Looking at Mean-Payoff and Total-Payoff through Windows*

Krishnendu Chatterjee[1,†], Laurent Doyen[2], Mickael Randour[3,‡], and Jean-François Raskin[4,§]

[1] IST Austria (Institute of Science and Technology Austria)
[2] LSV - ENS Cachan, France
[3] Computer Science Department, Université de Mons (UMONS), Belgium
[4] Département d'Informatique, Université Libre de Bruxelles (U.L.B.), Belgium

Abstract. We consider two-player games played on weighted directed graphs with mean-payoff and total-payoff objectives, two classical quantitative objectives. While for single-dimensional games the complexity and memory bounds for both objectives coincide, we show that in contrast to multi-dimensional mean-payoff games that are known to be coNP-complete, multi-dimensional total-payoff games are undecidable. We introduce conservative approximations of these objectives, where the payoff is considered over a local finite window sliding along a play, instead of the whole play. For single dimension, we show that (i) if the window size is polynomial, deciding the winner takes polynomial time, and (ii) the existence of a bounded window can be decided in NP $\cap$ coNP, and is at least as hard as solving mean-payoff games. For multiple dimensions, we show that (i) the problem with fixed window size is EXPTIME-complete, and (ii) there is no primitive-recursive algorithm to decide the existence of a bounded window.

1 Introduction

Mean-payoff and total-payoff games. Two-player mean-payoff and total-payoff games are played on finite weighted directed graphs (in which every edge has an integer weight) with two types of vertices: in player-1 vertices, player 1 chooses the successor vertex from the set of outgoing edges; in player-2 vertices, player 2 does likewise. The game results in an infinite path through the graph, called a *play*. The mean-payoff (resp. total-payoff) value of a play is the long-run average (resp. sum) of the edge-weights along the path. While traditionally games on graphs with ω-regular objectives have been studied for system analysis, research efforts have recently focused on quantitative extensions to model resource constraints of embedded systems, such as power consumption, or buffer size [7]. Quantitative games, such as mean-payoff games, are crucial for the formal analysis of resource-constrained reactive systems. For the analysis of systems with multiple resources, multi-dimension games, where edge weights are integer vectors, provide the appropriate framework.

Decision problems. The decision problem for mean-payoff and total-payoff games asks, given a starting vertex, whether player 1 has a strategy that against all strategies of the opponent ensures a play with value at least 0. For both objectives, *memoryless* winning strategies exist for both players (where a memoryless strategy is independent of the past and depends only on the current state) [20,25]. This ensures that the decision problems belong to NP $\cap$ coNP; and they belong to the intriguing class of problems that are in NP $\cap$ coNP but whether they are in P (deterministic polynomial time) are long-standing open questions. The study of mean-payoff games has also been extended to multiple dimensions where the problem is shown to be coNP-complete [40,9]. While for one dimension all the results for mean-payoff and total-payoff coincide, our first contribution shows that quite unexpectedly (in contrast to multi-dimensional mean-payoff games) the multi-dimensional total-payoff games are undecidable.

Window objectives. On the one hand, the complexity of single-dimensional mean-payoff and total-payoff games is a long-standing open problem, and on the other hand, the multi-dimensional problem is undecidable for total-payoff games. In this work, we propose to study variants of these objectives, namely, *bounded window mean-payoff* and *fixed*

* Work partially supported by European project CASSTING (FP7-ICT-601148).
† Author supported by Austrian Science Fund (FWF) Grant No P 23499-N23, FWF NFN Grant No S11407 (RiSE), ERC Start Grant (279307: Graph Games), Microsoft faculty fellowship.
‡ Author supported by F.R.S.-FNRS fellowship.
§ Author supported by ERC Starting Grant (279499: inVEST).

	one-dimension			k-dimension		
	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.
$\underline{\mathsf{MP}}$ / $\overline{\mathsf{MP}}$	NP$\cap$coNP	mem-less		coNP-c. / NP$\cap$coNP	infinite	mem-less
$\underline{\mathsf{TP}}$ / $\overline{\mathsf{TP}}$	NP$\cap$coNP	mem-less		**undec.** (Thm. 1)	-	-
WMP: fixed polynomial window	**P-c.** (Thm. 2)	**mem. req.** $\leq$ **linear**($\lvert S\rvert \cdot l_{\max}$) (Thm. 2)		**PSPACE-h.** (Thm. 4) **EXP-easy** (Thm. 4)	**exponential** (Thm. 4)	
WMP: fixed arbitrary window	**P**($\lvert S\rvert, V, l_{\max}$) (Thm. 2)			**EXP-c.** (Thm. 4)		
WMP: bounded window problem	**NP$\cap$coNP** (Thm. 3)	**mem-less** (Thm. 3)	**infinite** (Thm. 3)	**NPR-h.** (Thm. 5)	-	-

Table 1: Complexity of deciding the winner and memory required, with $|S|$ the number of states of the game (vertices in the graph), V the length of the binary encoding of weights, and $l_{\max}$ the window size. New results in bold (h. for hard and c. for complete).

window mean-payoff objectives. In a bounded window mean-payoff objective instead of the long-run average along the whole play we consider payoffs over a local bounded window sliding along a play, and the objective is that the average weight must be at least zero over every bounded window from some point on. This objective can be seen as a strengthening of the mean-payoff objective (resp. of the total-payoff objective if we require that the window objective is satisfied from the beginning of the play rather than from some point on), i.e., winning for the bounded window mean-payoff objective implies winning for the mean-payoff objective. In the fixed window mean-payoff objective the window length is fixed and given as a parameter. Observe that winning for the fixed window objective implies winning for the bounded window objective.

Attractive features for window objectives. First, they are a strengthening of the mean-payoff objectives and hence provide conservative approximations for mean-payoff objectives. Second, the window variant is very natural to study in system analysis. Mean-payoff objectives require the average to satisfy certain threshold in the long-run (or in the limit of the infinite path), whereas the window objectives require to provide guarantee on the average, not in the limit, but within a bounded time, and thus provide better time guarantee than the mean-payoff objectives. Third, the window parameter provides flexibility, as it can be adjusted specific to applications requirement of strong or weak time guarantee for system behaviors. Finally, we will establish that our variant in the single dimension is more computationally tractable, which makes it an attractive alternative to mean-payoff objectives.

Applicability. In the context of ω-regular objectives, the traditional infinitary notion of liveness has been strengthened to finitary liveness [2], where instead of requiring that good events happen eventually, they are required to happen within a finite time bound. The notion of finitary parity games was introduced and studied in [16], and a polynomial time algorithm for finitary parity games was given in [17], and also studied for pushdown games [12]. The notion of finitary conditions has also been extended to prompt setting where the good events are required to happen as promptly as possible [31]. Our work extends the study of such finite time frames in the setting of quantitative objectives, and our window objectives can be viewed as an extension of finitary conditions for mean-payoff and total-payoff objectives.

With regard to applications, our window variants provide a natural framework to reason about quantitative properties under local finite horizons. To illustrate this point, consider a classical example of application with mean-payoff aspects, as presented by Bohy et al. in the context of synthesis from LTL specifications enriched with mean-payoff objectives [4]. Consider the synthesis of a suitable controller for a computer server having to grant requests to different types of clients. The LTL specification can express that all grants should eventually be granted. Adding quantities and a mean-payoff objective helps in defining priorities between requests and associating costs to the delays between requests and grants, depending of the relative priority of the request. Window objectives are useful for modeling such applications. Indeed, it is clear that in a desired controller, requests should not be placed on hold for an arbitrary long time. Similarly, if we have two types of requests, with different priorities, and we want to ensure guarantees on the mean waiting time per type of request, it seems natural that an adequate balance between the two types should be

observable within reasonable time frames (which can be defined as part of the specification with our new objectives) instead of possible great variations that are allowed by the classical mean-payoff objective.

Our contributions. The main contributions of this work (along with the undecidability of multi-dimensional total-payoff games) are as follows:

1. *Single dimension.* For the single-dimensional case we present an algorithm for the fixed window problem that is polynomial in the size of the game graph times the length of the binary encoding of weights times the size of the fixed window. Thus if the window size is polynomial, we have a polynomial-time algorithm. For the bounded window problem we show that the decision problem is in NP $\cap$ coNP, and at least as hard as solving mean-payoff games. However, winning for mean-payoff games does not imply winning for the bounded window mean-payoff objective, i.e., the winning sets for mean-payoff games and bounded window mean-payoff games do not coincide. Moreover, the structure of winning strategies is also very different, e.g., in mean-payoff games both players have memoryless winning strategies, but in bounded window mean-payoff games we show that player 2 requires infinite memory. We also show that if player 1 wins the bounded window mean-payoff objective, then a window of size $(|S|-1)\cdot(|S|\cdot W+1)$ is sufficient where S is the state space (the set of vertices of the graph), and W is the largest absolute weight value. Finally, we show that (i) a winning strategy for the bounded window mean-payoff objective ensures that the mean-payoff is at least 0 regardless of the strategy of the opponent, and (ii) a strategy that ensures that the mean-payoff is strictly greater than 0 is winning for the bounded window mean-payoff objective.
2. *Multiple dimensions*. For multiple dimensions, we show that the fixed window problem is EXPTIME-complete (both for arbitrary dimensions with weights in $\{-1,0,1\}$ and for two dimensions with arbitrary weights); and if the window size is polynomial, then the problem is PSPACE-hard. For the bounded window problem we show that the problem is non-primitive recursive hard (i.e., there is no primitive recursive algorithm to decide the problem).
3. *Memory requirements.* For all the problems for which we prove decidability we also characterize the memory required by winning strategies.

The relevant results are summarized in Table 1: our results are in bold fonts. In summary, the fixed window problem provides an attractive approximation of the mean-payoff and total-payoff games that we show have better algorithmic complexity. In contrast to the long-standing open problem of mean-payoff games, the one-dimension fixed window problem with polynomial window size can be solved in polynomial time; and in contrast to the undecidability of multi-dimensional total-payoff games, the multi-dimension fixed window problem is EXPTIME-complete.

Related work. This paper extends the results presented in its preceding conference version [10] and gives a full presentation of the technical details. A journal version was published in [11]. Mean-payoff games were first studied by Ehrenfeucht and Mycielski in [20] where it is shown that memoryless winning strategies exist for both players. This entails that the decision problem lies in NP $\cap$ coNP [30,41], and it was later shown to belong to UP $\cap$ coUP [27]. Despite many efforts [26,41,37,33,3], no polynomial-time algorithm for the mean-payoff games problem is known so far. Gurvich, Karzanov, Khachivan and Lebedev [26,30] provided the first (exponential) algorithm for mean-payoff games, later extended by Pisaruk [37]. The first pseudo-polynomial-time algorithm for mean-payoff games was given in [41] and was improved in [5]. Lifshits and Pavlov [33] propose an algorithm which is polynomial in the encoding of weights but exponential in the number of vertices of the graph: it is based on a graph decomposition procedure. Bjorklund and Vorobyov [3] present a *randomized* algorithm which is both subexponential and pseudo-polynomial. Special cases for mean-payoff games can also be solved in polynomial time depending on the weight structure [15], and the algorithmic problem has also been studied for graphs (with one player only) [29,14]. Extension of the worst-case threshold problem - the classical decision problem on mean-payoff games - with guarantees on the expected performance faced to a stochastic adversary has been considered in [6]. While all the above works are for single dimension, multi-dimensional mean-payoff games have been studied in [40,9,18]. One-dimension total-payoff games have been studied in [24] where it is shown that memoryless winning strategies exist for both players and the decision problem is in UP $\cap$ coUP.

2 Preliminaries

We consider two-player turn-based games and denote the two *players* by $\mathcal{P}_1$ and $\mathcal{P}_2$.

Multi-weighted two-player game structures. *Multi-weighted two-player game structures* are weighted graphs $G = (S_1, S_2, E, k, w)$ where (*i*) S_1 and S_2 resp. denote the finite sets of vertices, called *states*, belonging to $\mathcal{P}_1$ and $\mathcal{P}_2$, with $S_1 \cap S_2 = \emptyset$ and $S = S_1 \cup S_2$; (*ii*) $E \subseteq S \times S$ is the set of *edges* such that for all $s \in S$, there exists $s' \in S$ with $(s, s') \in E$; (*iii*) $k \in \mathbb{N}$ is the *dimension* of the weight vectors; and (*iv*) $w \colon E \to \mathbb{Z}^k$ is the multi-weight labeling function. When it is clear from the context that a game G is one-dimensional ($k = 1$), we omit k and write it as $G = (S_1, S_2, E, w)$. The game structure G is *one-player* if $S_2 = \emptyset$. We denote by W the largest absolute weight that appears in the game. For complexity issues, we assume that weights are encoded in binary. Hence we differentiate between pseudo-polynomial algorithms (polynomial in W) and truly polynomial algorithms (polynomial in $V = \lceil \log_2 W \rceil$, the number of bits needed to encode the weights).

A *play* in G from an initial state $s_{\mathsf{init}} \in S$ is an infinite sequence of states $\pi = s_0 s_1 s_2 \dots$ such that $s_0 = s_{\mathsf{init}}$ and $(s_i, s_{i+1}) \in E$ for all $i \geq 0$. The *prefix* up to the n-th state of π is the finite sequence $\pi(n) = s_0 s_1 \dots s_n$. Let $\mathsf{Last}(\pi(n)) = s_n$ denote the last state of $\pi(n)$. A prefix $\pi(n)$ belongs to $\mathcal{P}_i$, $i \in \{1, 2\}$, if $\mathsf{Last}(\pi(n)) \in S_i$. The set of plays of G is denoted by $\mathsf{Plays}(G)$ and the corresponding set of prefixes is denoted by $\mathsf{Prefs}(G)$. The set of prefixes that belong to $\mathcal{P}_i$ is denoted by $\mathsf{Prefs}_i(G)$. The infinite suffix of a play starting in s_n is denoted $\pi(n, \infty)$.

