent interpretation.

Remark 4. There is a slight asymmetry in the use of indices for the current window size in the two constructions: that is due to mean-payoff using weights on actions (two states being needed for one action) and parity using priorities on states.

Objectives. We treat mean-payoff and parity versions in a unified way from now on. As stated before, the set B represents in both cases windows being open for λ steps and should be avoided: at all times for direct variants, eventually for prefix-independent ones. We define the following objectives over the unfoldings:

$$\mathsf{Reach}(\mathcal{M}_\lambda) = (\tilde{S}A)^* B A (\tilde{S}A)^\omega, \qquad \mathsf{Safety}(\mathcal{M}_\lambda) = (\tilde{S}A)^\omega \setminus \mathsf{Reach}(\mathcal{M}_\lambda),$$
$$\mathsf{Buchi}(\mathcal{M}_\lambda) = ((\tilde{S}A)^* B A)^\omega, \qquad \mathsf{coBuchi}(\mathcal{M}_\lambda) = (\tilde{S}A)^\omega \setminus \mathsf{Buchi}(\mathcal{M}_\lambda).$$

Our ultimate goal is to prove that the safety and co-Büchi objectives in $\mathcal{M}_\lambda$ are *probability-wise equivalent* to the direct fixed window and fixed window ones in $\mathcal{M}$, modulo a well-defined *mapping* between histories, runs and strategies. This will induce the correctness of our reduction.

Mapping. Fix an initial state s_{init} in $\mathcal{M}$ and let $\tilde{s}_{\text{init}}$ be its corresponding initial state in $\mathcal{M}_\lambda$. Let $\mathsf{Hists}(\mathcal{M}, s_{\text{init}})$ denote the histories of $\mathcal{M}$ starting in s_{init}. We start by defining a bijective mapping $\pi_{\mathcal{M}_\lambda}$, and its inverse $\pi_{\mathcal{M}}$, between *histories* of $\mathsf{Hists}(\mathcal{M}, s_{\text{init}})$ and $\mathsf{Hists}(\mathcal{M}_\lambda, \tilde{s}_{\text{init}})$. We use $\pi_{\mathcal{M}_\lambda} \colon \mathsf{Hists}(\mathcal{M}, s_{\text{init}}) \to \mathsf{Hists}(\mathcal{M}_\lambda, \tilde{s}_{\text{init}})$ for the $\mathcal{M}$-to-$\mathcal{M}_\lambda$ direction, and $\pi_{\mathcal{M}} \colon \mathsf{Hists}(\mathcal{M}_\lambda, \tilde{s}_{\text{init}}) \to \mathsf{Hists}(\mathcal{M}, s_{\text{init}})$ for the opposite one. We define $\pi_{\mathcal{M}_\lambda}$ inductively as follows.

- $\pi_{\mathcal{M}_\lambda}(s_{\text{init}}) = \tilde{s}_{\text{init}}$.

– Let $h \in \mathsf{Hists}(\mathcal{M}, s_{\mathrm{init}})$, $\tilde{h} = \pi_{\mathcal{M}_\lambda}(h)$, $a \in A$, $s \in S$. Then, $\pi_{\mathcal{M}_\lambda}(h \cdot a \cdot s) = \tilde{h} \cdot a \cdot \tilde{s}$, where $\tilde{s}$ is obtained from $\mathsf{Last}(\tilde{h})$, a and s following the unfolding construction.

We define $\pi_{\mathcal{M}}$ as its inverse, i.e., the function projecting histories of $\mathsf{Hists}(\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}})$ to $(SA)^*S$. We naturally extend these mappings to *runs* based on this inductive construction.

Now, we also extend these mappings to *strategies*. Let σ be a strategy in $\mathcal{M}$. We define its twin strategy $\tilde{\sigma} = \pi_{\mathcal{M}_\lambda}(\sigma)$ in $\mathcal{M}_\lambda$ as follows: for all $\tilde{h} \in \mathsf{Hists}(\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}})$, $\tilde{\sigma}(\tilde{h}) = \sigma(\pi_{\mathcal{M}}(\tilde{h}))$. Note that this strategy is well-defined as $\mathcal{M}$ and $\mathcal{M}_\lambda$ share the same actions and $\pi_{\mathcal{M}}$ is a proper function over $\mathsf{Hists}(\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}})$ (i.e., each history $\tilde{h}$ has an image in $\mathsf{Hists}(\mathcal{M}, s_{\mathrm{init}})$). Similarly, given a strategy $\tilde{\sigma}$ in $\mathcal{M}_\lambda$, we build a twin strategy $\sigma = \pi_{\mathcal{M}}(\tilde{\sigma})$ using $\pi_{\mathcal{M}_\lambda}$. Hence we also have a bijection over strategies.

We say that two objects (histories, runs, strategies) are π-*corresponding* if they are the image of one another through mappings $\pi_{\mathcal{M}}$ and $\pi_{\mathcal{M}_\lambda}$.

Probability-wise equivalence. For any any history h, we denote by $\mathsf{Cyl}(h)$ the *cylinder set* spanned by it, i.e., the set of all runs prolonging h. Cylinder sets are the building blocks of probability measures in MCs, as all measurable sets belong to the σ-algebra built upon them [4]. We show that our mappings preserve probabilities of cylinder sets.

Lemma 5. *Let $\mathcal{M}$, $\mathcal{M}_\lambda$, $\pi_{\mathcal{M}}$ and $\pi_{\mathcal{M}_\lambda}$ be defined as above. Fix any couple of π-corresponding strategies $(\sigma, \tilde{\sigma})$ for $\mathcal{M}$ and $\mathcal{M}_\lambda$ respectively. Then, for any couple of π-corresponding histories $(h, \tilde{h})$ in $\mathsf{Hists}(\mathcal{M}, s_{init}) \times \mathsf{Hists}(\mathcal{M}_\lambda, \tilde{s}_{init})$, we have that*

$$\mathbb{P}^{\sigma}_{\mathcal{M}, s_{init}}[\mathsf{Cyl}(h)] = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}_{init}}[\mathsf{Cyl}(\tilde{h})]. \tag{1}$$

Proof. We fix $\mathcal{M}$, $\mathcal{M}_\lambda$, $\pi_{\mathcal{M}}$, $\pi_{\mathcal{M}_\lambda}$, and a couple of π-corresponding strategies $(\sigma, \tilde{\sigma})$. We prove the equality by induction on histories. The base case, for $h = s_{\mathrm{init}}$ and $\tilde{h} = \tilde{s}_{\mathrm{init}}$, is trivial, as we have

$$\mathbb{P}^{\sigma}_{\mathcal{M}, s_{\mathrm{init}}}[\mathsf{Cyl}(h)] = \mathbb{P}^{\sigma}_{\mathcal{M}, s_{\mathrm{init}}}[\mathsf{Runs}(\mathcal{M}, s_{\mathrm{init}})] = 1 = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}}}[\mathsf{Runs}(\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}})] = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}}}[\mathsf{Cyl}(\tilde{h})].$$

Now, assume Eq. (1) true for a couple of π-corresponding histories $(h, \tilde{h})$. Consider any one-step extension of these histories, say $(h \cdot a \cdot s, \tilde{h} \cdot a \cdot \tilde{s})$ for $a \in A$, $s \in S$ and $\tilde{h} \cdot a \cdot \tilde{s} = \pi_{\mathcal{M}_\lambda}(h \cdot a \cdot s)$. We need to prove that Eq. (1) still holds for this pair of histories. Let us expand the equality to prove as follows:

$$\mathbb{P}^{\sigma}_{\mathcal{M}, s_{\mathrm{init}}}[\mathsf{Cyl}(h \cdot a \cdot s)] = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}}}[\mathsf{Cyl}(\tilde{h} \cdot a \cdot \tilde{s})]$$
$$\iff \mathbb{P}^{\sigma}_{\mathcal{M}, s_{\mathrm{init}}}[h] \cdot \sigma(h)(a) \cdot \delta(\mathsf{Last}(h), a, s) = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}}}[\tilde{h}] \cdot \tilde{\sigma}(\tilde{h})(a) \cdot \tilde{\delta}(\mathsf{Last}(\tilde{h}), a, \tilde{s}).$$

Now, observe that $\sigma(h)(a) = \tilde{\sigma}(\tilde{h})(a)$ since σ and $\tilde{\sigma}$ are π-corresponding strategies and h and $\tilde{h}$ are π-corresponding histories. Furthermore, $\delta(\mathsf{Last}(h), a, s) = \tilde{\delta}(\mathsf{Last}(\tilde{h}), a, \tilde{s})$ by construction of the unfolding. Now since $\mathbb{P}^{\sigma}_{\mathcal{M}, s_{\mathrm{init}}}[h] = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}_{\mathrm{init}}}[\tilde{h}]$ holds by induction hypothesis, we are done. $\square$

Since the previous lemma holds for all cylinders, we may extend the result to any event (see, e.g., [4] for the construction of the σ-algebra and related questions).