The *total-payoff* of a prefix $\rho = s_0 \dots s_n$ is $\mathsf{TP}(\rho) = \sum_{i=0}^{i=n-1} w((s_i, s_{i+1}))$, and its *mean-payoff* is $\mathsf{MP}(\rho) = \frac{1}{n}\mathsf{TP}(\rho)$. This is naturally extended to plays by considering the componentwise limit behavior (i.e., limit taken on each dimension). The *infimum (resp. supremum) total-payoff* of a play π is $\underline{\mathsf{TP}}(\pi) = \liminf_{n\to\infty} \mathsf{TP}(\pi(n))$ (resp. $\overline{\mathsf{TP}}(\pi) = \limsup_{n\to\infty} \mathsf{TP}(\pi(n))$). The *infimum (resp. supremum) mean-payoff* of π is $\underline{\mathsf{MP}}(\pi) = \liminf_{n\to\infty} \mathsf{MP}(\pi(n))$ (resp. $\overline{\mathsf{MP}}(\pi) = \limsup_{n\to\infty} \mathsf{MP}(\pi(n))$).

Strategies. A *strategy* for $\mathcal{P}_i$, $i \in \{1, 2\}$, in G is a function $\lambda_i \colon \mathsf{Prefs}_i(G) \to S$ such that $(\mathsf{Last}(\rho), \lambda_i(\rho)) \in E$ for all $\rho \in \mathsf{Prefs}_i(G)$. A strategy λ_i for $\mathcal{P}_i$ has *finite-memory* if it can be encoded by a deterministic Moore machine $(M, m_0, \alpha_u, \alpha_n)$ where M is a finite set of states (the memory of the strategy), $m_0 \in M$ is the initial memory state, $\alpha_u \colon M \times S \to M$ is an update function, and $\alpha_n \colon M \times S_i \to S$ is the next-action function. If the game is in $s \in S_i$ and $m \in M$ is the current memory value, then the strategy chooses $s' = \alpha_n(m, s)$ as the next state of the game. When the game leaves a state $s \in S$, the memory is updated to $\alpha_u(m, s)$. Formally, $\langle M, m_0, \alpha_u, \alpha_n \rangle$ defines the strategy λ_i such that $\lambda_i(\rho \cdot s) = \alpha_n(\hat{\alpha}_u(m_0, \rho), s)$ for all $\rho \in S^*$ and $s \in S_i$, where $\hat{\alpha}_u$ extends α_u to sequences of states as expected. A strategy is *memoryless* if $|M| = 1$, i.e., it does not depend on history but only on the current state of the game. We resp. denote by Λ_i, Λ_i^F, and Λ_i^M the sets of general (i.e., possibly infinite-memory), finite-memory, and memoryless strategies for player $\mathcal{P}_i$.

A play π is said to be *consistent* with a strategy λ_i of $\mathcal{P}_i$ if for all $n \geq 0$ such that $\mathsf{Last}(\pi(n)) \in S_i$, we have $\mathsf{Last}(\pi(n+1)) = \lambda_i(\pi(n))$. Given an initial state $s_{\mathsf{init}} \in S$, and two strategies, λ_1 for $\mathcal{P}_1$ and λ_2 for $\mathcal{P}_2$, the unique play from s_{init} consistent with both strategies is the *outcome* of the game, denoted by $\mathsf{Outcome}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2)$.

Attractors. The *attractor* for $\mathcal{P}_1$ of a set $A \subseteq S$ in G is denoted by $\mathsf{Attr}_G^{\mathcal{P}_1}(A)$ and computed as the fixed point of the sequence $\mathsf{Attr}_G^{\mathcal{P}_1,n+1}(A) = \mathsf{Attr}_G^{\mathcal{P}_1,n}(A) \cup \{s \in S_1 \mid \exists (s,t) \in E, t \in \mathsf{Attr}_G^{\mathcal{P}_1,n}(A)\} \cup \{s \in S_2 \mid \forall (s,t) \in E, t \in \mathsf{Attr}_G^{\mathcal{P}_1,n}(A)\}$, with $\mathsf{Attr}_G^{\mathcal{P}_1,0}(A) = A$. The attractor $\mathsf{Attr}_G^{\mathcal{P}_1}(A)$ is exactly the set of states from which $\mathcal{P}_1$ can ensure to reach A no matter what $\mathcal{P}_2$ does. The attractor $\mathsf{Attr}_G^{\mathcal{P}_2}(A)$ for $\mathcal{P}_2$ is defined symmetrically.

Objectives. An *objective* for $\mathcal{P}_1$ in G is a set of plays $\phi \subseteq \mathsf{Plays}(G)$. A play $\pi \in \mathsf{Plays}(G)$ is *winning* for an objective ϕ if $\pi \in \phi$. Given a game G and an initial state $s_{\mathsf{init}} \in S$, a strategy λ_1 of $\mathcal{P}_1$ is winning if $\mathsf{Outcome}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2) \in \phi$ for all strategies λ_2 of $\mathcal{P}_2$. Given a rational threshold vector $v \in \mathbb{Q}^k$, we define the *infimum (resp. supremum) total-payoff (resp. mean-payoff) objectives* as follows:

- $\mathsf{TotalInf}_G(v) = \{\pi \in \mathsf{Plays}(G) \mid \underline{\mathsf{TP}}(\pi) \geq v\}$
- $\mathsf{TotalSup}_G(v) = \{\pi \in \mathsf{Plays}(G) \mid \overline{\mathsf{TP}}(\pi) \geq v\}$
- $\mathsf{MeanInf}_G(v) = \{\pi \in \mathsf{Plays}(G) \mid \underline{\mathsf{MP}}(\pi) \geq v\}$
- $\mathsf{MeanSup}_G(v) = \{\pi \in \mathsf{Plays}(G) \mid \overline{\mathsf{MP}}(\pi) \geq v\}$

Decision problem. Given a game structure G, an initial state $s_{\mathsf{init}} \in S$, and an inf./sup. total-payoff/mean-payoff objective $\phi \subseteq \mathsf{Plays}(G)$, the *threshold problem* asks to decide if $\mathcal{P}_1$ has a winning strategy for this objective. For the mean-payoff, the threshold v can be taken equal to $\{0\}^k$ (where $\{0\}^k$ denotes the k-dimension zero vector) w.l.o.g. as we transform the weight function w to $b \cdot w - a$ for any threshold $\frac{a}{b}$, $a \in \mathbb{Z}^k$, $b \in \mathbb{N}_0 = \mathbb{N} \setminus \{0\}$. For the total-payoff, the same result can be achieved by adding an initial edge of value $-a$ to the game.

3 Mean-Payoff and Total-Payoff Objectives

In this section, we discuss classical mean-payoff and total-payoff objectives. We show that while they are closely related in one dimension, this relation breaks in multiple dimensions. Indeed, we establish that the threshold problem for total-payoff becomes undecidable, both for the infimum and supremum variants.

First, consider one-dimension games. In this case, memoryless strategies exist for both players for both objectives [34,20,23,25] and the sup. and inf. mean-payoff problems coincide (which is not the case for total-payoff). Threshold problems for mean-payoff and total-payoff are closely related as witnessed by Lemma 1 and both have been shown to be in NP$\cap$coNP [41,24].

Lemma 1. *Let $G = (S_1, S_2, E, k, w)$ be a two-player game structure and $s_{\mathsf{init}} \in S$ be an initial state. Let A, B, C and D resp. denote the following assertions.*

A. Player $\mathcal{P}_1$ has a winning strategy for $\mathsf{MeanSup}_G(\{0\}^k)$.
B. Player $\mathcal{P}_1$ has a winning strategy for $\mathsf{MeanInf}_G(\{0\}^k)$.
C. There exists a threshold $v \in \mathbb{Q}^k$ such that $\mathcal{P}_1$ has a winning strategy for $\mathsf{TotalInf}_G(v)$.
D. There exists a threshold $v' \in \mathbb{Q}^k$ such that $\mathcal{P}_1$ has a winning strategy for $\mathsf{TotalSup}_G(v')$.

For games with one-dimension ($k = 1$) weights, all four assertions are equivalent. For games with multi-dimension ($k > 1$) weights, the only implications that hold are: $C \Rightarrow D \Rightarrow A$ and $C \Rightarrow B \Rightarrow A$. All other implications are false.

The statement of Lemma 1 is depicted in Fig. 1: the only implications that extend to the multi-dimension case are depicted by solid arrows.

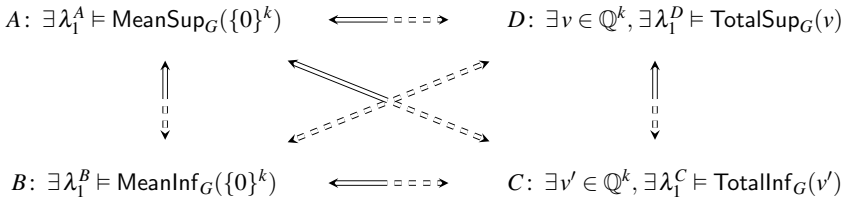

Fig. 1: Equivalence between threshold problems for mean-payoff and total-payoff objectives. Dashed implications are only valid for one-dimension games.

Proof. Specifically, the implications that remain true in multi-weighted games are the trivial ones: satifaction of the infimum version of a given objective trivially implies satisfaction of its supremum version, and satisfaction of infimum (resp. supremum) total-payoff for some finite threshold $v \in \mathbb{Q}^k$ implies satisfaction of infimum (resp. supremum) mean-payoff for threshold $\{0\}^k$ as from some point on, the corresponding sequence of mean-payoff infima (resp. suprema) in all dimensions t, $1 \leq t \leq k$, can be lower-bounded by a sequence of elements of the form $\frac{v(t)}{n}$ with n the length of the prefix, which tends to zero for an infinite play. That is thanks to the sequence of total-payoffs over prefixes being a sequence of integers: it always achieves the value of its limit $v(t)$ instead of only tending to it asymptotically as could a sequence of rationals such as the mean-payoffs. This sums up to $C \Rightarrow D \Rightarrow A$ and $C \Rightarrow B \Rightarrow A$ being true even in the multi-dimension setting.

In the one-dimension case, all assertions are equivalent. First, we have that infimum and supremum mean-payoff problems coincide as memoryless strategies suffice for both players. Thus, we add $A \Rightarrow B$ and $D \Rightarrow B$ by transitivity. Second, consider an optimal strategy for $\mathcal{P}_1$ for the mean-payoff objective of threshold 0. This strategy is such that all cycles formed in the outcome have non-negative effect, otherwise $\mathcal{P}_1$ cannot ensure winning. Thus, the total-payoff over any outcome that is consistent with the same optimal strategy is at all times bounded from below by $-2 \cdot (|S| - 1) \cdot W$ (once for the initial cycle-free prefix, and once for the current cycle being formed). Therefore, we have that $B \Rightarrow C$, and we obtain all other implications by transitive closure.

For multi-weighted games, all dashed implications are false. We specifically consider two of them.

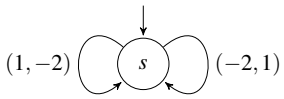

Fig. 2: Satisfaction of supremum TP does not imply satisfaction of infimum MP.

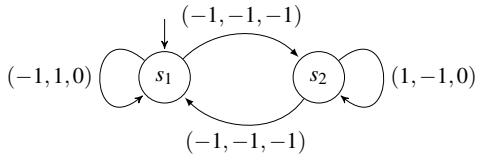

Fig. 3: Satisfaction of infimum MP does not imply satisfaction of supremum TP.

1. To show that implication $D \Rightarrow B$ does not hold, consider the one-player game depicted in Fig. 2. Clearly, any finite vector $v \in \mathbb{Q}^2$ for the supremum total-payoff objective can be achieved by an infinite memory strategy consisting in playing both loops successively for longer and longer periods, each time switching after getting back above the threshold in the considered dimension. However, it is impossible to build any strategy, even with infinite memory, that provides an infimum mean-payoff of $(0,0)$ as the limit mean-payoff would be at best a linear combination of the two cycles values, i.e., strictly less than 0 in at least one dimension in any case.
2. Lastly, implication $B \Rightarrow D$ failure in multi-weighted games can be witnessed in Fig. 3. Clearly, the strategy that plays for n steps in the left cycle, then goes for n steps in the right one, then repeats for $n' > n$ and so on, is a winning strategy for the infimum mean-payoff objective of threshold $(0,0,0)$. Nevertheless, for any strategy of $\mathcal{P}_1$, the outcome is such that either (i) it only switches between cycles a finite number of time, in which case the sum in dimension 1 or 2 will decrease to infinity from some point on, or (ii) it switches infinitely and the sum of weights in dimension 3 decreases to infinity. In both cases, the supremum total-payoff objective is not satisfied for any finite vector $v \in \mathbb{Q}^3$.

All other implications are deduced false as they would otherwise contradict the last two cases by transitivity. □

In multi-dimension games, recent results have shown that the threshold problem for inf. mean-payoff is coNP-complete whereas it is in NP ∩ coNP for sup. mean-payoff [40,39]. In both cases, $\mathcal{P}_1$ needs infinite memory to win, and memoryless strategies suffice for $\mathcal{P}_2$ [9,39]. When restricted to finite-memory strategies, the problem is coNP-complete [9,39] and requires memory at most exponential for $\mathcal{P}_1$ [18].

The case of total-payoff objectives in multi-weighted game structures has never been considered before. Surprisingly, the relation established in Lemma 1 cannot be fully transposed in this context. We show that the threshold problem indeed becomes undecidable for multi-weighted game structures, even for a fixed number of dimensions.

Theorem 1. *The threshold problem for infimum and supremum total-payoff objectives is undecidable in multi-dimension games, for five dimensions.*

Proof. We reduce the halting problem for two-counter machines (2CMs) to the threshold problem for two-player total-payoff games with five dimensions. From a two-counter machine $\mathcal{M}$, we construct a two-player game G with five dimensions and an infimum (equivalently supremum) total-payoff objective such that $\mathcal{P}_1$ wins for threshold $(0,0,0,0,0)$ if and only if the 2CM halts. Counters take values $(v_1, v_2) \in \mathbb{N}^2$ along an execution, and can be incremented or decremented (if positive). A counter can be tested for equality to zero, and the machine can branch accordingly. The halting problem for 2CMs is undecidable [36]. Assume w.l.o.g. that we have a 2CM $\mathcal{M}$ such that if it halts, it halts with the two counters equal to zero. This is w.l.o.g. as it suffices to plug a machine that decreases both counters to zero at the end of the execution of the considered machine. In the game we construct, $\mathcal{P}_1$ has to faithfully simulate the 2CM $\mathcal{M}$. The role of $\mathcal{P}_2$ is to ensure that he does so by retaliating if it is not the case, hence making the outcome losing for the total-payoff objective.

The game is built as follows. The states of G are copies of the control states of $\mathcal{M}$ (plus some special states discussed in the following). Edges represent transitions between these states. The payoff function maps edges to 5-dimensional vectors of the form $(c_1, -c_1, c_2, -c_2, d)$, that is, two dimensions for the first counter C_1, two for the second counter C_2, and one additional dimension. Each increment of counter C_1 (resp. C_2) in $\mathcal{M}$ is implemented in G as a transition of weight $(1,-1,0,0,-1)$ (resp. $(0,0,1,-1,-1)$. For decrements, we have weights respectively $(-1,1,0,0,-1)$ and $(0,0,-1,1,-1)$ for C_1 and C_2. Therefore, the current value of counters (v_1, v_2) along an execution

of the 2CM $\mathcal{M}$ is represented in the game as the current sum of weights, $(v_1, -v_1, v_2, -v_2, -v_3)$, with v_3 the number of steps of the computation. Hence, along a faithful execution, the 1st and 3rd dimensions are always non-negative, while the 2nd, 4th and 5th are always non-positive. The two dimensions per counter are used to enforce faithful simulation of non-negativeness of counters and zero test. The last dimension is decreased by one for every transition, except when the machine halts, from when it is incremented forever (i.e., the play in G goes to an absorbing state with self-loop $(0,0,0,0,1)$). This is used to ensure that a play in G is winning iff $\mathcal{M}$ halts.

We now discuss how this game G ensures faithful simulation of the 2CM $\mathcal{M}$ by $\mathcal{P}_1$.

- *Increment and decrement* of counter values are easily simulated using the first four dimensions.
- *Values of counters may never go below zero*. To ensure this, we allow $\mathcal{P}_2$ to branch after every step of the 2CM simulation to two special states, $s^1_{stop\_neg}$ and $s^2_{stop\_neg}$, which are absorbing and with self-loops of respective weights $(0,1,1,1,1)$ and $(1,1,0,1,1)$. If a negative value is reached on counter C_1 (resp. C_2), $\mathcal{P}_2$ can clearly win the game by branching to state $s^1_{stop\_neg}$ (resp. $s^2_{stop\_neg}$), as the total-payoff in the dimension corresponding to the negative counter will always stay strictly negative. On the contrary, if $\mathcal{P}_2$ decides to go to $s^1_{stop\_neg}$ (resp. $s^2_{stop\_neg}$) when the value of C_1 (resp. C_2) is positive, then $\mathcal{P}_1$ wins the game as this dimension will be positive and the other four will grow boundlessly. So these transitions are only used if $\mathcal{P}_1$ cheats.
- *Zero tests are correctly executed*. In the same spirit, we allow $\mathcal{P}_2$ to branch to two absorbing special states after a zero test, $s^1_{pos\_zero}$ and $s^2_{pos\_zero}$ with self-loops of weights $(1,0,1,1,1)$ and $(1,1,1,0,1)$. Such states are used by $\mathcal{P}_2$ if $\mathcal{P}_1$ cheats on a zero test (i.e., pass the test with a strictly positive counter value). Indeed, if a zero test was passed with the value of counter C_1 (resp. C_2) strictly greater than zero, then the current sum $(v_1, -v_1, v_2, -v_2, v_3)$ is such that $-v_1$ (resp. $-v_2$) is strictly negative. By going to $s^1_{pos\_zero}$ (resp. $s^2_{pos\_zero}$), $\mathcal{P}_2$ ensures that this sum will remain strictly negative in the considered dimension forever and the play is lost for $\mathcal{P}_1$.

Therefore, if $\mathcal{P}_1$ does not faithfully simulate $\mathcal{M}$, he is guaranteed to lose in G. On the other hand, if $\mathcal{P}_2$ stops a faithful simulation, $\mathcal{P}_1$ is guaranteed to win. It remains to argue that he wins iff the machine halts. Indeed, if the machine $\mathcal{M}$ halts, then $\mathcal{P}_1$ simulates its execution faithfully and either he is interrupted and wins, or the simulation ends in an absorbing state with a self-loop of weight $(0,0,0,0,1)$ and he also wins. Indeed, given that this state can only be reached with values of counters equal to zero (by hypothesis on the machine $\mathcal{M}$, without loss of generality), the running sum of weights will reach values $(0,0,0,0,n)$ where n grows to infinity, which ensures satisfaction of the infimum (and thus supremum) total-payoff objective for threshold $(0,0,0,0,0)$. On the opposite, if the 2CM $\mathcal{M}$ does not halt, $\mathcal{P}_1$ has no way to reach the halting state by means of a faithful simulation and the running sum in the fifth dimension always stays negative, thus inducing a losing play for $\mathcal{P}_1$, for both variants of the objective.

Consequently, we have that solving multi-weighted games for either the supremum or the infimum total-payoff objective is undecidable. □

We end this section by noting that in multi-weighted total-payoff games, $\mathcal{P}_1$ may need infinite memory to win, even when all states belong to him ($S_2 = \emptyset$). Consider the game depicted in Fig. 2. As discussed in the proof of Lemma 1, given any threshold vector $v \in \mathbb{Q}^2$, $\mathcal{P}_1$ has a strategy to win the supremum total-payoff objective: it suffices to alternate between the two loops for longer and longer periods, each time waiting to get back above the threshold in the considered dimension before switching. This strategy needs infinite memory and actually, there exists no finite-memory strategy that can achieve a finite threshold vector: the negative amount to compensate grows boundlessly with each alternation, and thus no amount of finite memory can ensure to go above the threshold infinitely often.

4 Window Mean-Payoff Objective

In one dimension, no polynomial algorithm is known for mean-payoff and total-payoff, and in multiple dimensions, total-payoff is undecidable. In this section, we introduce the *window mean-payoff objective*, a conservative approximation in which local deviations from the threshold must be compensated in a parametrized number of steps. We consider a *window*, sliding along a play, within which the compensation must happen. Our approach can be applied both to mean-payoff and total-payoff objectives. Since we consider *finite* windows, both versions coincide for threshold zero. Hence we present our results for mean-payoff.

In Sec. 4.1, we define the objective and discuss its relation with mean-payoff and total-payoff objectives. We then divide our analysis into two subsections: Sec. 4.2 for one-dimension games and Sec. 4.3 for multi-dimension games. Both provide thorough analysis of the *fixed window problem* (the bound on the window size is a parameter) and the *bounded window problem* (existence of a bound is the question). We establish solving algorithms, prove complexity lower bounds, and study the memory requirements of these objectives. In Sec. 4.4, we briefly discuss the extension of our results to a variant of our objective modeling stronger requirements.

4.1 Definition and comparison

Objectives and decision problems. Given a multi-weighted two-player game $G = (S_1, S_2, E, k, w)$ and a rational threshold $v \in \mathbb{Q}^k$, we define the following objectives.

- Given $l_{\max} \in \mathbb{N}_0$, the *good window* objective

$$\mathsf{GW}_G(v, l_{\max}) = \Big\{\pi \in \mathsf{Plays}(G) \mid \forall t,\, 1 \leq t \leq k,\, \exists l \leq l_{\max},\, \frac{1}{l}\sum_{p=0}^{l-1} w\Big(e_\pi(p, p+1)\Big)(t) \geq v(t)\Big\}, \tag{1}$$

  where $e_\pi(p, p+1)$ is the edge $(\mathsf{Last}(\pi(p)), \mathsf{Last}(\pi(p+1)))$, requires that for all dimensions, there exists a window starting in the first position and bounded by $l_{\max}$ over which the mean-payoff is at least equal to the threshold.
- Given $l_{\max} \in \mathbb{N}_0$, the *direct fixed window mean-payoff* objective

$$\mathsf{DirFixWMP}_G(v, l_{\max}) = \Big\{\pi \in \mathsf{Plays}(G) \mid \forall\, j \geq 0,\ \pi(j, \infty) \in \mathsf{GW}_G(v, l_{\max})\Big\} \tag{2}$$

  requires that good windows bounded by $l_{\max}$ exist in all positions along the play.
- The *direct bounded window mean-payoff* objective

$$\mathsf{DirBndWMP}_G(v) = \Big\{\pi \in \mathsf{Plays}(G) \mid \exists\, l_{\max} > 0,\ \pi \in \mathsf{DirFixWMP}_G(v, l_{\max})\Big\} \tag{3}$$

  asks that there exists a bound $l_{\max}$ such that the play satisfies the direct fixed objective.
- Given $l_{\max} \in \mathbb{N}_0$, the *fixed window mean-payoff* objective

$$\mathsf{FixWMP}_G(v, l_{\max}) = \Big\{\pi \in \mathsf{Plays}(G) \mid \exists\, i \geq 0,\ \pi(i, \infty) \in \mathsf{DirFixWMP}_G(v, l_{\max})\Big\} \tag{4}$$

  is the *prefix-independent* version of the direct fixed window objective: it asks for the existence of a suffix of the play satisfying it.
- The *bounded window mean-payoff* objective

$$\mathsf{BndWMP}_G(v) = \Big\{\pi \in \mathsf{Plays}(G) \mid \exists\, l_{\max} > 0,\ \pi \in \mathsf{FixWMP}_G(v, l_{\max})\Big\} \tag{5}$$

  is the *prefix-independent* version of the direct bounded window objective.

For any $v \in \mathbb{Q}^k$ and $l_{\max} \in \mathbb{N}_0$, the following inclusions are true:

$$\mathsf{DirFixWMP}_G(v, l_{\max}) \subseteq \mathsf{FixWMP}_G(v, l_{\max}) \subseteq \mathsf{BndWMP}_G(v), \tag{6}$$

$$\mathsf{DirFixWMP}_G(v, l_{\max}) \subseteq \mathsf{DirBndWMP}_G(v) \subseteq \mathsf{BndWMP}_G(v). \tag{7}$$

Similarly to classical objectives, all objectives can be equivalently expressed for threshold $v = \{0\}^k$ by modifying the weight function. Hence, given any variant of the objective, the associated *decision problem* is to decide the existence of a winning strategy for $\mathcal{P}_1$ for threshold $\{0\}^k$. Lastly, for complexity purposes, we make a difference between *polynomial* (in the size of the game) and *arbitrary* (i.e., non-polynomial) window sizes.

Notice that all those objectives define Borel sets. Hence they are determined by Martin's theorem [35].

Let $\pi = s_0 s_1 s_2 \dots$ be a play. Fix any dimension $t, 1 \leq t \leq k$. The window from position j to j', $0 \leq j < j'$, is *closed* iff there exists j'', $j < j'' \leq j'$ such that the sum of weights in dimension t over the sequence $s_j \dots s_{j''}$ is non-negative. Otherwise the window is *open*. Given a position j' in π, a window is still open in j' iff there exists a position $0 \leq j < j'$ such that the window from j to j' is open. Consider any edge (s_i, s_{i+1}) appearing along π. If the edge is non-negative in dimension t, the window starting in i immediately closes. If not, a window opens that must be closed within $l_{\max}$ steps. Consider the *first* position i' such that this window closes, then we have that all intermediary opened windows also get closed by i', that is, for any i'', $i < i'' \leq i'$, the window starting in i'' is closed before or when reaching position i'. Indeed, the sum of weights over the window from i'' to i' is strictly greater than the sum over the window from i to i', which is non-negative. We call this fact the *inductive property of windows*.

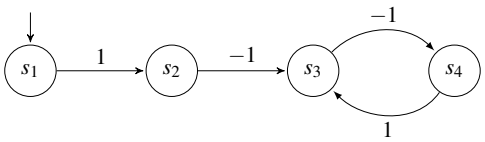

Fig. 4: Fixed window is satisfied for $l_{\max} \geq 2$, whereas even direct bounded window is not.

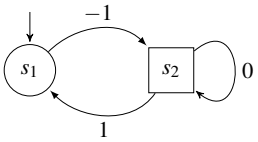

Fig. 5: Mean-payoff is satisfied but none of the window objectives is.

Illustration. Consider the game depicted in Fig. 4. It has a unique outcome, and it is winning for the classical mean-payoff objective of threshold 0, as well as for the infimum (resp. supremum) total-payoff objective of threshold -1 (resp. 0). Consider the fixed window mean-payoff objective for threshold 0. If the size of the window is bounded by 1, the play is losing.[5] However, if the window size is at least 2, the play is winning, as in s_3 we close the window in two steps and in s_4 in one step. Notice that by definition of the objective, it is clear that it is also satisfied for all larger sizes.[6] As the fixed window objective is satisfied for size 2, the bounded window objective is also satisfied. On the other hand, if we restrict the objectives to their direct variants, then none is satisfied, as from s_2, no window, no matter how large it is, gets closed.

Consider the game of Fig. 5. Again, the unique strategy of $\mathcal{P}_1$ satisfies the mean-payoff objective for threshold 0. It also ensures value -1 for the infimum and supremum total-payoffs. Consider the strategy of $\mathcal{P}_2$ that takes the self-loop once on the first visit of s_2, twice on the second, and so on. Clearly, it ensures that windows starting in s_1 stay open for longer and longer numbers of steps (we say that $\mathcal{P}_2$ *delays* the closing of the window), hence making the outcome losing for the bounded window objective (and thus the fixed window objective for any $l_{\max} \in \mathbb{N}_0$). This illustrates the added guarantee (compared to mean-payoff) asked by the window objective: in this case, no upper bound can be given on the time needed for a window to close, i.e., on the time needed to get the local sum back to non-negative. Note that $\mathcal{P}_2$ has to go back to s_1 at some point: otherwise, the prefix-independence of the objectives[7] allows $\mathcal{P}_1$ to wait for $\mathcal{P}_2$ to settle on cycling and win. For the direct variants, $\mathcal{P}_2$ has a simpler winning strategy consisting in looping forever, as enforcing one permanently open window is sufficient.

Relation with classical objectives. We introduce the bounded window objectives as conservative approximations of mean-payoff and total-payoff in one-dimension games. Indeed, in Lemma 2, we show that winning the bounded window (resp. direct bounded window) objective implies winning the mean-payoff (resp. total-payoff) objective while the converse implication is only true if a strictly positive mean-payoff (resp. arbitrary high total-payoff) can be ensured.

Lemma 2. *Given a one-dimension game $G = (S_1, S_2, E, w)$, the following assertions hold.*

(a) If the answer to the bounded window mean-payoff problem is YES*, then the answer to the mean-payoff threshold problem for threshold zero is also* YES.