Corollary 6. *Let $\mathcal{M}$, $\mathcal{M}_\lambda$, $\pi_{\mathcal{M}}$ and $\pi_{\mathcal{M}_\lambda}$ be defined as above. Fix any couple of π-corresponding strategies $(\sigma, \tilde{\sigma})$ for $\mathcal{M}$ and $\mathcal{M}_\lambda$ respectively. Then, for any couple of π-corresponding measurable objectives $(E, \tilde{E}) \subseteq \mathsf{Runs}(\mathcal{M}, s_{init}) \times \mathsf{Runs}(\mathcal{M}_\lambda, \tilde{s}_{init})$, we have that*

$$\mathbb{P}^{\sigma}_{\mathcal{M}, s_{init}}[E] = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}_{init}}[\tilde{E}].$$

Correctness of the reductions. We may now establish our reductions.

Lemma 7. *Let $\mathcal{M} = (S, A, \delta)$ be an MDP, $\lambda > 0$ be the window size, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\mathcal{M}_\lambda = (\tilde{S}, A, \tilde{\delta})$ be the unfolding of $\mathcal{M}$ defined as above for variant Ω, $s \in S$ be a state of $\mathcal{M}$, and $\tilde{s} \in \tilde{S}$ be its π-corresponding state in $\mathcal{M}_\lambda$. The following assertions hold.*

1. *For any strategy σ in $\mathcal{M}$, there exists a strategy $\tilde{\sigma}$ in $\mathcal{M}_\lambda$ such that*

$$\mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}}[\mathsf{Safety}(\mathcal{M}_\lambda)] = \mathbb{P}^{\sigma}_{\mathcal{M}, s}[\mathsf{DFW}_\Omega(\lambda)] \qquad \wedge \qquad \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}}[\mathsf{coBuchi}(\mathcal{M}_\lambda)] = \mathbb{P}^{\sigma}_{\mathcal{M}, s}[\mathsf{FW}_\Omega(\lambda)].$$

2. *For any strategy $\tilde{\sigma}$ in $\mathcal{M}_\lambda$, there exists a strategy σ in $\mathcal{M}$ such that*

$$\mathbb{P}^{\sigma}_{\mathcal{M}, s}[\mathsf{DFW}_\Omega(\lambda)] = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}}[\mathsf{Safety}(\mathcal{M}_\lambda)] \qquad \wedge \qquad \mathbb{P}^{\sigma}_{\mathcal{M}, s}[\mathsf{FW}_\Omega(\lambda)] = \mathbb{P}^{\tilde{\sigma}}_{\mathcal{M}_\lambda, \tilde{s}}[\mathsf{coBuchi}(\mathcal{M}_\lambda)].$$

Moreover, such strategies can be obtained through mappings $\pi_{\mathcal{M}}$ and $\pi_{\mathcal{M}_\lambda}$.

Proof. To prove it, it suffices to see that $\mathsf{Safety}(\mathcal{M}_\lambda)$ (resp. $\mathsf{coBuchi}(\mathcal{M}_\lambda)$) is π-corresponding with $\mathsf{DFW}_\Omega(\lambda)$ (resp. $\mathsf{FW}_\Omega(\lambda)$) and invoke Cor. 6. As noted earlier, this correspondence is trivial by construction of the unfoldings. Consider the safety case: a run $\tilde{\rho}$ in $\mathcal{M}_\lambda$ belongs to $\mathsf{Safety}(\mathcal{M}_\lambda)$ *if and only if* all the windows along $\rho = \pi_{\mathcal{M}}(\tilde{\rho})$ close within λ steps. Similarly, a run $\tilde{\rho}$ in $\mathcal{M}_\lambda$ belongs to $\mathsf{coBuchi}(\mathcal{M}_\lambda)$ *if and only if* it visits B finitely often, hence if and only if $\rho = \pi_{\mathcal{M}}(\tilde{\rho})$ contains a finite number of windows left open for λ steps. □

Intuitively, to obtain a strategy σ in $\mathcal{M}$ from a strategy $\tilde{\sigma}$ in the unfolding $\mathcal{M}_\lambda$, we have to integrate in the memory of σ the additional information encoded in $\tilde{S}$: hence the memory required by σ is the one used by $\tilde{\sigma}$ with a blow-up polynomial in $|\tilde{S}|$.

4.2 Memory requirements and complexity

Upper bounds. Thanks to the reductions established in Lem. 7, along with the fact that *pure memoryless* strategies suffice for safety and co-Büchi objectives in MDPs [4], we obtain the following result.

Theorem 8. *Pure finite-memory strategies suffice for the threshold probability problem for all fixed window objectives. That is, given MDP $\mathcal{M} = (S, A, \delta)$, initial state $s \in S$, window size $\lambda > 0$, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, objective $E \in \{\mathsf{DFW}_\Omega(\lambda), \mathsf{FW}_\Omega(\lambda)\}$ and threshold probability $\alpha \in [0, 1] \cap \mathbb{Q}$, if there exists a strategy $\sigma \in \Sigma$ such that $\mathbb{P}^{\sigma}_{\mathcal{M}, s}[E] \geq \alpha$, then there exists a pure finite-memory strategy σ' such that $\mathbb{P}^{\sigma'}_{\mathcal{M}, s}[E] \geq \alpha$.*

These reductions also yield algorithms for the threshold probability problem in the fixed window case. We only use them for the direct variants, as the generic approach we develop in Sect. 6 proves to be more efficient for the prefix-independent one, for two reasons: first, we may restrict the co-Büchi-like analysis to end-components; second, we use a more tractable analysis than the co-Büchi unfolding for mean-payoff. However, the reduction established for prefix-independent variants is not without interest: it yields sufficiency of finite-memory strategies, which is a *key ingredient* to establish our generic approach (used in Lem. 18).

Theorem 9. *The threshold probability problem is*

(a) *in* P *for direct fixed window parity objectives, and pure polynomial-memory optimal strategies can be constructed in polynomial time.*

(b) *in* EXPTIME *for direct fixed window mean-payoff objectives, and pure pseudo-polynomial-memory optimal strategies can be constructed in pseudo-polynomial time.*

Proof. The algorithm is simple: given $\mathcal{M} = (S, A, \delta)$ and $\lambda > 0$, build $\mathcal{M}_\lambda$ and solve the corresponding safety problem. This can be done in polynomial time in $|\mathcal{M}_\lambda|$ and pure memoryless strategies suffice over $\mathcal{M}_\lambda$ [4].

For parity, the unfolding is of size polynomial in $|\mathcal{M}|$, the number of priorities d and the window size λ. Since both d (anyway bounded by $\mathcal{O}(|S|)$) and λ are assumed to be given in unary, it yields the result.

For mean-payoff, the unfolding is of size polynomial in $|\mathcal{M}|$, the largest absolute weight W and the window size λ. Since weights are assumed to be encoded in binary, we only have a pseudo-polynomial-time algorithm. □

Lower bounds. We complement the results of Thm. 9 with almost-matching lower bounds, showing that our approach is close to optimal, complexity-wise.

Theorem 10. *The threshold probability problem is*

(a) P-*hard for direct fixed window parity objectives, and polynomial-memory strategies are in general necessary;*

(b) PSPACE-*hard for direct fixed window mean-payoff objectives (even for acyclic MDPs), and pseudopolynomial-memory strategies are in general necessary.*

Proof. Item (a). We establish P-hardness for direct fixed window parity objectives through a reduction from two-player reachability games, which are known to be P-complete [6,35]. Let $\mathcal{G} = (V = V_1 \uplus V_2, E)$ be a game graph, where states in V_1 (resp. V_2) belong to player 1 (resp. player 2), and $E \subseteq V \times V$ is the set of transitions. Without loss of generality, we assume this graph to be strictly alternating, i.e., $E \subseteq V_1 \times V_2 \uplus V_2 \times V_1$, and without deadlock. The reachability objective $\mathsf{Reach}(T)$ for $T \subseteq V$ accepts all plays that eventually visit the set T. Again w.l.o.g., we assume that $T \subseteq V_1$. Given an initial state $v_{\mathrm{init}} \in V_1$ and target set $T \subseteq V_1$, deciding if player 1 has a winning strategy ensuring that T is visited whatever the strategy of player 2 is P-hard. We reduce this question to a threshold probability problem for a direct fixed window parity objective as follows. From $\mathcal{G}$, we build the MDP $\mathcal{M} = (S, A, \delta)$ such that $S = V_1$, $A = V_2$ and δ is constructed in the following manner:

– for all $v_1 \in V_1 \setminus T$, $v_1' \in V_1$, $v_2 \in V_2$, $(v_1, v_2, v_1') \in \mathsf{Supp}(\delta)$ iff $(v_1, v_2) \in E$ and $(v_2, v_1') \in E$;

– probabilities are taken uniform, i.e., $\delta(v_1, v_2, v_1') = \left(\dfrac{1}{|\mathsf{Supp}(\delta(v_1, v_2))|} \right)$;

– states in T are made absorbing, i.e., they only allow an action a such that $\delta(v, a, v) = 1$ for any $v \in T$.

We add a priority function $p \colon S \to \{0, 1\}$ that assigns 1 to all states in S except states that correspond to states in $T \subseteq V_1$: those states get priority 0. Let us fix objective $\mathsf{DFW}_{\mathsf{par}}(\lambda = |V_1|)$. We claim that player 1 has a winning strategy from v_{init} in the reachability game $\mathcal{G}$ if and only if the controller has a strategy almost-surely satisfying objective $\mathsf{DFW}_{\mathsf{par}}(\lambda = |V_1|)$ from v_{init} in the MDP $\mathcal{M}$. Observe that this reduction is polynomial, as both $\mathcal{M}$ and λ are polynomial in the size of $\mathcal{G}$ and the number of priorities is fixed. It remains to check its correctness.

Consider what happens in $\mathcal{M}$. The only way to close the window that will open in v_{init} is to reach T, and we ought to close it to obtain a satisfying run as we consider the *direct* objective. Furthermore, once T is reached, the run is necessarily winning as we stay in T forever and always see priority 0. We also know that if T can be reached, λ steps suffice to do so, as memoryless strategies suffice in the game $\mathcal{G}$ (note that we do not count actions here). Lifting a winning strategy from $\mathcal{G}$ to $\mathcal{M}$ is thus trivial: we mimic a memoryless one w.l.o.g. and reach T *surely* in $\mathcal{M}$ within λ steps, ensuring the objective.

Hence, only the other direction remains: given a strategy σ such that $\mathsf{AS}^{\sigma}_{\mathcal{M}, v_{\mathrm{init}}}[\mathsf{DFW}_{\mathsf{par}}(\lambda)]$, we need to construct a winning strategy in $\mathcal{G}$. This may seem more difficult as we need to go from an almost-surely winning strategy to a surely winning one: something which is not possible in general. Yet, we show that σ is actually surely winning for $\mathsf{DFW}_{\mathsf{par}}(\lambda)$. By contradiction, assume it is not the case, i.e., that there exists a consistent run ρ such that $\rho \notin \mathsf{DFW}_{\mathsf{par}}(\lambda)$: then, a finite prefix $\rho[0, n]$ can be extracted, such that a window remains open for λ steps along it. Now, since we consider a direct objective, the cylinder spanned by this prefix only contains losing runs, and since $\rho[0, n]$ is finite, it has a strictly positive probability. Hence σ is not almost-surely winning and we have our contradiction. This proves that σ is actually surely-winning, and it is then trivial to mimic it in $\mathcal{G}$ to obtain a winning strategy for player 1. This shows that the reduction from reachability games to the threshold probability problem for direct fixed window parity objectives holds, thus that the latter is P-hard.

Regarding memory, the proof established for direct fixed window parity games in [15] carries over easily to our setting by replacing the states of the opponent by stochastic actions, in the natural way. Hence the lower bound is trivial to establish. Yet, we illustrate the need for memory in Ex. 13 to help the reader understand the phenomenon.

Item (b). Consider the PSPACE-hardness of the mean-payoff variant. We proceed via a reduction from the threshold probability problem for shortest path objectives [32,39]. This problem is as follows. Let $\mathcal{M} =$

(S, A, δ) be an MDP with weight function $w \colon A \to \mathbb{N}_0$ (we use strictly positive weights w.l.o.g.). We fix a target set $T \subseteq S$ and define the *truncated sum up to T* as the function $\mathsf{TS}^T \colon \mathsf{Runs}(\mathcal{M}) \to \mathbb{N} \cup \{\infty\}$ given by

$$\mathsf{TS}^T(\rho) = \begin{cases} \sum_{i=0}^{n-1} w(a_i) & \text{if } \rho[n] \in T \,\wedge\, \forall\, 0 \leq j < n,\ \rho[j] \notin T, \\ \infty & \text{if } \forall\, j \geq 0,\ \rho[j] \notin T. \end{cases}$$

Given an upper bound $\ell \in \mathbb{N}$, the shortest path objective is $\mathsf{ShortestPath}(\ell) = \{\rho \in \mathsf{Runs}(\mathcal{M}) \mid \mathsf{TS}^T(\rho) \leq \ell\}$. Deciding if there exists a strategy σ such that $\mathbb{P}^\sigma_{\mathcal{M},s}[\mathsf{ShortestPath}(\ell)] \geq \alpha$, given $s \in S$ and $\alpha \in [0,1] \cap \mathbb{Q}$, is known to be PSPACE-hard, even for *acyclic* MDPs [32]. The target set T is assumed to be made of absorbing states (i.e., with self-loops): the acyclicity is to be interpreted over the rest of the underlying graph. We establish a reduction from this problem, in the acyclic case, to a threshold probability problem for a direct fixed window mean-payoff objective, maintaining the acyclicity of the underlying graph (except in target states, again).

Given the original MDP $\mathcal{M}$, we only modify the weight function to obtain MDP $\mathcal{M}'$. We define w' from w, the target T and the bound ℓ as follows:

– for all $(s, a, s') \in \mathsf{Supp}(\delta)$, $s' \notin T$, $w'(a) = -w(a)$;
– for all $(s, a, t) \in \mathsf{Supp}(\delta)$, $s \notin T$, $t \in T$, $w'(a) = -w(a) + \ell$;
– for all $(t, a, t) \in \mathsf{Supp}(\delta)$, $t \in T$, $w'(a) = 0$.

Intuitively, we take the opposite of all weights; add the bound when entering the target; and make the target cost-free. We then define the objective $\mathsf{DFW}_{\mathsf{mp}}(\lambda = |S|)$ and claim that there exists a strategy σ in $\mathcal{M}$ to ensure $\mathbb{P}^\sigma_{\mathcal{M},s}[\mathsf{ShortestPath}(\ell)] \geq \alpha$ if and only if there exists a strategy σ' in $\mathcal{M}'$ to ensure $\mathbb{P}^{\sigma'}_{\mathcal{M}',s}[\mathsf{DFW}_{\mathsf{mp}}(\lambda)] \geq \alpha$.

Proving it is fairly easy. Observe that, by construction, the sum of weights over a prefix in $\mathcal{M}'$ that is not yet in T is strictly negative, and the opposite of the sum over the same prefix in the original MDP $\mathcal{M}$. Due to the addition of ℓ on entering T, we have that any run $\rho' \in \mathsf{Runs}(\mathcal{M}')$ sees all its windows closed if and only if the very same run $\rho \in \mathsf{Runs}(\mathcal{M})$ is such that $\mathsf{TS}^T(\rho) \leq \ell$ in $\mathcal{M}$. Now, using the acyclicity of the underlying graph, we know that if a run reaches T, it does so in at most λ steps. We thus conclude that $\rho' \in \mathsf{DFW}_{\mathsf{mp}}(\lambda)$ in $\mathcal{M}'$ if and only if $\rho \in \mathsf{ShortestPath}(\ell)$ in $\mathcal{M}$. It is then trivial to derive the desired result and establish the correctness of the reduction: this concludes our proof of PSPACE-hardness for the threshold probability problem for direct fixed window mean-payoff objectives.

Finally, the need for pseudo-polynomial memory is also obtained through this reduction. Indeed, there is a chain of reductions from *subset-sum games* [42,26] to our setting, via the shortest path problem presented above [32]. Subset-sum games require pseudo-polynomial-memory strategies, and it carries over to our setting. □

Remark 11. Throughout this paper, we assume the window size λ to be given in unary, or equivalently, to be polynomial in the description of the MDP (as for practical purposes, having exponential window sizes would somewhat defeat the purpose of time bounds in specifications). It is thus interesting to note that the hardness results we established in Thm. 10 do hold under this assumption: the complexity essentially comes from the weight structure, as in other counting-like problems in MDPs [32,39,14].

Remark 12. As noted in the proof of Thm. 10, almost-surely winning coincides with surely winning for the *direct fixed* window objectives. Therefore, the threshold probability problem for $\mathsf{DFW}_{\mathsf{mp}}(\lambda)$ collapses to P if $\alpha = 1$ [18].

Example 13. Consider the direct fixed window parity objective in the MDP depicted in Fig. 3, inspired by [15]. For the sake of readability, we did not write all details in the figure: all actions have uniform probability distributions over their successors, and all unlabeled states have priority $d = 6$. This example can easily be generalized to any even $d \geq 0$. Observe that the MDP has size $|S| = 2 + \frac{d}{2} \cdot (\frac{d}{2} + 1)$, hence polynomial in d.