[5] A window size of one actually requires that all infinitely often visited edges are of non-negative weights.

[6] The existential quantification on the window size l, bounded by $l_{\max}$, is indeed crucial in Eq. (1) to ensure monotonicity with increasing maximal window sizes, a desired behavior of the definition for theoretical properties and intuitive use in specifications.

[7] Fixed and bounded window mean-payoff objectives are prefix-independent: for all $\rho \in \mathsf{Prefs}(G)$, $\pi \in \mathsf{Plays}(G)$, we have that $\rho \cdot \pi$ is winning if and only if π is winning.

(b) If there exists $\varepsilon > 0$ such that the answer to the mean-payoff threshold problem for threshold ε is YES*, then the answer to the bounded window mean-payoff problem is also* YES.
(c) If the answer to the direct bounded window mean-payoff problem is YES*, then the answer to the supremum total-payoff threshold problem for threshold zero is also* YES.
(d) If the answer to the supremum total-payoff threshold problem is YES *for all integer thresholds (i.e., the total-payoff value is ∞), then the answer to the direct bounded window mean-payoff problem is also* YES.

Assertions *(a)* and *(c)* follow from the decomposition of winning plays into bounded windows of non-negative weights. The key idea for assertions *(b)* and *(d)* is that mean-payoff and total-payoff objectives always admit *memoryless* winning strategies, for which the consistent outcomes can be decomposed into *simple cycles* (i.e., with no repeated edge) over which the mean-payoff is at least equal to the threshold and which length is bounded. Hence they correspond to closing windows. Note that strict equivalence with the classical objectives is not verified, as witnessed before (Fig. 5).

Proof. Assertion (a). In the one-dimension case, sup. and inf. mean-payoff problems coincide. Let $\pi \in \mathsf{Plays}(G)$ be such that $\pi \in \mathsf{BndWMP}_G(0)$. There exists $i \geq 0$ such that the suffix of π starting in i can be decomposed into an infinite sequence of bounded segments (i.e., windows) of non-negative weight. Thus, this suffix satisfies the sup. mean-payoff objective as there are infinitely many positions where the total sum from i is non-negative. Since the mean-payoff objective is prefix-independent, the play π is itself winning.

Assertion (b). Consider a memoryless winning strategy of $\mathcal{P}_1$ for the mean-payoff of threshold $\varepsilon > 0$. Only strictly positive simple cycles can be induced by such a strategy. Consider any outcome $\pi = s_0 s_1 s_2 \dots$ consistent with it. We claim that for any position j along this play, there exists a position $j+l$, with $l \leq l_{\max} = (|S|-1)\cdot(1+|S|\cdot W)$, such that the sum of weights over the sequence $\rho = s_j \dots s_{j+l}$ is non-negative. Clearly, if it is the case, then objective $\mathsf{FixWMP}_G(v, l_{\max})$ is satisfied and so is objective $\mathsf{BndWMP}_G(v)$. Consider the cycle decomposition $\mathcal{A}\mathcal{C}_1\mathcal{C}_2\dots\mathcal{C}_n\mathcal{B}$ of this sequence obtained as follows. We push successively $s_0, s_1, \dots$ onto a stack, and whenever we push a state that is already in the stack, a simple cycle is formed that we remove from the stack and append to the cycle decomposition. The sequence ρ is decomposed into an acyclic part $(\mathcal{A}\cup\mathcal{B})$, whose length[8] is at most $(|S|-1)$ and whose total sum is at least $-(|S|-1)\cdot W$, and simple cycles of total sum at least 1 and length at most $|S|$. Given the window size $l_{\max}$, we have at least $(|S|-1)\cdot W$ simple cycles in the cycle decomposition. Hence, the total sum over ρ is at least zero, which proves our point.

Assertion (c). Consider a play $\pi \in \mathsf{DirBndWTP}_G(0)$. Using the same decomposition argument as for assertion *(a)*, we have that the sequence of total sums takes infinitely often values at least equal to zero. Thus the limit of this sequence of moments bounds from below the limit of the sequence of suprema and is at least equal to zero, which shows that the supremum total-payoff objective is also satisfied by play π.

Assertion (d). In one-dimension games, the value of the total-payoff (i.e., the largest threshold for which $\mathcal{P}_1$ has a winning strategy) is ∞ if and only if the value of mean-payoff is strictly positive [24]. Hence, we apply the argument of assertion *(b)*, further noticing that the window open in position j is closed in at most $l_{\max}$ steps for any $j \geq 0$, which is to say that the *direct* objective is satisfied. □

4.2 Games with one dimension

We now study the *fixed window mean-payoff* and the *bounded window mean-payoff* objectives in one-dimension games. For the fixed window problem, we establish an algorithm that runs in time polynomial in the size of the game and in the size of the window and we show that memory is needed for both players. Note that this is in contrast to the mean-payoff objective, where $\mathcal{P}_2$ is memoryless even in the multi-dimension case (cf. Table 1). Moreover, the problem is shown to be P-hard even for polynomial window sizes. For the bounded window problem, we show equivalence with the fixed window problem for size $(|S|-1)\cdot(|S|\cdot W+1)$, i.e., this window size is sufficient to win if possible. The bounded window problem is then shown to be in NP∩coNP and at least as hard as mean-payoff games.

Fixed window: algorithm. Given a game $G = (S_1, S_2, E, w)$ and a window size $l_{\max} \in \mathbb{N}_0$, we present an iterative algorithm FWMP (Alg. 1) to compute the winning states of $\mathcal{P}_1$ for the objective $\mathsf{FixWMP}_G(0, l_{\max})$. Initially, all states

[8] The length of a sequence is the number of *edges* it involves.

are potentially losing for $\mathcal{P}_1$. The algorithm iteratively declares states to be winning, removes them, and continues the computation on the remaining subgame as follows. In every iteration, *i)* DirectFWMP computes the set W_d of states from which $\mathcal{P}_1$ can win the direct fixed window objective; *ii)* it computes the attractor to W_d; and then proceeds to the next iteration on the remaining subgame (the restriction of G to a subset of states $A \subseteq S$ is denoted $G \restriction A$). In every iteration, the states of the computed set W_d are obviously winning for the fixed window objective. Thanks to the prefix-independence of the fixed window objective, the attractor to W_d is also winning. Since $\mathcal{P}_2$ must avoid entering this attractor, $\mathcal{P}_2$ must restrict his choices to stay in the subgame, and hence we iterate on the remaining subgame. Thus states removed over all iterations are winning for $\mathcal{P}_1$. This sequence of steps is essentially the computation of a greatest fixed point. The key argument to establish correctness is as follows: when the algorithm stops, the remaining set of states $\overline{W}$ is such that $\mathcal{P}_2$ can ensure to stay in $\overline{W}$ and falsify the direct fixed window objective by forcing the appearance of one open window larger than $l_{\max}$. Since he stays in $\overline{W}$, he can repeatedly use this strategy to falsify the fixed window objective. Thus the remaining set $\overline{W}$ is winning for $\mathcal{P}_2$, and the correctness of the algorithm follows.

Algorithm 1 $\mathsf{FWMP}(G, l_{\max})$

Require: $G = (S_1, S_2, E, w)$ and $l_{\max} \in \mathbb{N}_0$
Ensure: W is the set of winning states for $\mathcal{P}_1$ for $\mathsf{FixWMP}_G(0, l_{\max})$
  $n := 0$; $W := \emptyset$
  repeat
    $W_d^n := \mathsf{DirectFWMP}(G, l_{\max})$
    $W_{attr}^n := \mathsf{Attr}_G^{\mathcal{P}_1}(W_d^n)$ {attractor for $\mathcal{P}_1$}
    $W := W \cup W_{attr}^n$; $G := G \restriction (S \setminus W)$; $n := n+1$
  until $W = S$ or $W_{attr}^{n-1} = \emptyset$
  return W

Algorithm 2 $\mathsf{DirectFWMP}(G, l_{\max})$

Require: $G = (S_1, S_2, E, w)$ and $l_{\max} \in \mathbb{N}_0$
Ensure: W_d is the set of winning states for $\mathcal{P}_1$ for the objective $\mathsf{DirFixWMP}_G(0, l_{\max})$
  $W_{gw} := \mathsf{GoodWin}(G, l_{\max})$
  if $W_{gw} = S$ or $W_{gw} = \emptyset$ **then**
    $W_d := W_{gw}$
  else
    $W_{safe} := W_{gw} \setminus \mathsf{Attr}_G^{\mathcal{P}_2}(S \setminus W_{gw})$
    $W_d := \mathsf{DirectFWMP}(G \restriction W_{safe}, l_{\max})$
  return W_d

Algorithm 3 $\mathsf{GoodWin}(G, l_{\max})$

Require: $G = (S_1, S_2, E, w)$ and $l_{\max} \in \mathbb{N}_0$
Ensure: W_{gw} is the set of winning states for $\mathsf{GW}_G(0, l_{\max})$
  for all $s \in S$ **do**
    $C_0(s) := 0$
  for all $i \in \{1, \ldots, l_{\max}\}$ **do**
    for all $s \in S_1$ **do**
      $C_i(s) := \max_{(s,s') \in E} \left\{ \max\left\{w((s,s')),\ w((s,s')) + C_{i-1}(s')\right\} \right\}$
    for all $s \in S_2$ **do**
      $C_i(s) := \min_{(s,s') \in E} \left\{ \max\left\{w((s,s')),\ w((s,s')) + C_{i-1}(s')\right\} \right\}$
  return $W_{gw} := \{s \in S \mid C_{l_{\max}}(s) \geq 0\}$

The main idea of algorithm DirectFWMP (Alg. 2) is that to win the direct fixed window objective, $\mathcal{P}_1$ must be able to repeatedly win the good window objective, which consists in ensuring a non-negative sum in at most $l_{\max}$ steps. Thus the algorithm consists in computing a least fixed point. A winning strategy of $\mathcal{P}_1$ in a state s is a strategy that enforces a non-negative sum and, *as soon as the sum turns non-negative* (in some state s'), starts doing the same from s'. It is important to start again immediately as it ensures that all suffixes along the path from s to s' also have a non-negative sum thanks to the inductive property of windows. That is, for any state s'' in between, the window from s'' to s' is closed. The set of states from which $\mathcal{P}_1$ can ensure winning for the good window objective is computed by subroutine GoodWin (Alg. 3). Intuitively, given a state $s \in S$ and a number of steps $i \geq 1$, the value $C_i(s)$ is computed iteratively (from $C_{i-1}(s)$) and represents the best sum that $\mathcal{P}_1$ can ensure from s in exactly i steps:

$$\forall s \in S,\ C_0(s) = 0 \ \wedge\ C_{i \geq 1}(s) = \begin{cases} \max_{(s,s') \in E} \left\{ \max\{w((s,s')),\ w((s,s')) + C_{i-1}(s')\} \right\} & \text{if } s \in S_1, \\ \min_{(s,s') \in E} \left\{ \max\{w((s,s')),\ w((s,s')) + C_{i-1}(s')\} \right\} & \text{if } s \in S_2. \end{cases}$$

Hence, the set of winning states for $\mathcal{P}_1$ is the set of states for which there exists some i, $1 \leq i \leq l_{\max}$ such that $C_i(s) \geq 0$. We state the correctness of GoodWin in Lemma 3.

Lemma 3. *Algorithm* GoodWin *computes the set of winning states of $\mathcal{P}_1$ for the good window objective in time $\mathcal{O}(|E| \cdot l_{\max} \cdot V)$, with $V = \lceil \log_2 W \rceil$, the length of the binary encoding of weights.*

Proof. Let $\mathcal{W}_g \subseteq S$ denote the winning states for $\mathsf{GW}_G(0, l_{\max})$. We prove that (a) $s \in \mathcal{W}_g \Rightarrow s \in \mathsf{GoodWin}(G, l_{\max})$, and (b) $s \in \mathsf{GoodWin}(G, l_{\max}) \Rightarrow s \in \mathcal{W}_g$.

We first consider case (a). From s, there exists a strategy of $\mathcal{P}_1$ that enforces a non-negative sum after l steps, for some l, $1 \leq l \leq l_{\max}$. Hence, the value $C_l(s)$ computed by the algorithm is non-negative and $s \in \mathsf{GoodWin}(G, l_{\max})$.

Case (b). Assume $s \in \mathsf{GoodWin}(G, l_{\max})$. By definition of the algorithm GoodWin, there exists some $l \leq l_{\max}$ such that $C_l(s)$ is positive. Consequently, taking the choice of l edges that achieves the maximum value defines a strategy for $\mathcal{P}_1$ that ensures a positive sum after l steps, hence closing the window started in s. That is, $s \in \mathcal{W}_g$.

It remains to discuss the complexity of GoodWin. Clearly, it takes a number of elementary arithmetic operations which is bounded by $\mathcal{O}(|E| \cdot l_{\max})$ to compute the set W_{gw} as each edge only needs to be visited once at each step i. Each elementary arithmetic operation takes time linear in the number of bits V of the encoding of weights, that is, logarithmic in the largest weight W. Hence, the time complexity of GoodWin is $\mathcal{O}(|E| \cdot l_{\max} \cdot V)$. □

Thanks to the previous lemma, we establish the algorithm solving the direct fixed window objective.

Lemma 4. *Algorithm* DirectFWMP *computes the set of winning states of $\mathcal{P}_1$ for the direct fixed window mean-payoff objective in time $\mathcal{O}(|S| \cdot |E| \cdot l_{\max} \cdot V)$, with $V = \lceil \log_2 W \rceil$, the length of the binary encoding of weights.*

Proof. Let $\mathcal{W}$ be the set of winning states for $\mathsf{DirFixWMP}_G(0, l_{\max})$, i.e.,

$$s \in \mathcal{W} \Leftrightarrow \exists\, \lambda_1 \in \Lambda_1,\ \forall\, \lambda_2 \in \Lambda_2,\ \mathsf{Outcome}_G(s, \lambda_1, \lambda_2) \in \mathsf{DirFixWMP}_G(0, l_{\max}).$$

We first prove (a) $s \in \mathsf{DirectFWMP}(G, l_{\max}) \Rightarrow s \in \mathcal{W}$, and then (b) $s \in \mathcal{W} \Rightarrow s \in \mathsf{DirectFWMP}(G, l_{\max})$. First of all, notice that DirectFWMP exactly computes the set of states W_d such that a non-negative sum is achievable in at most $l_{\max}$ steps, using only states from which a non-negative sum can also be achieved in at most $l_{\max}$ steps (hence the property is defined recursively).