Fix the objective $\mathsf{DFW}_{\mathsf{par}}(\lambda = \frac{d}{2} + 2)$. We claim that the controller may achieve it almost-surely (even surely) with a strategy using memory of size $\frac{d}{2}$. Indeed, each time action a results in taking the path of
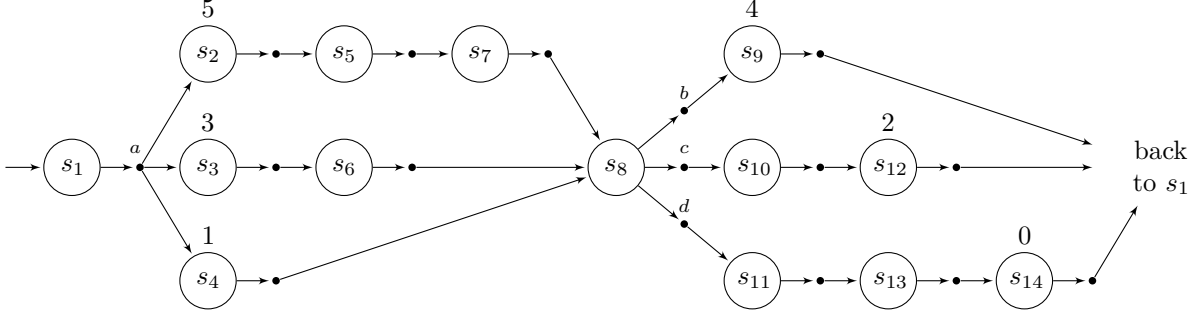
Fig. 3. This MDP with $d = 6$ priorities admits an almost-surely (even surely) winning strategy for $\mathsf{DFW}_{\mathsf{par}}(\lambda = \frac{d}{2} + 2)$: the controller must answer to the path with priority 5 (resp. 3, 1) by choosing b (resp. c, d). This requires $\mathcal{O}(\frac{d}{2})$ memory states.

priority $(d-1)$ (resp. $(d-3), \ldots, 1$), the only possibility for the controller is to choose the path with priority $(d-2)$ (resp. $(d-4), \ldots, 0$), otherwise a window stays open for λ steps. Doing this ensures that the objective is satisfied surely.

Now, if the controller uses less than $\frac{d}{2}$ memory, he has to answer to two different odd priorities with the same choice of action in s_8, which results in a window staying open for λ steps with strictly positive probability. Hence, there is no almost-sure winning strategy with such limited memory: polynomial-memory strategies are needed in this example. ◁

5 The case of end-components

We have already solved the case of direct fixed window objectives: it remains to consider prefix-independent fixed and bounded variants. As seen in Sect. 2, the analysis of MDPs with prefix-independent objectives crucially relies on *end-components* (ECs). Indeed, they are almost-surely reached in the long run.

In this section, we study what happens in ECs: how to play optimally and what can be achieved. In Sect. 6, we will use this knowledge as the cornerstone of our algorithm for general MDPs. The main result here is a *strong link between ECs and two-player games*: intuitively, either the probability to win a window objective in an EC is zero, or it is one and there exists a sub-EC where the controller can actually win surely, i.e., as in a two-player game played on this sub-EC.

Safe ECs. We start by defining the notion of λ-safety, that will characterize such sub-ECs.

Definition 14 (λ-safety). *Let $\mathcal{M}$ be an MDP, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\lambda > 0$, and $\mathcal{C} = (S_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{EC}(\mathcal{M})$, we say that $\mathcal{C}$ is λ-safe$_\Omega$ if there exists a strategy $\sigma \in \Sigma$ in $\mathcal{C}$ such that, from all $s \in S_\mathcal{C}$, $\mathsf{S}^\sigma_{\mathcal{C},s}[\mathsf{DFW}_\Omega(\lambda)]$.*

Classifying an EC as λ-safe$_\Omega$ or not boils down to interpreting it as a *two-player game* (the duality between MDPs and games is further explored in [14,7]). In this setting, the uncertainty becomes adversarial: when there is some uncertainty about the outcome of an action, we do not consider probabilities but we let the opponent decide the outcome of the action. On entering a state s of the MDP after some history, the controller chooses an action a following its strategy and the opponent then chooses a successor state s' such that $s' \in \mathsf{Supp}(\delta(s, a))$ without taking into account the exact values of probabilities. In such a view, the opponent tries to prevent the controller from achieving its objective. A *winning strategy* for the controller in the game interpretation is a strategy that ensures the objective regardless of its opponent's strategy. An EC is thus said to be λ-safe$_\Omega$ if and only if its two-player interpretation admits a winning strategy for $\mathsf{DFW}_\Omega(\lambda)$.

Remark 15. Throughout this paper, all the strategies we consider are uniform in the game-theoretic sense. That is, if an objective is achievable from a set of states, we do not need to have a different strategy for each

starting state, and we may instead use the very same strategy from all starting states. This uniformity is not needed in our approach but we have for free for both the classical MDP strategies (reachability, Büchi, etc) [4] and the two-player window games [18,15]. We use it for the sake of readability, as it permits to have one strategy per EC instead of one per state of the EC for example. Hence all our statements are written with that in mind.

Proposition 16. *Let $\mathcal{M}$ be an MDP, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\lambda > 0$, and $\mathcal{C} = (S_{\mathcal{C}}, A_{\mathcal{C}}, \delta_{\mathcal{C}}) \in \mathsf{EC}(\mathcal{M})$ be λ-safe$_\Omega$. Then, there exists a pure polynomial-memory strategy $\sigma_{safe}^{\Omega,\lambda,\mathcal{C}}$ in $\mathcal{C}$ such that $\mathsf{S}_{\mathcal{C},s}^{\sigma_{safe}^{\Omega,\lambda,\mathcal{C}}}[\mathsf{DFW}_\Omega(\lambda)]$ for all $s \in S_{\mathcal{C}}$.*

Proof. Straightforward by definition of λ-safety and pure polynomial-memory strategies being sufficient in two-player zero-sum direct fixed window games, both for mean-payoff [18] and parity [15] variants.

Good ECs. As sketched above, the existence of sub-ECs that are λ-safe is crucial in order to satisfy any window objective in an EC. We thus introduce the notion of *good* ECs.

Definition 17. *Let $\mathcal{M}$ be an MDP, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, and $\mathcal{C} \in \mathsf{EC}(\mathcal{M})$, we say that*

- $\mathcal{C}$ *is λ-good$_\Omega$, for $\lambda > 0$, if it contains a sub-EC $\mathcal{C}'$ which is λ-safe$_\Omega$.*
- $\mathcal{C}$ *is BW-good$_\Omega$ if it contains a sub-EC $\mathcal{C}'$ which is λ-safe$_\Omega$ for some $\lambda > 0$.*

By definition, any BW-good$_\Omega$ EC is also λ-good$_\Omega$ for an appropriate $\lambda > 0$. Yet, we use a different terminology as in the BW case, we do not fix λ a priori: this will be important complexity-wise.

We now establish that good$_\Omega$ ECs are exactly the ones where window objectives can be satisfied with non-zero probability, and actually, with probability one.

Lemma 18 (Zero-one law). *Let $\mathcal{M}$ be an MDP, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$ and $\mathcal{C} = (S_{\mathcal{C}}, A_{\mathcal{C}}, \delta_{\mathcal{C}}) \in \mathsf{EC}(\mathcal{M})$. The following assertions hold.*

(a) For all $\lambda > 0$,
  (i) either $\mathcal{C}$ is λ-good$_\Omega$ and there exists a strategy σ in $\mathcal{C}$ such that $\mathsf{AS}_{\mathcal{C},s}^{\sigma}[\mathsf{FW}_\Omega(\lambda)]$ for all $s \in S_{\mathcal{C}}$,
  (ii) or for all $s \in S_{\mathcal{C}}$, for all strategy σ in $\mathcal{C}$, $\mathbb{P}_{\mathcal{C},s}^{\sigma}[\mathsf{FW}_\Omega(\lambda)] = 0$.
(b) (i) Either $\mathcal{C}$ is BW-good$_\Omega$ and there exists a strategy σ in $\mathcal{C}$ such that $\mathsf{AS}_{\mathcal{C},s}^{\sigma}[\mathsf{BW}_\Omega]$ for all $s \in S_{\mathcal{C}}$,
  (ii) or for all $s \in S_{\mathcal{C}}$, for all strategy σ in $\mathcal{C}$, $\mathbb{P}_{\mathcal{C},s}^{\sigma}[\mathsf{BW}_\Omega] = 0$.

Proof. We begin with the fixed variant (a). Case (i). Fix $\lambda > 0$ and assume there exists a sub-EC $\mathcal{C}'$ with state space $S_{\mathcal{C}'}$ that is λ-safe$_\Omega$. By Prop. 16, there exists a strategy $\sigma_{\mathrm{safe}}^{\Omega,\lambda,\mathcal{C}'}$ in $\mathcal{C}'$ such that $\mathsf{S}_{\mathcal{C}',s}^{\sigma_{\mathrm{safe}}^{\Omega,\lambda,\mathcal{C}'}}[\mathsf{DFW}_\Omega(\lambda)]$ for all $s \in S_{\mathcal{C}'}$. Now, since $\mathcal{C}$ is an EC, there exists a (pure memoryless) strategy σ_{reach} in $\mathcal{C}$ that ensures eventually reaching $\mathcal{C}'$ almost-surely from any state $s \in S_{\mathcal{C}}$ [4]. Hence, the desired strategy σ can be defined as follows: play according to σ_{reach} until $\mathcal{C}'$ is reached, then switch to $\sigma_{\mathrm{safe}}^{\Omega,\lambda,\mathcal{C}'}$ forever. It is straightforward to check that σ almost-surely satisfies $\mathsf{FW}_\Omega(\lambda)$ from anywhere in $\mathcal{C}$, thanks to prefix-independence. Note that it does not ensure it surely in general, as σ_{reach} does not guarantee to reach $\mathcal{C}'$ surely either.