Consider case (a). Let $s \in W_d$. Consider the following strategy of $\mathcal{P}_1$.

1. Play the strategy prescribed by GoodWin until a non-negative sum is reached. This is guaranteed to be the case in at most $l_{\max}$ steps. Let s' be the state that is reached in this manner.
2. By construction of W_d, we have that $s' \in W_d$. Thus, play the strategy prescribed by GoodWin in s'.
3. Continue ad infinitum.

We denote this strategy by λ_1 and claim it is winning for the direct fixed window objective, i.e., $s \in \mathcal{W}$. Indeed, consider any strategy of $\mathcal{P}_2$ and let $\pi = \mathsf{Outcome}_G(s, \lambda_1, \lambda_2)$. We have $\pi = s_1 s_2 \dots s_{m_1} s_{m_1+1} \dots s_{m_2} s_{m_2+1} \dots$ with for all $j \geq 0$, $s_j \in S$ and $s_1 = s_{m_0} = s$, such that all sequences $\rho(n) = s_{m_n} \dots s_{m_{n+1}}$ are of length at most $l_{\max} + 1$ ($l_{\max}$ steps) and such that all strict prefixes of $\rho(n)$ are strictly negative and all suffixes of $\rho(n)$ are positive. Indeed, starting in some state s_{m_n}, the strategy λ_1 keeps a memory of the current sum and tries to reach a non-negative value (using the strategy prescribed by GoodWin). As soon as such a value is reached in a state $s_{m_{n+1}}$, the memory of the current sum kept by the strategy is reset to zero and the process is restarted. That way, for all j, $m_n \leq j < m_{n+1}$, we have that the sum over the sequence from s_j to $s_{m_{n+1}}$ is non-negative, hence all intermediate windows are also closed. Thus, the window property is satisfied everywhere along the play π, starting in $s_1 = s$, which proves that $s \in \mathcal{W}$.

Case (b). Let λ_1 be a winning strategy of $\mathcal{P}_1$ for $\mathsf{DirFixWMP}_G(0, l_{\max})$. For any strategy λ_2 of $\mathcal{P}_2$, the outcome is a play $\pi = s_1 s_2 \dots$ with $s_1 = s$ such that the window property is satisfied from all states. In particular, this implies, that for all s_j, strategy λ_1 enforces a positive sum in at most $l_{\max}$ steps, that is, $s_j \in \mathsf{GoodWin}(G, l_{\max})$. Since it is the case for all states s_j, we have that $\mathcal{P}_1$ has a strategy to ensure a positive sum in at most $l_{\max}$ steps using only states from which this property is ensured. Therefore, we conclude that $s \in W_d$.

Again, the number of calls of this algorithm is at most the number of states $|S|$. Let $\mathbb{C}_{\mathsf{GW}}$ denote the complexity of algorithm GoodWin. Then, the complexity of algorithm DirectFWMP is $\mathcal{O}(|S| \cdot \mathbb{C}_{\mathsf{GW}})$. □

Finally, we prove the correctness of the algorithm for the fixed window problem.

Lemma 5. *Algorithm* FWMP *computes the set of winning states of $\mathcal{P}_1$ for the fixed window mean-payoff objective in time $\mathcal{O}\left(|S|^2 \cdot |E| \cdot l_{\max} \cdot V\right)$, with $V = \lceil \log_2 W \rceil$, the length of the binary encoding of weights.*

Proof. Let $\mathcal{W} \subseteq S$ be the set of states that are winning for $\mathsf{FixWMP}_G(0, l_{\max})$, i.e.,

$$s \in \mathcal{W} \Leftrightarrow \exists \lambda_1 \in \Lambda_1, \forall \lambda_2 \in \Lambda_2, \mathsf{Outcome}_G(s, \lambda_1, \lambda_2) \in \mathsf{FixWMP}_G(0, l_{\max}).$$

Note that since we set the threshold to be 0 (w.l.o.g.), we may ignore the division by the window size l in Eq. (1). We claim that $\mathsf{FWMP}(G, l_{\max}) = \mathcal{W}$. The proof is in two parts: (a) $s \in \mathsf{FWMP}(G, l_{\max}) \Rightarrow s \in \mathcal{W}$, and (b) $s \in \mathcal{W} \Rightarrow s \in \mathsf{FWMP}(G, l_{\max})$.

We begin with (a). Let $(W_d)^{n \geq 0}$ and $(W_{attr})^{n \geq 0}$ be the finite sequences of sets computed by the algorithm. We have that $\mathsf{FWMP}(G, l_{\max}) = \bigcup_{n \geq 0} W^n_{attr}$. For any n, n' such that $n \neq n'$, we have that $W^n_{attr} \cap W^{n'}_{attr} = \emptyset$ and $W^n_d \cap W^{n'}_d = \emptyset$. Moreover, for all $n \geq 0$, $W^n_d \subseteq W^n_{attr}$. Let $s \in \mathsf{FWMP}(G, l_{\max})$. There exists a unique $n \geq 0$ such that $s \in W^n_{attr}$. By construction, from s, $\mathcal{P}_1$ has a strategy to reach and stay in $W^n_d \cup W^{n-1}_{attr} \cup W^{n-2}_{attr} \cup \ldots W^0_{attr}$ and thus s is winning in the subgame $G \restriction (S \setminus W^{n-1}_{attr})$. However, $\mathcal{P}_2$ still has the possibility to leave W^n_d and reach the set $W^{n-1}_{attr} \cup W^{n-2}_{attr} \cup \ldots W^0_{attr}$. Since the sequence is finite and $\mathcal{P}_2$ cannot leave W^0_d, we have that at some point, any outcome is trapped in some set W^m_d, $0 \leq m \leq n$, in which $\mathcal{P}_1$ wins the direct fixed window objective. Let x be the length of the finite prefix outside the set W^m_d. The outcome satisfies the fixed window mean-payoff objective for $i = x$. Therefore, we have that $s \in \mathcal{W}$.

Now consider (b). Let $s \in \mathcal{W}$ be a winning state for $\mathsf{FixWMP}_G(0, l_{\max})$. We claim that $s \in \mathsf{FWMP}(G, l_{\max})$. Suppose it is not the case and consider the sequences $(W_d)^{n \geq 0}$ and $(W_{attr})^{n \geq 0}$ as before. We have that for all $n \geq 0$, $s \notin W^n_{attr}$. In particular, $\mathcal{P}_2$ can force staying in $S_{trap} = S \setminus \bigcup_{n \geq 0} W^n_{attr}$ when starting in s. Since the algorithm has stopped, we have that $\mathsf{DirectFWMP}(G \restriction S_{trap}, l_{\max}) = \emptyset$. As algorithm DirectFWMP is correct, from all states of S_{trap}, $\mathcal{P}_2$ has a strategy to spoil the direct fixed window game, i.e., $\mathcal{P}_2$ can force a sequence of states such that there exists a position j along it for which the window starting in j stays open for at least $(l_{\max} + 1)$ steps, and such that this sequence remains in S_{trap}. Therefore, $\mathcal{P}_2$ can force staying in S_{trap} and seeing infinitely often such sequences, hence $\mathcal{P}_1$ is losing for the fixed window mean-payoff objective, which contradicts the fact that $s \in \mathcal{W}$.

Finally, consider the complexity of the recursive algorithm FWMP. Notice that at least one state is declared winning at each iteration. The number of calls is thus at most the number of states $|S|$. Computing the attractor is linear in the number of edges $|E| \leq |S|^2$. The overall complexity is thus $\mathcal{O}(|S| \cdot (|E| + \mathbb{C}_{\mathsf{DW}}))$, where $\mathbb{C}_{\mathsf{DW}}$ is the complexity of the DirectFWMP algorithm. □

Fixed window: lower bounds. Thanks to the correctness of algorithm FWMP, we also deduce linear upper bounds (in $|S| \cdot l_{\max}$) on the memory needed for both players (Lemma 6). Indeed, let $s \in S$ be a winning state for $\mathcal{P}_1$. A winning strategy λ_1 for $\mathcal{P}_1$ is to (a) reach the set of states W^n_d that are winning for the direct fixed window objective in the subgame restricted to states $W^n_d \setminus W^{n-1}_{attr}$, then (b) repeatedly play the strategy prescribed by GoodWin in this subgame (i.e., enforce a non-negative sum in less than $l_{\max}$ steps, see proof of Lemma 4). If $\mathcal{P}_2$ leaves for a lower subgame restricted to $W^{n'}_{attr}$, $n' < n$, the strategy is to start again part (a) in this subgame. Part (a) is memoryless as it uses a classical attractor strategy. Part (b) requires to consider, for each state s' in the set computed by DirectFWMP, a number of memory states which is bounded by $l_{\max}$, as the only memory needed is to select the corresponding successor state that will maximize the $C_l(s')$ value, for all possible values of l, the number of steps remaining to close a window. Similarly, $\mathcal{P}_2$ needs to be able to prevent the closing of a window repeatedly, and therefore also possibly needs $l_{\max}$ memory states for each state of the game.

To illustrate that memory is needed by both players, consider the following examples. First, consider a game where all states belong to $\mathcal{P}_1$ and such that the play starts in a central state s and in s, there are three outgoing edges, towards three simple cycles $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. All other states have only one outgoing edge. Cycle $\mathcal{C}_1$ is composed of six edges of successive weights $3, 3, 5, -1, -1$ and -5. Cycle $\mathcal{C}_2$ is $7, -1$ and -9. Cycle $\mathcal{C}_3$ is $5, 5$ and -11. The objective is $\mathsf{FixWMP}_G(0, l_{\max} = 4)$. Clearly, from some point on, a winning strategy of $\mathcal{P}_1$ has to infinitely alternate between cycles in the following way: $(\mathcal{C}_1\mathcal{C}_2\mathcal{C}_3)^\omega$. Any other alternation leads to a bad window appearing infinitely often: hence, the decision of $\mathcal{P}_1$ in s depends on the remaining number of steps to ensure a good window. Second, consider a similar game but with all states belonging to $\mathcal{P}_2$. Again, the initial state is central and there are two cycles $\mathcal{C}_1$ and $\mathcal{C}_2$ such that

$\mathcal{C}_1$ is 1 followed by -1, and $\mathcal{C}_2$ is $-1,-1$ and 2. The objective is $\mathsf{FixWMP}_G(0, l_{\max}=3)$. If $\mathcal{P}_2$ is memoryless, both possible strategies induce a winning play for $\mathcal{P}_1$. On the other hand, if $\mathcal{P}_2$ is allowed to alternate, he can choose the play $(\mathcal{C}_1\mathcal{C}_2)^\omega$ which will be losing for $\mathcal{P}_1$ as the window $-1,-1,-1$ will appear infinitely often.

Lemma 6. *In one-dimension games with a fixed window mean-payoff objective, memory is needed by both players and linear memory in the number of states times the window size is sufficient.*

Through Lemma 5, we have shown that the fixed window problem admits a polynomial (in $|S|$, V and $l_{\max}$) algorithm. In Lemma 7, we prove that even for window size $l_{\max}=1$ and weights $\{-1,1\}$, the problem is P-hard. This is via a reduction from reachability games. By making the target states absorbing with a self-loop of weight 1, and giving weight -1 on all other edges, we obtain the reduction, as reaching a target state is now the only way to ensure that windows close.

Lemma 7. *In two-player one-dimension games, the fixed window mean-payoff problem is P-hard, even for $l_{\max}=1$ and weights $\{-1,1\}$.*

Proof. Let $G_r=(S_1,S_2,E)$ be an unweighted game with a reachability objective asking to visit (at least once) a state of the set $R\subseteq S$. We build the game $G=(S_1,S_2,E',w)$ by (a) making the target states absorbing with a self-loop of weight 1, i.e., for all $s\in R$, we have $(s,s)\in E'$ and $w((s,s))=1$, and (b) putting weight -1 on all other edges, i.e., for all edge $(s,t)\in E$ such that $s\notin R$, we have $(s,t)\in E'$ and $w((s,s))=-1$. We claim that $\mathcal{P}_1$ has a winning strategy in G_r from a state $s\in S$ if and only if he has a winning strategy for the objective $\mathsf{FixWMP}_G(0,l_{\max}=1)$ in G from $s\in S$. Indeed, it is clear that any outcome that never reaches the target set is such that all windows stay indefinitely open, and conversely, an outcome that reaches this set after n steps is winning for the fixed window objective with $i=n$. Since deciding the winner in reachability games is P-complete, this concludes our proof. □

Fixed window: summary. We sum up the complexity analysis of the fixed window problem in Theorem 2.

Theorem 2. *In two-player one-dimension games, (a) the fixed arbitrary window mean-payoff problem is decidable in time $\mathcal{O}\left(|S|^2\cdot|E|\cdot l_{\max}\cdot V\right)$, with $V=\lceil\log_2 W\rceil$, the length of the binary encoding of weights, and (b) the fixed polynomial window mean-payoff problem is P-complete. In general, both players require memory, and memory of size linear in $|S|\cdot l_{\max}$ is sufficient.*

Bounded window: algorithm. In the following, we focus on the bounded window mean-payoff problem for two-player one-dimension games. We start with two technical lemmas related to the classical supremum total-payoff threshold problem. Using these lemmas, we establish an algorithm to solve the bounded window problem. This algorithm uses a polynomial number of calls to an oracle solving the total-payoff threshold problem, hence proving that the bounded window problem is in NP∩coNP [24]. As a corollary, we get an interesting bound on the window size needed to win the fixed window problem if possible.

The first technical lemma (Lemma 8) states that if $\mathcal{P}_1$ has a strategy to win the supremum total-payoff objective from some state s_{init}, then he can force a non-negative sum from this state in at most $(|S|-1)\cdot(|S|\cdot W+1)$ steps, i.e., he wins the good window objective for this window size.

Lemma 8. *Let $G=(S_1,S_2,E,w)$ be a two-player one-dimension game. If $\mathcal{P}_1$ has a strategy to win for objective $\mathsf{TotalSup}_G(0)$ from initial state $s_{\mathsf{init}}\in S$, then $\mathcal{P}_1$ also has a strategy to win for the good window objective $\mathsf{GW}_G(0,l_{\max})$ from s_{init} for $l_{\max}=(|S|-1)\cdot(|S|\cdot W+1)$.*

This result is obtained by considering a memoryless winning strategy of $\mathcal{P}_1$ for the total-payoff and the decomposition in simple cycles of any consistent outcome where (a) either simple cycles are strictly positive, or (b) they are of value zero but preceded by a non-negative prefix.

Proof. Let $\lambda_1\in\Lambda_1^M$ be a memoryless winning strategy of $\mathcal{P}_1$ for $\mathsf{TotalSup}_G(0)$. Our claim is that for all possible outcome π consistent with λ_1 starting in the initial state s_{init}, there exists a prefix ρ of π of size at most $l_{\max}$ such that the total sum of weights over ρ is non-negative. Let π be any outcome consistent with λ_1 and ρ_1 its prefix of length $(|S|-1)\cdot(|S|\cdot W+1)$. Consider the cycle decomposition (see the proof of Lemma 2) of ρ_1: $\mathcal{A},\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_m,\mathcal{B}$, with $\mathcal{A}$ the prefix before the first cycle and $\mathcal{B}$ the suffix after the last cycle in ρ_1. The total length of the acyclic part is $|\mathcal{A}|+|\mathcal{B}|<|S|-1$. We claim that there exists a prefix ρ of ρ_1 such that the total sum of weights over ρ is non-negative. Consider the following arguments:

1. No cycle $\mathcal{C}$ in $\{\mathcal{C}_1,\dots,\mathcal{C}_m\}$ can be strictly negative. Otherwise, since λ_1 is memoryless, $\mathcal{P}_2$ could force cycling in such a cycle forever and the play would be losing for the supremum total-payoff objective, which contradicts λ_1 being a winning strategy.
2. Assume that there exists a cycle $\mathcal{C}$ in $\{\mathcal{C}_1,\dots,\mathcal{C}_m\}$ such that the sum of weights over this cycle is zero. We define the *high point* of a cycle as the first state where the sum from the start of the cycle takes its highest value. Then, the prefix ρ of ρ_1 up to this high point is non-negative and we are done. Indeed, assume it is not the case. Then, the running sum over the outcome π is strictly negative when reaching the high point, and stays strictly negative in all positions along the cycle $\mathcal{C}$, by definition of the high point. Therefore, $\mathcal{P}_2$ can force cycling forever in $\mathcal{C}$ since λ_1 is memoryless and the outcome becomes losing for the total-payoff objective.
3. So assume there are only strictly positive cycles in the cycle decomposition of ρ_1, that is, they all have a total sum of value at least 1. The total sum over $\mathcal{C}_1,\dots,\mathcal{C}_m$ is at least equal to m. Since each cycle is of length at most $|S|$ and $\mathcal{A}\cup\mathcal{B}$ is of length at most $|S|-1$, we have that the number of cycles m in the cycle decomposition of ρ_1 is at least $((|S|-1)\cdot(|S|\cdot W+1)-(|S|-1))\,/\,|S| = (|S|-1)\cdot W$. Given that the total sum over prefix $\mathcal{A}$ is at least $-(|S|-1)\cdot W$, we obtain that $\rho = \mathcal{A}\mathcal{C}_1\mathcal{C}_2\dots\mathcal{C}_m$ is the desired prefix with a non-negative total sum, and its length is bounded by $(|S|-1)\cdot(|S|\cdot W+1)$.

This concludes our proof. □

The second technical lemma (Lemma 9) shows that if $\mathcal{P}_2$ has a strategy to ensure that the supremum total-payoff from some state s_{init} is strictly negative, then he has a memoryless strategy to do so and any outcome π starting in s_{init} and consistent with this strategy is such that the direct bounded window mean-payoff objective is not satisfied.

Lemma 9. *Let $G=(S_1,S_2,E,w)$ be a two-player one-dimension game. If $\mathcal{P}_2$ has a spoiling strategy for objective $\mathsf{TotalSup}_G(0)$ from initial state $s_{\mathsf{init}}\in S$, then $\mathcal{P}_2$ has a strategy $\lambda_2\in\Lambda_2^M$ to ensure that for all possible outcome $\pi=s_0s_1\dots$ consistent with λ_2 starting in $s_0=s_{\mathsf{init}}$, there exists a position $i\geq 0$ such that for all window sizes $l\geq 1$, the total sum of weights on the window from s_i to s_{i+l} is strictly negative.*

Proof. By contradiction. Let $\lambda_2\in\Lambda_2^M$ be a memoryless spoiling strategy for objective $\mathsf{TotalSup}_G(0)$ from $s_{\mathsf{init}}\in S$. Let π be a consistent outcome and assume that it does not respect the lemma, i.e., for all positions $i\geq 0$, there exists a window size $l\geq 1$ such that the window from s_i to s_{i+l} is non-negative. Then the play π can be decomposed as a sequence of finite windows of non-negative weights. Hence, the total sum from $s_0=s_{\mathsf{init}}$ takes infinitely often values at least equal to zero and the limit of its suprema is non-negative. This is in contradiction to λ_2 being a winning strategy for $\mathcal{P}_2$. □

Thanks to Lemma 8 and Lemma 9, we are now able to establish an algorithm (Alg. 4) to solve the bounded window mean-payoff problem on two-player one-dimension games, and to deduce NP∩coNP-membership of the problem. Lemma 10 states its correctness.

Algorithm BoundedProblem (Alg. 4) computes via a subroutine UnbOpenWindow the set of states from which $\mathcal{P}_2$ can force the visit of a position such that the window opening in this position never closes. Clearly, to prevent $\mathcal{P}_1$ from winning the bounded window problem, $\mathcal{P}_2$ must be able to do so repeatedly as the prefix-independence of the objective otherwise gives the possibility to wait that all such bad positions are encountered before taking the windows into account. Therefore, the states that are not in $\mathsf{UnbOpenWindow}(G)$, as well as their attractor, are winning for $\mathcal{P}_1$. Since the choices of $\mathcal{P}_2$ are reduced because of the attractor of $\mathcal{P}_1$ being declared winning, we compute in several steps, adding new states to the set of winning states for $\mathcal{P}_1$ up to stabilization.

Now consider the subroutine UnbOpenWindow (Alg. 5). Its correctness is based on Lemma 9. Indeed, it computes the set of states from which $\mathcal{P}_2$ can force a position for which the window never closes. To do so, it suffices to compute the attractor for $\mathcal{P}_2$ of the set of states from which $\mathcal{P}_2$ can enforce a strictly negative supremum total-payoff. Routine NegSupTP denotes a call to an oracle solving the total-payoff problem, which is known to belong to NP∩coNP [24]. Precisely,

$$\mathsf{NegSupTP}(G)=\left\{s\in S\mid \exists\lambda_2\in\Lambda_2,\forall\lambda_1\in\Lambda_1,\forall\pi\in\mathsf{Outcome}_G(s,\lambda_1,\lambda_2),\,\overline{\mathsf{TP}}(\pi)<0\right\}.$$

Again, we compute the fixed point of the sequence as the choices of $\mathcal{P}_1$ are reduced at each iteration.

Algorithm 4 BoundedProblem(G)

Require: Game $G = (S_1, S_2, E, w)$
Ensure: W_{bp} is the set of winning states for $\mathcal{P}_1$ for the bounded window mean-payoff problem
  $W_{bp} := \emptyset$
  $L :=$ UnbOpenWindow(G)
  while $L \neq S \setminus W_{bp}$ **do**
    $W_{bp} := \mathsf{Attr}_G^{\mathcal{P}_1}(S \setminus L)$
    $L :=$ UnbOpenWindow$\Big(G \restriction (S \setminus W_{bp})\Big)$
  return W_{bp}

Algorithm 5 UnbOpenWindow(G)

Require: Game $G = (S_1, S_2, E, w)$
Ensure: L is the set of states from which $\mathcal{P}_2$ can force a position for which the window never closes
  $p := 0$; $L_0 := \emptyset$
  repeat
    $L_{p+1} := L_p \cup \mathsf{Attr}^{\mathcal{P}_2}_{G \restriction (S \setminus L_p)}\Big(\mathsf{NegSupTP}\big(G \restriction (S \setminus L_p)\big)\Big)$
    $p := p + 1$
  until $L_p = L_{p-1}$
  return $L := L_p$

The main idea of the correctness proof is that from all states in $\overline{W_{bp}}$, $\mathcal{P}_2$ has an infinite-memory winning strategy which is played in rounds, and in round n ensures an open window of size at least n by playing the total-payoff strategy of $\mathcal{P}_2$ for at most $n \cdot |S|$ steps, and then proceeds to round $(n+1)$ to ensure an open window of size $(n+1)$, and so on. Hence, windows stay open for arbitrary large periods and the bounded window objective is falsified.

Lemma 10. *Given a two-player one-dimension game $G = (S_1, S_2, E, w)$, the algorithm* BoundedProblem *computes the set of winning states for $\mathcal{P}_1$ for the bounded window mean-payoff objective of threshold 0 in time $\mathcal{O}(|S|^2 \cdot (|E| + \mathbb{C}))$, where $\mathbb{C}$ is the complexity of algorithm* NegSupTP*, i.e., the complexity of computing the set of winning states in a two-player one-dimension supremum total-payoff game. Thus, algorithm* BoundedProblem *is in NP $\cap$ coNP.*

Proof. It suffices to show that for all states in $W_{bp} =$ BoundedProblem(G), there exists a winning strategy of $\mathcal{P}_1$, whereas for all states in $S \setminus W_{bp}$, there exists one of $\mathcal{P}_2$.

Consider a state $s \in W_{bp}$. Consider $(L^m)_{0 \leq m \leq n}$, the finite sequence of sets L that are computed by BoundedProblem, with $L_0 =$ UnbOpenWindow(G); and $(W_{bp}^m)_{0 \leq m \leq n}$, the corresponding finite sequence of sets W_{bp} where $W_{bp}^0 = \emptyset$ is empty and $W_{bp}^n = W_{bp}$ is the returned set of winning states. For all m', m, $0 \leq m' < m \leq n$, we have that $W_{bp}^m \supset W_{bp}^{m'}$ and $L^m \subset L^{m'}$. By construction, there exists m, $1 \leq m \leq n$ such that $s \in W_{bp}^m = \mathsf{Attr}_G^{\mathcal{P}_1}(S \setminus L^{m-1})$. In the subgame $G \restriction ((S \setminus L^{m-1}) \setminus W_{bp}^{m-1})$, $\mathcal{P}_1$ has a memoryless [25] winning strategy for the supremum total-payoff objective. Hence, consider the strategy λ_1 of $\mathcal{P}_1$ which is to reach the set $(S \setminus L^{m-1})$ (in at most $|S|$ steps) and then play the memoryless total-payoff strategy in the subgame. It is possible for $\mathcal{P}_2$ to force leaving this subgame for a lower subset $W_{bp}^{m'} \subset W_{bp}^m$ with $m' < m$ but since the sequence is finite, any outcome is ultimately trapped in some subgame $G \restriction ((S \setminus L^{m''}) \setminus W_{bp}^{m''})$. Therefore, repeating the strategy λ_1 in each subgame ensures that after a finite number of steps (and hence a finite number of positions for which windows never close), a bottom subgame $G \restriction ((S \setminus L^{m''}) \setminus W_{bp}^{m''})$ is reached and, by Lemma 8, strategy λ_1 ensures satisfaction of the good window objective for $l_{\max} = (|S| - 1) \cdot (|S| \cdot W + 1)$ in this subgame. Moreover, since this strategy never visits states out of the bottom subgame, it ensures an inductive window from every state, regardless of the past. Hence, all intermediate windows are also closed and this strategy is winning for $\mathsf{FixWMP}_G(0, l_{\max}) \subseteq \mathsf{BndWMP}_G(0)$ from the initial state s. The states that are only visited finitely often before reaching the bottom subgame have no consequence thanks to the prefix-independence of the bounded window mean-payoff objective.

As for $\mathcal{P}_2$, consider a state $s \in S \setminus W_{bp}$. Consider $(L_p)_{0 \leq p \leq q}$, the finite sequence of sets L that are computed in the last call to UnbOpenWindow by BoundedProblem, with $L_0 = \emptyset$. We define the sequences $(N_p)_{1 \leq p \leq q}$ and $(A_p)_{1 \leq p \leq q}$ as $N_p = \mathsf{NegSupTP}(G \restriction (S \setminus L_{p-1}))$ and $A_p = L_p \setminus L_{p-1} = \mathsf{Attr}^{\mathcal{P}_2}_{G \restriction (S \setminus L_{p-1})}(N_p)$. We have that $s \in L_p$ for some p between 1 and q. An infinite memory winning strategy for $\mathcal{P}_2$ is played in rounds. In round n, $\mathcal{P}_2$ acts as follows. (a) If the current state is in A_p, play the attractor to N_p and then play the optimal strategy for the supremum total-payoff in N_p to ensure that no window will have a non-negative sum for n steps. (b) $\mathcal{P}_1$ can leave the set A_p for some lower set $A_{p'}$, $1 \leq p' < p$. If so, play the attractor to $N_{p'}$ and continue. Ultimately, any outcome is trapped in some set $N_{p''} \setminus A_{p''-1}$, with $1 \leq p'' \leq q$ and $A_0 = \emptyset$, as in N_1, $\mathcal{P}_1$ cannot leave. There $\mathcal{P}_1$ cannot prevent the window being strictly negative for

n steps. When such a window has been enforced for n steps, move to round $n+1$ and start again. This strategy ensures that the bounded window problem is not satisfied as, infinitely often, windows stay open for arbitrary large periods along any outcome.

Finally, we discuss the complexity of algorithm BoundedProblem. Let $\mathbb{C}$ be the complexity of routine NegSupTP, that is, the complexity of solving a one-dimension supremum total-payoff game. The total complexity of subalgorithm UnbOpenWindow is $\mathcal{O}(|S| \cdot (|E| + \mathbb{C}))$ as the sequence of computations is of length at most $|S|$ and each computation takes time $\mathcal{O}(|E| + \mathbb{C})$. The overall complexity of BoundedProblem is thus $\mathcal{O}(\mathbb{C} + |S| \cdot (|E| + |S| \cdot (|E| + \mathbb{C}))) = \mathcal{O}(|S|^2 \cdot (|E| + \mathbb{C}))$. □

An interesting corollary of Lemma 8 and Lemma 10 is that the sets of winning states coincide for objectives $\mathsf{FixWMP}_G(0, l_{\max} = (|S|-1) \cdot (|S| \cdot W + 1))$ and $\mathsf{BndWMP}_G(0)$, therefore proving NP ∩ coNP-membership for the subset of fixed window problems with window size at least $l_{\max}$ (hence an algorithm independent of the window size whereas Lemma 4 gives an algorithm which is polynomial in the window size).