Case (ii). Now assume that such a λ-safe$_\Omega$ sub-EC does not exist. Recall that finite-memory strategies suffice for $\mathsf{FW}_\Omega(\lambda)$ objectives by Thm. 8, hence we can restrict our study to such strategies without loss of generality. Fix any finite-memory strategy σ in $\mathcal{C}$ and state $s \in S_{\mathcal{C}}$. The induced MC $\mathcal{M}_s^\sigma$ is finite, hence runs almost-surely end up in a BSCC [4]. Let $\mathcal{B}$ be any BSCC of $\mathcal{M}_s^\sigma$ reached with positive probability. Since there exists no λ-safe$_\Omega$ sub-EC in $\mathcal{C}$, there must exist a run $\widehat{\rho}$ in $\mathcal{B}$ such that $\widehat{\rho} \notin \mathsf{DFW}_\Omega(\lambda)$ – otherwise, σ would be witness that the EC obtained by projecting $\mathcal{B}$ over $S_{\mathcal{C}}$ is λ-safe$_\Omega$. From $\widehat{\rho}$, we extract a history $\widehat{h}$ ending with a window open for λ steps (it exists otherwise $\widehat{\rho}$ would be in $\mathsf{DFW}_\Omega(\lambda)$). This history is finite: it has a probability lower-bounded by some $\varepsilon > 0$ to happen whenever its starting state is visited. Now, since all states in $\mathcal{B}$ are almost-surely visited infinitely often, we conclude that this history also happens infinitely often with probability one. Therefore, the probability to win the prefix-independent objective $\mathsf{FW}_\Omega(\lambda)$ when reaching $\mathcal{B}$ is zero. Since this holds for any BSCC induced by σ, we obtain the claim.

Let us continue with the bounded case (b). Case (i) is trivial thanks to (a)(i) and $\mathsf{FW}_\Omega(\lambda) \subseteq \mathsf{BW}_\Omega$. Now consider case (ii). By (a)(ii), we have that $\mathbb{P}^\sigma_{\mathcal{C},s}[\mathsf{FW}_\Omega(\lambda)] = 0$ for all $s \in S_\mathcal{C}$, $\lambda > 0$ and σ in $\mathcal{C}$. Observe that by definition, we have

$$\mathsf{BW}_\Omega = \bigcup_{\lambda > 0} \mathsf{FW}_\Omega(\lambda).$$

Fix any strategy σ in $\mathcal{C}$ and $s \in S_\mathcal{C}$. By countable additivity of probability measures, we have that

$$\mathbb{P}^\sigma_{\mathcal{C},s}[\mathsf{BW}_\Omega] \ \leq \ \sum_{\lambda > 0} \mathbb{P}^\sigma_{\mathcal{C},s}[\mathsf{FW}_\Omega(\lambda)].$$

Since the right term is equal to zero, we obtain the claim. $\qquad\square$

Remark 19. An interesting consequence of Lem. 18 is the existence of uniform bounds on λ in ECs, in contrast to the general MDP case, as seen in Sect. 3.3. This is indeed natural, as we established that winning with positive probability within an EC coincides with winning surely in a sub-EC; sub-EC that can be seen as a two-player zero-sum game where uniform bounds are granted by [18,15].

By Lem. 18, we know that interesting strategies exist in good$_\Omega$ ECs. Let us describe them.

Proposition 20. *Let $\mathcal{M}$ be an MDP, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, and $\mathcal{C} = (S_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{EC}(\mathcal{M})$.*

- *If $\mathcal{C}$ is λ-good$_\Omega$, for some $\lambda > 0$, there exists a pure polynomial-memory strategy $\sigma^{\Omega,\lambda,\mathcal{C}}_{good}$ such that $\mathsf{AS}^{\sigma^{\Omega,\lambda,\mathcal{C}}_{good}}_{\mathcal{C},s}[\mathsf{FW}_\Omega(\lambda)]$ for all $s \in S_\mathcal{C}$.*
- *If $\mathcal{C}$ is BW-good$_\Omega$, there exists a pure memoryless strategy $\sigma^{\Omega,\mathsf{BW},\mathcal{C}}_{good}$ such that $\mathsf{AS}^{\sigma^{\Omega,\mathsf{BW},\mathcal{C}}_{good}}_{\mathcal{C},s}[\mathsf{BW}_\Omega]$ for all $s \in S_\mathcal{C}$.*

Intuitively, such strategies first mimic a pure memoryless strategy reaching a safe$_\Omega$ sub-EC almost-surely, then switch to a strategy surely winning in this sub-EC, which is lifted from the game interpretation.

Proof. Consider the λ-good$_\Omega$ case. Strategy $\sigma^{\Omega,\lambda,\mathcal{C}}_{good}$ is the one described in the proof of Lem. 18(a)(i): it first plays as the pure memoryless strategy σ_{reach} then switches to strategy $\sigma^{\Omega,\lambda,\mathcal{C}'}_{safe}$ from Prop. 16 when the λ-safe$_\Omega$ sub-EC $\mathcal{C}'$ is reached. Hence it is pure and polynomial memory still suffices. The almost-sure satisfaction of the objective was proved in Lem. 18.

Now, consider the BW-good$_\Omega$ case. Note that using our current knowledge, it is easy to build a pure polynomial-memory strategy, but we want more: a pure *memoryless* one. For that, we use the following reasoning. Strategy $\sigma^{\Omega,\mathsf{BW},\mathcal{C}}_{good}$ will again first play as a pure memoryless strategy trying to reach a λ-safe$_\Omega$ sub-EC for some $\lambda > 0$ (we know one exists), then switch to a surely winning strategy inside this sub-EC. Still, we do not really need to win for $\mathsf{DFW}_\Omega(\lambda)$, we only need to win for the bounded variant $\mathsf{DBW}_\Omega = \bigcup_{\lambda > 0} \mathsf{DFW}_\Omega(\lambda)$. Results in two-player zero-sum games (which we are actually solving here when considering surely winning strategies) guarantee that in this case, pure memoryless strategies suffice for the controller, both for mean-payoff [18] and parity variants [15]. Since the switch is state-based (depending on whether we are in the sub-EC or not), we can wrap both strategies in $\sigma^{\Omega,\mathsf{BW},\mathcal{C}}_{good}$ and retain a pure memoryless strategy. Again, the almost-sure satisfaction of the objective follows from Lem. 18. $\qquad\square$

Classification. We may already sketch a general solution to the threshold probability problem based on Lem. 18 and the well-known fact that ECs are almost-surely reached under any strategy: an optimal strategy must maximize the probability to reach good$_\Omega$ ECs. It is therefore crucial to be able to identify such ECs efficiently. However, an MDP may in general contain an exponential number of ECs. Fortunately, the next lemma establishes that we do not have to test them all.

Lemma 21. *Let $\mathcal{M}$ be an MDP and $\mathcal{C} \in \mathsf{EC}(\mathcal{M})$. If $\mathcal{C}$ is λ-good$_\Omega$ (resp. BW-good$_\Omega$), then it is also the case of any super-EC $\mathcal{C}' \in \mathsf{EC}(\mathcal{M})$ containing $\mathcal{C}$.*

Proof. Trivial by Def. 17. □

Corollary 22. *Let $\mathcal{M}$ be an MDP and $\mathcal{C} \in \mathsf{MEC}(\mathcal{M})$ be a maximal EC. If $\mathcal{C}$ is not λ-good$_\Omega$ (resp. BW-good$_\Omega$), then neither is any of its sub-EC $\mathcal{C}' \in \mathsf{EC}(\mathcal{M})$.*

The interest of Cor. 22 is that the number of MECs is bounded by $|S|$ for any MDP $\mathcal{M} = (S, A, \delta)$ because they are all disjoints. Furthermore, the MEC decomposition can be done efficiently (e.g., quadratic time [19]). So, we know classifying MECs is sufficient and MECs can easily be identified in an MDP: it remains to discuss how to classify a MEC as good$_\Omega$ or not.

Let $\mathcal{M} = (S, A, \delta)$. Recall that a MEC $\mathcal{C} = (S_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{MEC}(\mathcal{M})$ is λ-good$_\Omega$ (resp. BW-good$_\Omega$) if and only if it contains a λ-safe$_\Omega$ sub-EC. By definition of λ-safety, this is equivalent to having a non-empty *winning set* for the controller in the two-player zero-sum game over $\mathcal{C}$ – naturally defined as explained above. This winning set contains all states in $S_\mathcal{C}$ from which the controller has a *surely winning* strategy. This winning set, if non-empty, necessarily contains at least one sub-EC of $\mathcal{C}$, as otherwise the opponent could force the controller to leave it and win the game (by prefix-independence). Thus, testing if a MEC is good$_\Omega$ boils down to solving its two-player game interpretation.

Theorem 23 (MEC classification). *Let $\mathcal{M}$ be an MDP and $\mathcal{C} \in \mathsf{MEC}(\mathcal{M})$. The following assertions hold.*

(a) *Deciding if $\mathcal{C}$ is λ-good$_\Omega$, for $\lambda > 0$, is in P for $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$. Furthermore, a corresponding pure polynomial-memory strategy $\sigma_{good}^{\Omega, \lambda, \mathcal{C}}$ can be constructed in polynomial time.*

(b) *Deciding if $\mathcal{C}$ is BW-good$_\mathsf{mp}$ is in $\mathsf{NP} \cap \mathsf{coNP}$ and a corresponding pure memoryless strategy $\sigma_{good}^{\mathsf{mp}, \mathsf{BW}, \mathcal{C}}$ can be constructed in pseudo-polynomial time.*

(c) *Deciding if $\mathcal{C}$ is BW-good$_\mathsf{par}$ is in P and a corresponding pure memoryless strategy $\sigma_{good}^{\mathsf{par}, \mathsf{BW}, \mathcal{C}}$ can be constructed in polynomial time.*

Proof. All complexities are expressed w.r.t. the representation of $\mathcal{C}$, as the larger MDP $\mathcal{M}$ is never used in the process. Complexity results follow from game solving algorithms presented in [18] for mean-payoff[6] and [15] for parity. Note that for mean-payoff, the bound on λ for which the fixed and bounded variants coincide is pseudo-polynomial, whereas for parity this bound is polynomial, hence the different results: the algorithm for bounded window mean-payoff games actually bypasses this bound to obtain $\mathsf{NP} \cap \mathsf{coNP}$ membership instead of simply $\mathsf{EXPTIME}$.

For constructing the strategy, we only need to plug the almost-sure-reachability strategy to the surely-winning one, as presented in Prop. 20. Such a reachability strategy can easily be computed in polynomial time [4], hence the complexity is dominated by the cost of computing the surely-winning one, as presented in [18,15]. □

6   General MDPs

Algorithms. We now have all the ingredients to establish an algorithm for the threshold probability problem in the general case. Intuitively, given an MDP $\mathcal{M}$, an initial state s and a window objective $\mathsf{FW}_\Omega(\lambda)$ for $\lambda > 0$ (resp. BW_Ω), we first compute the MEC decomposition of $\mathcal{M}$, then classify each MEC as λ-good$_\Omega$ (resp. BW-good$_\Omega$) or not, and finally compute an optimal strategy from s to reach the union of good$_\Omega$ MECs: the probability of reaching such MECs is then exactly the maximum probability to satisfy the window objective.

The fixed and bounded versions are presented in Fig. 4. Let $\mathcal{M} = (S, A, \delta)$ be the MDP. The MEC decomposition (Line 2) takes quadratic time [19] and yields at most $|S|$ MECs. The classification depends on the variant considered, as established in Thm. 23: it is in P for fixed variants and bounded window parity, and in $\mathsf{NP} \cap \mathsf{coNP}$ for bounded window mean-payoff (with a pseudo-polynomial-time procedure). Finally, sub-procedure $\mathsf{MaxReachability}(s, T)$ computes the maximum probability to reach the set T from s. It is

17

Algorithm 1 FixedWindow($\mathcal{M}, s, \Omega, \lambda$)	**Algorithm 2** BoundedWindow($\mathcal{M}, s, \Omega$)
Input: MDP $\mathcal{M}$, state s, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\lambda > 0$	**Input:** MDP $\mathcal{M}$, state s, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$
Output: Maximum probability of $\mathsf{FW}_\Omega(\lambda)$ from s	**Output:** Maximum probability of BW_Ω from s
1 $T \leftarrow \emptyset$	1 $T \leftarrow \emptyset$
2 **for all** $\mathcal{C} = (S_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{MEC}(\mathcal{M})$ **do**	2 **for all** $\mathcal{C} = (S_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{MEC}(\mathcal{M})$ **do**
3 **if** $\mathcal{C}$ is λ-good$_\Omega$ **then**	3 **if** $\mathcal{C}$ is BW-good$_\Omega$ **then**
4 $T \leftarrow T \uplus S_\mathcal{C}$	4 $T \leftarrow T \uplus S_\mathcal{C}$
5 $\nu = \mathsf{MaxReachability}(s, T)$	5 $\nu = \mathsf{MaxReachability}(s, T)$
6 **return** ν	6 **return** ν

Fig. 4. Algorithms computing the maximum probability of fixed and bounded window objectives in general MDPs.

well-known that this can be done in polynomial time and that pure memoryless optimal strategies exist [4]. Therefore the overall complexity of the algorithm is dominated by the classification step.

Correctness. We first prove that these algorithms are sound and complete.

Lemma 24. *Alg. 1 and Alg. 2 are correct: given an MDP $\mathcal{M} = (S, A, \delta)$, an initial state $s \in S$, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\lambda > 0$, we have that*

$$\mathsf{FixedWindow}(\mathcal{M}, s, \Omega, \lambda) = \max_{\sigma \in \Sigma} \mathbb{P}^\sigma_{\mathcal{M},s}[\mathsf{FW}_\Omega(\lambda)],$$

$$\mathsf{BoundedWindow}(\mathcal{M}, s, \Omega) = \max_{\sigma \in \Sigma} \mathbb{P}^\sigma_{\mathcal{M},s}[\mathsf{BW}_\Omega].$$

Proof. The proof is straightforward based on our previous results. First, recall that objectives $\mathsf{FW}_\Omega(\lambda)$ and BW_Ω are prefix-independent; that in MDPs, the limit-behavior under any strategy almost-surely coincides with an EC (see Sect. 2); and that MECs are pair-wise disjoint. We thus have that

$$\sup_{\sigma \in \Sigma} \mathbb{P}^\sigma_{\mathcal{M},s}[\mathsf{FW}_\Omega(\lambda)] = \sup_{\sigma \in \Sigma} \sum_{\mathcal{C} \in \mathsf{MEC}(\mathcal{M})} \mathbb{P}^\sigma_{\mathcal{M},s}[\Diamond\Box\mathcal{C}] \cdot \mathbb{P}^\sigma_{\mathcal{C}}[\mathsf{FW}_\Omega(\lambda)],$$

$$\sup_{\sigma \in \Sigma} \mathbb{P}^\sigma_{\mathcal{M},s}[\mathsf{BW}_\Omega] = \sup_{\sigma \in \Sigma} \sum_{\mathcal{C} \in \mathsf{MEC}(\mathcal{M})} \mathbb{P}^\sigma_{\mathcal{M},s}[\Diamond\Box\mathcal{C}] \cdot \mathbb{P}^\sigma_{\mathcal{C}}[\mathsf{BW}_\Omega],$$

where $\Diamond\Box\mathcal{C}$ uses the standard LTL notation as a shortcut for

$$\{\rho = s_0 a_0 s_1 \ldots \in \mathsf{Runs}(\mathcal{M}) \mid \exists\, i \geq 0, \forall\, j > i, \, s_j \in S_\mathcal{C}\}$$

with $S_\mathcal{C}$ the set of states of $\mathcal{C}$; and where we write $\mathbb{P}_\mathcal{C}$ without distinction on the entry point because the supremum probability to win for a prefix-independent objective is identical in all states of a MEC (as the controller may force reaching any state he wants almost-surely).

By Lem. 18, we know that *supremum* winning probabilities in ECs are zero-one laws: they are equal to one in good$_\Omega$ ECs and to zero in all other ECs. Furthermore, when almost-sure satisfaction is achievable, the corresponding strategy stays in the EC. Let us denote by $\mathsf{LGEC}_\Omega(\mathcal{M}, \lambda)$ and $\mathsf{BWGEC}_\Omega(\mathcal{M})$ the sets of λ-good$_\Omega$ and BW-good$_\Omega$ MECs of $\mathcal{M}$ respectively, and by L and B the disjoint union of the corresponding state spaces. That is,

$$L = \biguplus_{\mathcal{C} = (S_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{LGEC}_\Omega(\mathcal{M}, \lambda)} S_\mathcal{C}, \qquad B = \biguplus_{\mathcal{C} = (S_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{BWGEC}_\Omega(\mathcal{M})} S_\mathcal{C}.$$

---

[6] The published version of [18] contains a slight bug in sub-procedure GoodWin that was corrected in subsequent articles [16,15]. All results of [18] still hold modulo this correction.