Corollary 1. *In two-player one-dimension games, the fixed window mean-payoff problem is in NP ∩ coNP for window size at least equal to* $(|S|-1) \cdot (|S| \cdot W + 1)$.

Bounded window: lower bounds. Algorithm BoundedProblem (Lemma 10) provides memoryless winning strategies for $\mathcal{P}_1$ (attractor + memoryless strategy for total-payoff) and infinite-memory winning strategies for $\mathcal{P}_2$ (delaying the closing of windows for increasing number of steps each round) in one-dimension bounded window mean-payoff games. Lemma 11 states that infinite memory is necessary for $\mathcal{P}_2$, as discussed in Section 4.1: $\mathcal{P}_2$ cannot use the zero cycle forever, but he must cycle long enough to defeat any finite window. Hence, its strategy needs to cycle for longer and longer, which requires infinite memory.

Lemma 11. *In one-dimension games with a bounded window mean-payoff objective, (a) memoryless strategies suffice for $\mathcal{P}_1$, and (b) infinite-memory strategies are needed for $\mathcal{P}_2$ in general.*

In Lemma 14, we give a polynomial reduction from mean-payoff games to bounded window mean-payoff games, therefore showing that a polynomial algorithm for the bounded window problem would solve the long-standing question of the P-membership of the mean-payoff threshold problem. The proof relies on technical lemmas providing intermediary reductions. First, we prove that given a game G, deciding if $\mathcal{P}_1$ has a strategy to ensure a non-negative mean-payoff can be reduced to deciding if $\mathcal{P}_1$ has a strategy to ensure a strictly positive mean-payoff when weights are shifted positively by a sufficiently small ε (Lemma 12). Second, we apply Lemma 2 on the shifted game to prove that winning this objective implies winning the bounded window problem. This gives one direction of the reduction. For the other one, we show that given a game G, if $\mathcal{P}_1$ has a strategy to win the bounded window problem when weights are shifted positively by a sufficiently small ε, he has one to win the mean-payoff threshold problem in G.

We define the following notation: given a two-player one-dimension game $G = (S_1, S_2, E, w)$ and $\varepsilon \in \mathbb{Q}$, let $G_{+\varepsilon} = (S_1, S_2, E, w_{+\varepsilon})$ be the game obtained by shifting all weights by ε, that is, for all $e \in E$, $w_{+\varepsilon}(e) = w(e) + \varepsilon$.[9]

Lemma 12. *For all one-dimension game $G = (S_1, S_2, E, w)$ with integer weights, for all ε, $0 < \varepsilon < 1/|S|$, for all initial state $s \in S$, $\mathcal{P}_1$ has a strategy to ensure a non-negative mean-payoff in G if and only if $\mathcal{P}_1$ has a strategy to ensure a strictly positive mean-payoff in $G_{+\varepsilon}$.*

Proof. Consider a memoryless winning strategy of $\mathcal{P}_1$ in G from initial state $s \in S$. All simple cycles in consistent outcomes have a sum of weights at least equal to zero. Hence, the corresponding outcome in $G_{+\varepsilon}$ is such that all simple cycles of length n have sums at least equal to $n \cdot \varepsilon > 0$, which proves that the strategy is also winning in $G_{+\varepsilon}$.

Consider a memoryless winning strategy of $\mathcal{P}_2$ in G from initial state $s \in S$. All simple cycles in consistent outcomes have a strictly negative sum of weights, that is the sum is at most equal to -1. Hence, the corresponding outcome in $G_{+\varepsilon}$ is such that all simple cycles of length n have sums at most equal to $-1 + n \cdot \varepsilon$. Since $n \leq |S|$ and $\varepsilon < 1/|S|$, we have that the sum is strictly negative, which proves that the strategy is also winning in $G_{+\varepsilon}$.

By determinacy of mean-payoff games, we obtain the claim. □

[9] Note that $w_{+\varepsilon}$ can be transformed into an integer valued function without changing the answers to the considered decision problems.

Lemma 13. *For all one-dimension game $G = (S_1, S_2, E, w)$ with integer weights, for all ε, $0 < \varepsilon < 1/|S|$, for all initial state $s \in S$, if $\mathcal{P}_1$ has a strategy to win the bounded window mean-payoff problem in $G_{+\varepsilon}$, then $\mathcal{P}_1$ has a strategy to win the mean-payoff threshold problem in G.*

Proof. Assume there exists a winning strategy of $\mathcal{P}_1$ for the bounded window mean-payoff problem in $G_{+\varepsilon}$ from initial state $s \in S$. By Lemma 2, assertion (a), we have that this strategy ensures a non-negative mean-payoff in $G_{+\varepsilon}$. By shifting weights by $-\varepsilon$, this can be equivalently expressed as (Prop. A) the existence of a strategy of $\mathcal{P}_1$ ensuring a mean-payoff at least equal to $-\varepsilon$ in the game G.

For sufficiently small values of ε, that is for $0 < \varepsilon < 1/|S|$, we claim that (Prop. A) implies that (Prop. B) $\mathcal{P}_1$ has a strategy to ensure a non-negative mean-payoff in G. By contradiction, assume this implication is false, that is we have that (Prop. A) is true and (Prop. B) is not. It implies the following.

- *(Prop. A) is true*: $\mathcal{P}_1$ has a memoryless strategy to ensure that the mean-payoff is at least equal to $-\varepsilon$, i.e., strictly greater than $-1/|S|$.
- *(Prop. B) is false*: $\mathcal{P}_2$ has a memoryless strategy to ensure that all simple cycles in consistent outcomes have a sum of weights at most -1. Hence, this strategy ensures a mean-payoff at most equal to $-1/|S|$.

Obviously, it is not possible to have both (Prop. A) true and (Prop. B) false for any initial state $s \in S$, hence proving our claim. □

Lemma 14. *The one-dimension mean-payoff problem reduces in polynomial time to the bounded window mean-payoff problem.*

Proof. Let $G = (S_1, S_2, E, w)$ be a game with integer weights, and $s_{\mathsf{init}} \in S$ be the initial state. Let ε be any rational value such that $0 < \varepsilon < 1/|S|$. We claim that the answer to the mean-payoff threshold problem in G is YES if and only if the answer to the bounded window mean-payoff problem in $G_{+\varepsilon}$ is YES.

The left-to-right implication is proved in two steps. Assume the answer to the mean-payoff threshold problem in G is YES. First, by Lemma 12, we have that $\mathcal{P}_1$ has a strategy to ensure a strictly positive mean-payoff in $G_{+\varepsilon}$. Second, by Lemma 2, assertion (b), this implies that the answer to the bounded window mean-payoff problem in $G_{+\varepsilon}$ is YES.

The right-to-left implication is straightforward application of Lemma 13. □

Remark 1. The reduction established in Lemma 14 cannot be reversed in order to solve bounded window mean-payoff games via classical mean-payoff games. Indeed, the reduction relies on the absence of simple cycles of value zero in the game $G_{+\varepsilon}$, which is not verified in general if the reduction starts from arbitrary bounded window mean-payoff games. Indeed it does not suffice to shift the weights symmetrically by $-\varepsilon$ to obtain an equivalent mean-payoff game, as witnessed by Fig. 4, for which any negative shift gives a game losing for the mean-payoff threshold problem, while the bounded window problem on the original game is satisfied.

Bounded window: summary. We close our study of two-player one-dimension games with Theorem 3.

Theorem 3. *In two-player one-dimension games, the bounded window mean-payoff problem is in NP $\cap$ coNP and at least as hard as mean-payoff games. Memoryless strategies suffice for $\mathcal{P}_1$ and infinite-memory strategies are required for $\mathcal{P}_2$ in general.*

4.3 Games with k dimensions

In this section, we address the case of two-player games with multi-dimension weights. For the *fixed window mean-payoff problem*, we first present an EXPTIME algorithm that computes the winning states of $\mathcal{P}_1$. We also establish lower bounds on the complexity of the fixed window problem: we show that the problem is EXPTIME-hard (both in the case of fixed weights and arbitrary dimensions, and in the case of a fixed number of dimensions and arbitrary weights) for arbitrary window sizes, whereas it is PSPACE-hard for polynomial window sizes. We show that exponential memory is both sufficient and necessary in general for both players, even for polynomial window sizes. For the *bounded window mean-payoff problem*, we prove non-primitive recursive hardness.

Fixed window: algorithm. We start by providing an EXPTIME algorithm via a reduction from a fixed window mean-payoff game $G = (S_1, S_2, E, k, w)$ to an exponentially larger unweighted co-Büchi game G^c (where the objective of $\mathcal{P}_1$ is to avoid visiting a set of bad states infinitely often).

Lemma 15. *The fixed window mean-payoff problem over a multi-weighted game G reduces in exponential time to the co-Büchi problem on an exponentially larger game G^c.*

Recall that a winning play is such that, starting in some position $i \geq 0$, in all dimensions, all opening windows are closed in at most $l_{\max}$ steps. We keep a counter of the sum over the sequence of edges and as soon as it turns non-negative (in at most $l_{\max}$ steps), we reset the sum counter and start a new sequence (which also must become non-negative in at most $l_{\max}$ steps). Hence, the reduction is based on accounting for each dimension the current negative sum of weights since the last reset, and the number of steps that remain to achieve a non-negative sum. This accounting is encoded in the states of $G^c = (S_1^c, S_2^c, E^c)$, as from the original state space S, we go to the extended state space $S \times (\{-l_{\max} \cdot W, \ldots, 0\} \times \{1, \ldots, l_{\max}\})^k$: states of G^c are tuples representing a state of G and the current status of open windows in all dimensions (sum and remaining steps). We add states reached whenever a window reaches its maximum size $l_{\max}$ without closing. We label those as *bad* states. We have one bad state for every state of G. Transitions in G^c are built in order to accurately model the effect of transitions of G on open windows. Clearly, a play is winning for the fixed window problem if and only if the corresponding play in G^c is winning for the co-Büchi objective that asks that the set of bad states is not visited infinitely often, as that means that from some point on, all windows close in the required number of steps.

Proof. Let $G = (S_1, S_2, E, k, w)$ be a game with objective $\mathsf{FixWMP}_G(\{0\}^k, l_{\max} \in \mathbb{N}_0)$ and initial state $s_{\mathsf{init}} \in S$. Let W denote the maximal absolute value of any edge in E. We construct the unweighted game $G^c = (S_1^c, S_2^c, E^c)$ in the following way.

- $S_1^c = \Big(S_1 \times (\{-W \cdot l_{\max}, \ldots, 0\} \times \{1, \ldots, l_{\max}\})^k\Big) \cup \{\varsigma_1, \ldots, \varsigma_{|S|}\}$. States $\varsigma_1, \ldots, \varsigma_{|S|}$ denote special added *bad states*, one for each of the original states $s_1, \ldots, s_{|S|} \in S$. The other states are built as tuples that represent (a) a visited state in G, (b) for each dimension, a couple modeling (b.1) the current sum of weights since the last time the sum in this dimension was non-negative, and (b.2) the number of steps that remain to reach a non-negative sum in this dimension (i.e., before reaching the maximum window size).
- $S_2^c = S_2 \times (\{-W \cdot l_{\max}, \ldots, 0\} \times \{1, \ldots, l_{\max}\})^k$.
- We construct the edges $((s_a, (\sigma_a^1, \tau_a^1), \ldots, (\sigma_a^k, \tau_a^k)), (s_b, (\sigma_b^1, \tau_b^1), \ldots, (\sigma_b^k, \tau_b^k))$ of E^c as follows. For all $(s_a, s_b) \in E$, let $w_e = w((s_a, s_b))$, we have
  - $((s_a, (\sigma_a^1, \tau_a^1), \ldots, (\sigma_a^k, \tau_a^k)), \varsigma_b) \in E^c$, with ς_b the bad state associated to state s_b, iff $\exists t$, $1 \leq t \leq k$ such that $\tau_a^t = 1$ and $\sigma_a^t + w_e(t) < 0$,
  - $((s_a, (\sigma_a^1, \tau_a^1), \ldots, (\sigma_a^k, \tau_a^k)), (s_b, (\sigma_b^1, \tau_b^1), \ldots, (\sigma_b^k, \tau_b^k)) \in E^c$ iff $\forall t$, $1 \leq t \leq k$, we have
    - if $\sigma_a^t + w_e(t) \geq 0$ then $\sigma_b^t = 0, \tau_b^t = l_{\max}$,
    - if $\sigma_a^t + w_e(t) < 0 \wedge \tau_a^t > 1$ then $\sigma_b^t = \sigma_a^t + w_e(t), \tau_b^t = \tau_a^t - 1$,

  and we add edges $(\varsigma_i, (s_i, (0, l_{\max}, \ldots, (0, l_{\max}))$ to E^c for all states $s_i \in S$.

Intuitively, the game G^c is built by unfolding the game G and integrating the current sum of weights in the states of G^c, as well as the number of steps that remain to close a window, both for each dimension separately. The game G^c starts in the initial state $(s_{\mathsf{init}}, (0, l_{\max}), \ldots, (0, l_{\max}))$, and each time a transition (s, s') in the original game G is taken, the game G^c is updated to a state $(s', (\sigma^1, \tau^1), \ldots, (\sigma^k, \tau^k))$ such that (a) if the current sum becomes positive in a dimension t, the corresponding sum counter is reset to zero and the step counter is reset to its maximum value, $l_{\max}$, (b) if the sum is still strictly negative in a dimension t and the window for this dimension is not at its maximal size, the sum is updated and the step counter is decreased, and (c) if the sum stays strictly negative and the maximal size is reached in any dimension, the game visits the corresponding bad state and then, all counters are reset for all dimensions.