We have that

$$\sup_{\sigma \in \Sigma} \mathbb{P}^{\sigma}_{\mathcal{M},s}[\mathsf{FW}_{\Omega}(\lambda)] = \sup_{\sigma \in \Sigma} \sum_{\mathcal{C} \in \mathsf{LGEC}_{\Omega}(\mathcal{M}, \lambda)} \mathbb{P}^{\sigma}_{\mathcal{M},s}[\lozenge \square \mathcal{C}] = \sup_{\sigma \in \Sigma} \mathbb{P}^{\sigma}_{\mathcal{M},s}[\lozenge L],$$

$$\sup_{\sigma \in \Sigma} \mathbb{P}^{\sigma}_{\mathcal{M},s}[\mathsf{BW}_{\Omega}] = \sup_{\sigma \in \Sigma} \sum_{\mathcal{C} \in \mathsf{BWGEC}_{\Omega}(\mathcal{M})} \mathbb{P}^{\sigma}_{\mathcal{M},s}[\lozenge \square \mathcal{C}] = \sup_{\sigma \in \Sigma} \mathbb{P}^{\sigma}_{\mathcal{M},s}[\lozenge B].$$

Now, observe that the term on the right-hand side is exactly the probability computed by our algorithm through the call to sub-procedure MaxReachability on Line 5: our algorithm is thus correct, as it computes the supremum probability to achieve the corresponding window objective.

It remains to establish that this probability is actually a *maximum* probability, as claimed. Indeed, there exists a strategy to achieve the probability ν returned by our algorithm. From s, we know that pure memoryless optimal strategies exist to reach the computed set T (which coincides with L or B depending on the considered variant), and inside T, there exist pure finite-memory optimal strategies for all good$_{\Omega}$ MECs, as shown in Thm. 23. We can easily combine all these strategies in an optimal strategy for $\mathcal{M}$, which concludes our proof. □

Complexity. We wrap-up our results on prefix-independent window objectives in the following theorem.

Theorem 25. *The threshold probability problem is*

(a) *in* P *for fixed window parity objectives and fixed window mean-payoff objectives, and pure polynomial-memory optimal strategies can be constructed in polynomial time;*
(b) *in* P *for bounded window parity objectives, and pure memoryless optimal strategies can be constructed in polynomial time;*
(c) *in* NP ∩ coNP *for bounded window mean-payoff objectives, and pure memoryless optimal strategies can be constructed in pseudo-polynomial time.*

Proof. The complexity results follow from the analysis we made before:

– the MEC decomposition (Line 2) takes quadratic time [19] and yields at most $|S|$ MECs;
– classifying a MEC (Line 3) is in P for fixed variants and bounded window parity, and in NP ∩ coNP for bounded window mean-payoff (Thm. 23);
– MaxReachability(s, T) (Line 5) takes polynomial time [4].

Overall we have P-membership for all variants except bounded window mean-payoff, where we are in $\mathsf{P}^{\mathsf{NP} \cap \mathsf{coNP}}$, which is equal to NP ∩ coNP [10].

Regarding optimal strategies, they are by-products of Alg. 1 and Alg. 2. Inside good$_{\Omega}$ MECs, we play the strategy granted by Thm. 23, and outside, we simply play a pure memoryless optimal reachability strategy obtained through sub-procedure MaxReachability(s, T) [4]. □

Lower bounds. We complement the results of Thm. 25 with matching lower bounds, showing that our approach is optimal complexity-wise.

Theorem 26. *The threshold probability problem is*

(a) P-*hard for fixed window parity objectives and fixed window mean-payoff objectives, and polynomial-memory strategies are in general necessary;*
(b) P-*hard for bounded window parity objectives;*
(c) *as hard as mean-payoff games for bounded window mean-payoff objectives.*

Proof. Complexity-wise, the lower bounds follow from the results in two-player zero-sum window mean-payoff [18] and window parity [15] games, coupled with the equivalence between the threshold probability problem and solving a two-player game established in Sect. 5. Note that formally, this equivalence only holds if the MDP is an EC, hence we have to make sure that the corresponding game problems retain their

hardness when considered over arenas that correspond to ECs (i.e., when the underlying graph is strongly connected). Fortunately, careful inspection of the related results in [18,15] shows that this is the case. Note that mean-payoff games are widely considered as a canonical "hard" problem for the class $\mathsf{NP} \cap \mathsf{coNP}$ (see, e.g., [18]).

Regarding strategies, the fixed case is the only one where memory is needed, as stated in Thm. 25. Again, the necessity of polynomial memory is witnessed through the equivalence with two-player games in ECs, proved in Sect. 5, along with results on fixed window mean-payoff and fixed window parity games [18,15].

□

7   Limitations and perspectives

Recall that we summarized our results in Table 1, and compared them to the state of the art in Sect. 1. The goal of this section is to discuss the limitations of our work and some extensions within arm's reach.

Direct bounded window objectives. We left out a specific variant of window objectives that was considered in games [18,15]: the *direct bounded* variant, defined as $\mathsf{DBW}_\Omega = \bigcup_{\lambda > 0} \mathsf{DFW}_\Omega(\lambda)$ (see Sect. 3.3). This variant is maybe not the most natural as it is *not* prefix-independent, yet allows to close the windows of a run in an arbitrarily large – but bounded along the run – number of steps.

This variant gives rise to complex behaviors in MDPs, notably due to its interaction with the almost-sure reachability of ECs. Let us illustrate it on an example.

Example 27. Consider the MDP in Fig. 5 and objective $\mathsf{DBW}_{\mathsf{mp}}$. A window opens in the first step due to action a. The only way to close it is to loop in s_2 using action b up to the point where the running sum of weights becomes non-negative. Note also that when it does, all windows are closed and the controller may safely switch to s_5. Now, observe that taking action b repeatedly induces a *symmetric random walk* [31]. Classical probability results ensure that a non-negative sum will be obtained almost-surely, but the number of times b is played must remain unbounded (as for any bounded number, there exists a strictly positive probability to obtain only -1's for example). Therefore, in this example, there exists an infinite-memory strategy σ such that $\mathsf{AS}^\sigma_{\mathcal{M},s_1}[\mathsf{DBW}_{\mathsf{mp}}]$, but no finite-memory strategy can do as good.

Now, if we modify the probabilities in b to $\delta(s_2, b) = \{s_3 \mapsto 0.6, s_4 \mapsto 0.4\}$, the random walk becomes *asymmetric*, with a strictly positive chance to diverge toward $-\infty$. While the best possible strategy is still the one defined above, it only guarantees a probability strictly less than one to satisfy the objective. ◁
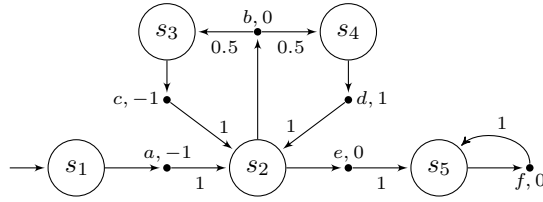

Fig. 5. There exists a strategy σ ensuring $\mathsf{AS}^\sigma_{\mathcal{M},s_1}[\mathsf{DBW}_{\mathsf{mp}}]$ but it requires infinite memory as it needs to use b up to the point where the running sum becomes non-negative, then switch to e.

What do we observe? First, *infinite-memory strategies are required*, which is a problem for practical applications. Second, even for *qualitative* questions (is the probability zero or one?), the actual probabilities of the MDP must be considered, not only the existence of a transition. This is in stark contrast to most problems in MDPs [4]: in that sense, the direct bounded window objective is not well-behaved. This is due to the connection with random walks we just established. Imagine that we replace the gadget related to action b by a much more complex EC: the corresponding random walk will be quite tedious to analyze. It is well-known that complex random walks are difficult to tackle for verification and synthesis. For example, even simple asymmetric random walks, like the one we sketched, are not *decisive* MCs, a large and robust class of MCs where reachability questions can be answered [1].

Markov chains. Our work focuses on the threshold probability problem for MDPs, and the corresponding strategy synthesis problem. Better complexities could possibly be obtained in the case of MCs, where there are no non-deterministic choices. To achieve this, a natural direction would be to focus on the classification of ECs (Sect. 5), the bottleneck of our approach: for MCs, this classification would involve *one-player window games* (for the opponent), whose precise complexity has yet to be explored and would certainly be lower than for two-player games.

Note however that complexity classes in the MC case are unlikely to be much lower: all parity variants are already in P, and the high complexity of the direct fixed window mean-payoff case would remain: a construction similar to the PSPACE-hardness proof (Thm. 10) easily shows this problem to be PP-hard, already for *acyclic* MCs (again using [32]). Let us recall that PP-hard problems are widely believed to be outside NP, as otherwise the polynomial hierarchy would collapse to P^{NP} by Toda's theorem [41].

Expected value problem. Given an MDP $\mathcal{M}$ and an initial state s, we may be interested in synthesizing a strategy σ that minimizes the *expected window size* for a fixed window objective (say $\mathsf{FW}_\Omega(\lambda)$ for the sake of illustration), which we straightforwardly define as

$$\mathbb{E}^\sigma_{\mathcal{M},s,\Omega}(\lambda) = \sum_{\lambda>0}^\infty \lambda \cdot \mathbb{P}^\sigma_{\mathcal{M},s}[\mathsf{FW}_\Omega(\lambda) \setminus \mathsf{FW}_\Omega(\lambda-1)],$$

with $\mathsf{FW}_\Omega(0) = \emptyset$. This meets the natural desire to build strategies that strive to maintain the *best time bounds possible in their local environment* (e.g., EC of $\mathcal{M}$). Note that this is totally different from the value function used in [8].

For prefix-independent variants, *we already have all the necessary machinery* to solve this problem. First, we refine the classification process to identify the best window size achievable in each MEC, if any. Indeed, if a MEC is λ-good, it necessarily is for some λ between one and the upper bound derived from the game-theoretic interpretation (Rmk. 19): we determine the smallest value of λ for each MEC via a binary search coupled with the classification procedure. Second, using classical techniques (e.g., [39]), we contract each MEC to a single-state EC, and give it a weight that represents the best window size we can ensure in it (hence this weight may be infinite if a MEC is not BW-good). Finally, we construct a global strategy that favors reaching MECs with the lowest weights, for example by synthesizing a strategy minimizing the classical mean-payoff value. Note that if λ-good MECs cannot be reached almost-surely, the expected value will be infinite, as wanted. Observe that *such an approach maintains tractability*, as we end up with a polynomial-time algorithm.

Direct variants would require more involved techniques, as the unfoldings developed in Sect. 4 are strongly linked to the fixed window size λ, and cannot be that easily combined for different values of λ.

Multi-objective problems. Window games have also been considered in the multidimension setting, where several weight (resp. priority) functions are given, and the objective is defined as the intersection of all one-dimension objectives [18,15]. Again, *our generic approach supports effortless extension to this setting*.

In the direct case, the unfoldings of Sect. 4 can easily be generalized to multiple dimensions, as in [18,15]. For prefix-independent variants, the EC classification needs to be adapted to handle multidimension window games, which we can solve using the techniques of [18,15]. Then, we also need to consider a multi-objective reachability problem [39]. While almost all cases of multidimension window games are EXPTIME-complete, note that the decidability of the bounded mean-payoff case is still open – it is however known to be non-primitive recursive hard.

Tool support. Thanks to its low complexity and its adequacy w.r.t. applications, our window framework lends itself well to tool development. We are currently building a tool suite for MDPs with window objectives based on the main results of this paper along with the aforementioned extensions. Our aim is to provide a dedicated extension of STORM, a cutting-edge probabilistic model checker [24].

References

1. Parosh Aziz Abdulla, Noomene Ben Henda, and Richard Mayr. Decisive Markov chains. *Logical Methods in Computer Science*, 3(4), 2007.

2. Pranav Ashok, Krishnendu Chatterjee, Przemyslaw Daca, Jan Kretínský, and Tobias Meggendorfer. Value iteration for long-run average reward in Markov decision processes. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 201–221. Springer, 2017.

3. Christel Baier. Reasoning about cost-utility constraints in probabilistic models. In Mikolaj Bojanczyk, Slawomir Lasota, and Igor Potapov, editors, *Reachability Problems - 9th International Workshop, RP 2015, Warsaw, Poland, September 21-23, 2015, Proceedings*, volume 9328 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2015.

4. Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

5. Christel Baier, Joachim Klein, Sascha Klüppelholz, and Sascha Wunderlich. Weight monitoring with linear temporal logic: complexity and decidability. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 11:1–11:10. ACM, 2014.

6. Catriel Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Trans. Database Syst.*, 5(3):241–259, 1980.

7. Raphaël Berthon, Mickael Randour, and Jean-François Raskin. Threshold constraints with guarantees for parity objectives in Markov decision processes. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 121:1–121:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

8. Benjamin Bordais, Shibashis Guha, and Jean-François Raskin. Expected window mean-payoff. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pages 32:1–32:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

9. Patricia Bouyer, Mauricio González, Nicolas Markey, and Mickael Randour. Multi-weighted Markov decision processes with reachability objectives. In Andrea Orlandini and Martin Zimmermann, editors, *Proceedings Ninth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2018, Saarbrücken, Germany, 26-28th September 2018.*, volume 277 of *EPTCS*, pages 250–264, 2018.

10. Gilles Brassard. A note on the complexity of cryptography (corresp.). *IEEE Trans. Information Theory*, 25(2):232–233, 1979.

11. Tomás Brázdil, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kucera. Trading performance for stability in Markov decision processes. *J. Comput. Syst. Sci.*, 84:144–170, 2017.

12. Tomás Brázdil, Vojtech Forejt, Antonín Kucera, and Petr Novotný. Stability in graphs and games. In Desharnais and Jagadeesan [25], pages 10:1–10:14.

13. Thomas Brihaye, Florent Delgrange, Youssouf Oualhadj, and Mickael Randour. Life is random, time is not: Markov decision processes with window objectives. In Wan Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 26-31, 2019, Amsterdam, The Netherlands*, volume 140 of *LIPIcs*, pages 4:1–4:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

14. Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Inf. Comput.*, 254:259–295, 2017.

15. Véronique Bruyère, Quentin Hautem, and Mickael Randour. Window parity games: an alternative approach toward parity games with time bounds. In Domenico Cantone and Giorgio Delzanno, editors, *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016.*, volume 226 of *EPTCS*, pages 135–148, 2016.

16. Véronique Bruyère, Quentin Hautem, and Jean-François Raskin. On the complexity of heterogeneous multidimensional games. In Desharnais and Jagadeesan [25], pages 11:1–11:15.

17. Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 252–263. ACM, 2017.

18. Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. *Inf. Comput.*, 242:25–52, 2015.

19. Krishnendu Chatterjee and Monika Henzinger. Efficient and dynamic algorithms for alternating Büchi games and maximal end-component decomposition. *J. ACM*, 61(3):15:1–15:40, 2014.

20. Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1):1:1–1:27, 2009.

21. Krishnendu Chatterjee, Marcin Jurdzinski, and Thomas A. Henzinger. Quantitative stochastic parity games. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 121–130. SIAM, 2004.

22. Wojciech Czerwinski, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdzinski, Ranko Lazic, and Pawel Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2333–2349. SIAM, 2019.

23. Laure Daviaud, Marcin Jurdzinski, and Ranko Lazic. A pseudo-quasi-polynomial algorithm for mean-payoff parity games. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 325–334. ACM, 2018.

24. Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, volume 10427 of *Lecture Notes in Computer Science*, pages 592–600. Springer, 2017.

25. Josée Desharnais and Radha Jagadeesan, editors. *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, volume 59 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

26. John Fearnley and Marcin Jurdzinski. Reachability in two-clock timed automata is PSPACE-complete. *Inf. Comput.*, 243:26–36, 2015.

27. Nathanaël Fijalkow, Pawel Gawrychowski, and Pierre Ohlmann. The complexity of mean payoff games using universal graphs. *CoRR*, abs/1812.07072, 2018.

28. Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer, 1997.

29. Thomas Gawlitza and Helmut Seidl. Games through nested fixpoints. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2009.

30. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

31. Charles M. Grinstead and J. Laurie Snell. *Introduction to probability*. American Mathematical Society, 1997.

32. Christoph Haase and Stefan Kiefer. The odds of staying on budget. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2015.

33. Arnd Hartmanns, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Multi-cost bounded reachability in MDP. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, volume 10806 of *Lecture Notes in Computer Science*, pages 320–339. Springer, 2018.

34. Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Looking at mean payoff through foggy windows. *Acta Inf.*, 55(8):627–647, 2018.

35. Neil Immerman. Number of quantifiers is better than number of tape cells. *J. Comput. Syst. Sci.*, 22(3):384–406, 1981.

36. Marcin Jurdzinski. Deciding the winner in parity games is in UP ∩ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.

37. Mickael Randour. Automated synthesis of reliable and efficient systems through game theory: A case study. In *Proc. of ECCS 2012*, Springer Proceedings in Complexity XVII, pages 731–738. Springer, 2013.

38. Mickael Randour, Jean-François Raskin, and Ocan Sankur. Variations on the stochastic shortest path problem. In Deepak D'Souza, Akash Lal, and Kim Guldstrand Larsen, editors, *Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*, volume 8931 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2015.

39. Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional Markov decision processes. *Formal Methods in System Design*, 50(2-3):207–248, 2017.

40. Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by Boolean combination of winning conditions. In Sumit Ganguly and Paritosh K. Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, volume 122 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

41. Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.
42. Stephen D. Travers. The complexity of membership problems for circuits over sets of integers. *Theor. Comput. Sci.*, 369(1-3):211–229, 2006.
43. Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. of FOCS*, pages 327–338. IEEE, 1985.