We argue that a play π in G is winning for the fixed window mean-payoff objective if and only if the corresponding play π^c in G^c is winning for the co-Büchi objective asking not to visit the set $S_\varsigma = \{\varsigma_1, \ldots, \varsigma_{|S|}\}$ infinitely often. Indeed, consider a play π winning for objective $\mathsf{FixWMP}_G(\{0\}^k, l_{\max})$. By Eq. (4), this play only sees a finite number of bad windows (windows that are not closed in $l_{\max}$ steps in some dimension). By construction of G^c, the corresponding play π^c only visits the set S_ς a finite number of times, hence it is winning for the co-Büchi objective. Now, let π^c be a winning play for the co-Büchi objective. By definition, there exists a position i in π^c such that all states appearing after position i belong to $S \setminus S_\varsigma$. It remains to prove that for any position $j \geq i$, for any dimension t, $1 \leq t \leq k$, there is a valid window of size at most $l_{\max}$. Again we use the inductive property of windows. We know by construction that a reset of

the sum happens in at most $l_{\max}$ steps, otherwise we go to a bad state. Assume j is a position with a sum counter of zero in some dimension t, and j' is the next such position. Since resets are done *as soon as* the sum becomes non-negative, all suffixes of the sequence from j to j' are non-negative. Hence, it is clear that for all position j'', $j < j'' < j'$, the window from j'' to j' in dimension t is closed. Consequently, the corresponding play π in G is winning for the fixed window mean-payoff objective of threshold 0 and window size $l_{\max}$. □

As a direct corollary of this reduction, we obtain an EXPTIME algorithm to solve the fixed window mean-payoff problem on multi-dimension games, as solving co-Büchi games takes quadratic time in the size of the game [13].

Corollary 2. *Given a two-player multi-dimension game $G = (S_1, S_2, E, k, w)$ and a window size $l_{\max} \in \mathbb{N}_0$, the fixed window mean-payoff problem can be solved in time $\mathcal{O}(|S|^2 \cdot (l_{\max})^{4\cdot k} \cdot W^{2\cdot k})$ via a reduction to co-Büchi games.*

Proof. Lemma 15 uses a co-Büchi game whose state space is of size

$$\left|S \times \left(\{-W \cdot l_{\max}, \ldots, 0\} \times \{1, \ldots, l_{\max}\}\right)^k\right| + |S| = \mathcal{O}\Big(|S| \cdot (l_{\max})^{2\cdot k} \cdot W^k\Big).$$

The quadratic algorithm for co-Büchi games described in [13] implies the result. □

A natural question is whether a distinct algorithm is useful in the one-dimension case. Remark 2 notes that it is.

Remark 2. The multi-dimension algorithm described in Corollary 2 yields a procedure which is polynomial in the size of the state space, the window size, and the largest weight for the subclass of one-dimension games, hence only *pseudo-polynomial* (i.e., exponential in V, the length of the encoding of weights), whereas Lemma 5 gives a truly polynomial algorithm.

Fixed window: lower bounds. We first consider the fixed *arbitrary* window mean-payoff problem for which we show (i) in Lemma 16, EXPTIME-hardness for $\{-1, 0, 1\}$ weights and arbitrary dimensions via a reduction from the *membership problem for alternating polynomial-space Turing machines (APTMs)* [8], and (ii) in Lemma 17, EXPTIME-hardness for two dimensions and arbitrary weights via a reduction from *countdown games* [28].

Given an APTM $\mathcal{M}$ and a word $\zeta \in \{0,1\}^*$, such that the tape contains at most $p(|\zeta|)$ cells, where p is a polynomial function, the membership problem asks to decide if $\mathcal{M}$ accepts ζ. We build a fixed arbitrary window mean-payoff game G so that $\mathcal{P}_1$ has to simulate the run of $\mathcal{M}$ on ζ, and $\mathcal{P}_1$ has a winning strategy in G if and only if the word is accepted by the machine. For each tape cell $h \in \{1, 2, \ldots, p(|\zeta|)\}$, we have two dimensions, $(h, 0)$ and $(h, 1)$ such that a sum of weights of value -1 (i.e., an open window) in dimension (h, i), $i \in \{0, 1\}$ encodes that in the current configuration of $\mathcal{M}$, tape cell h contains a bit of value i. In each step of the simulation (Fig. 6), $\mathcal{P}_1$ has to disclose the symbol under the tape head: if in position h, $\mathcal{P}_1$ discloses a 0 (resp. a 1), he obtains a reward 1 in dimension $(h, 0)$ (resp. $(h, 1)$). To ensure that $\mathcal{P}_1$ was faithful, $\mathcal{P}_2$ is then given the choice to either let the simulation continue, or assign a reward 1 in all dimensions except $(h, 0)$ and $(h, 1)$ and then restart the game after looping in a zero self-loop for an arbitrary long time. If $\mathcal{P}_1$ cheats by not disclosing the correct symbol under tape cell h, $\mathcal{P}_2$ can punish him by branching to the restart state and ensuring a sufficiently long open window in the corresponding dimension before restarting (as in Fig. 5). But if $\mathcal{P}_1$ discloses the correct symbol and $\mathcal{P}_2$ still branches, all windows close. In the accepting state, all windows are closed and the game is restarted. The window size $l_{\max}$ of the game is function of the existing bound on the length of an accepting run. To force $\mathcal{P}_1$ to go to the accepting state, we add an additional dimension, with weight -1 on the initial edge of the game and weight 1 on reaching the accepting state.

Lemma 16. *The fixed arbitrary window mean-payoff problem is EXPTIME-hard in multi-dimension games with $\{-1, 0, 1\}$ weights and arbitrary dimensions.*

Proof. An *alternating Turing machine* (ATM) [8] is a tuple $\mathcal{M} = (Q, q_0, \Sigma_{\mathsf{in}}, \delta, q_{\mathsf{acc}})$ where:

- Q is the finite set of control states with a partition $(Q_\vee, Q_\wedge)$ of Q into existential and universal states;
- $q_0 \in Q$ is the initial state;
- $\Sigma_{\mathsf{in}} = \{0, 1\}$ is the input alphabet and $\Sigma_{\mathsf{tape}} = \Sigma_{\mathsf{in}} \cup \{\#\}$ the tape alphabet, with # the blank symbol;
- $\delta \subseteq Q \times \Sigma_{\mathsf{tape}} \times Q \times \Sigma_{\mathsf{tape}} \times \{-1, 1\}$ is a transition relation;

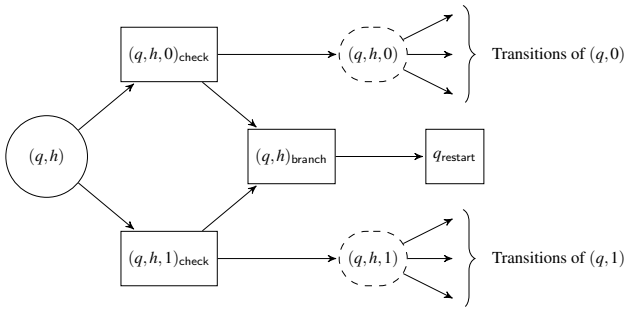

Fig. 6: Gadget ensuring a correct simulation of the APTM on tape cell h.

– there is a special accepting state $q_{\mathsf{acc}} \in Q_\vee$ (without loss of generality).

We say that $\mathcal{M}$ is a *polynomial-space* alternating Turing machine (APTM) if for some polynomial function p, the space used by $\mathcal{M}$ on any input word $\zeta \in \Sigma_{\mathsf{in}}^*$ is bounded by $p(|\zeta|)$.

We define the AND-OR graph of the APTM $(\mathcal{M}, p)$ on the input word $\zeta \in \Sigma_{\mathsf{in}}^*$ as $\mathcal{G}(\mathcal{M}, p) = \langle S_\vee, S_\wedge, s_0, \Delta, R\rangle$ where

– $S_\vee = \{(q,h,t) \mid q \in Q_\vee,\ 1 \le h \le p(|\zeta|) \text{ and } t \in \Sigma_{\mathsf{tape}}^{p(|\zeta|)}\}$;
– $S_\wedge = \{(q,h,t) \mid q \in Q_\wedge,\ 1 \le h \le p(|\zeta|) \text{ and } t \in \Sigma_{\mathsf{tape}}^{p(|\zeta|)}\}$;
– $s_0 = (q_0, 1, t)$ where $t = \zeta \cdot \#^{p(|\zeta|)-|\zeta|}$;
– $((q_1,h_1,t_1),(q_2,h_2,t_2)) \in \Delta$ iff there exists $(q_1, t_1(h_1), q, \gamma, d) \in \delta$ such that $q_2 = q$, $h_2 = h_1 + d$, $t_2(h_1) = \gamma$ and $t_2(h) = t_1(h)$ for all $h \neq h_1$;
– $R = \{(q,h,t) \in S_\vee \mid q = q_{\mathsf{acc}}\}$.

Intuitively, states of the graph correspond to configurations (q,h,t) where q is a control state of the machine, h the position of the tape head, and t the current word written on the tape. Given a state q of the machine $\mathcal{M}$, tape head on cell h and a word t on the tape, a transition from (q,h,t) to (q',h',t') exists in the graph $\mathcal{G}(\mathcal{M},p)$ if the transition relation δ of the machine $\mathcal{M}$ admits a transition that given this configuration, updates the content of cell h to the symbol $t'(h)$, such that the tape now contains the word t', and then goes to control state q' and moves the tape head to an adjacent cell h'.

A word $\zeta \in \Sigma_{\mathsf{in}}^*$ is *accepted* by an APTM $(\mathcal{M}, p)$ if there exists a run tree (obtained by choosing a child in existential nodes and keeping all children in universal nodes) of $\mathcal{M}$ on ζ such that all leafs are accepting configurations. That is, a word is accepted if and only if, in the two-player game defined by $\mathcal{G}(\mathcal{M},p)$, player $\mathcal{P}_\vee$ has a strategy to reach the set of accepting states R. Deciding the acceptance of a word by an APTM is an EXPTIME-complete problem, known as the membership problem [8].

We construct a fixed window mean-payoff game $G = (S_1, S_2, E, k, w)$ simulating the machine $(\mathcal{M}, p)$ as follows. Let $k = 2 \cdot p(|\zeta|) + 1$: there is a dimension for each pair $(h,0)$ and $(h,1)$, for all $1 \le h \le p(|\zeta|)$, and one additional dimension. The set of states S of the game is

$$\begin{aligned} S = &\{q_{\mathsf{restart}}\} \cup \{q_{\mathsf{in}}\} \cup \{\widehat{q_{\mathsf{acc}}}\} \cup \{(q,h) \mid q \in Q,\ 1 \le h \le p(|\zeta|)\} \cup \{(q,h,i)_{\mathsf{check}} \mid q \in Q,\ 1 \le h \le p(|\zeta|), i \in \{0,1\}\} \\ &\cup \{(q,h)_{\mathsf{branch}} \mid q \in Q,\ 1 \le h \le p(|\zeta|)\} \cup \{(q,h,i) \mid q \in Q,\ 1 \le h \le p(|\zeta|), i \in \{0,1\}\}. \end{aligned}$$

States of the form (q,h) belong to $\mathcal{P}_1$. States of the form (q,h,i) belong to $\mathcal{P}_1$ if $q \in Q_\vee$ in the machine $\mathcal{M}$. All other states belong to $\mathcal{P}_2$. The initial state is q_{restart}. It has two outgoing edges with weights zero in all dimensions: one self-loop, and one edge to q_{in}. The latter is assigned the following weights: -1 for dimension (h,i) if the letter at position h of ζ is i, -1 in the very last dimension $(2 \cdot p(|\zeta|) + 1)$, and zero everywhere else. From q_{in}, the game goes to $(q_0, 1)$ and the simulation of $\mathcal{M}$ begins.

The game mimics runs of $\mathcal{M}$, and it is ensured that if the current state of the game is (q,h) and the cell content is i, then the sum of weights since the last visit of q_{in} in dimension (h,i) is -1. We refer to the segment of play since

the last visit of q_{in} as the *current round*. We depict a step of the simulation in Fig. 6. At state (q,h), $\mathcal{P}_1$ has the choice between states $(q,h,0)_{\mathsf{check}}$ and $(q,h,1)_{\mathsf{check}}$, resp. corresponding to declaring a content 0 or 1 of the tape cell h. The reward for dimension (h,i), $i \in \{0,1\}$ is 1 on state $(q,h,i)_{\mathsf{check}}$. At state $(q,h,i)_{\mathsf{check}}$, a state of $\mathcal{P}_2$, $\mathcal{P}_2$ checks whether $\mathcal{P}_1$ has correctly revealed the tape content as follows: (i) Player $\mathcal{P}_2$ can choose to go to state $(q,h)_{\mathsf{branch}}$, in which all dimensions other than $(h,0)$ and $(h,1)$, including the very last, are increased by 1, and then go to q_{restart} on which $\mathcal{P}_2$ will be able to delay the play; (ii) Player $\mathcal{P}_2$ can choose to proceed and continue the simulation: the game then goes to state (q,h,i). State (q,h,i) is either a state of $\mathcal{P}_1$ or $\mathcal{P}_2$, depending on the affiliation of state q in the APTM. Such a gadget ensures that if $\mathcal{P}_1$ cheats by not disclosing the correct symbol, $\mathcal{P}_2$ can force an open window of arbitrary length in the current round by looping on q_{restart} for some time, and then restarting the game. On the other hand, if $\mathcal{P}_1$ is faithful and $\mathcal{P}_2$ still decides to branch to $(q,h)_{\mathsf{branch}}$, then all windows will be closed for the current round.

If $\mathcal{P}_1$ does not cheat and $\mathcal{P}_2$ acknowledges it by not branching, the game advances to a state of the form (q,h,i). At such a state, we add transitions as follows: if there exists a transition from (q,h,i) to (q',h',i') in $\mathcal{M}$, then we add an edge from (q,h,i) to (q',h') in the game G, and assign weight -1 in dimension (h,i'), as the tape cell at position h contains i' and we ensure that the sum in dimension (h,i') in the current round is -1. At the accepting states (q_{acc},h), all dimensions are assigned reward 1, and the next state is $\widehat{q_{\mathsf{acc}}}$. State $\widehat{q_{\mathsf{acc}}}$ is followed by q_{restart}. Again there is no risk in looping as all dimensions are now non-negative.

Formally, blank symbols need to be added. For brevity and simplicity of the presentation, we omit these technical details.

We fix the window size $l_{\max}$ equal to three times the size of the configuration graph (bound on the length of a run) plus three, and we argue that the game G is a faithful simulation of the machine $\mathcal{M}$, that is, $\mathcal{P}_1$ wins the fixed window mean-payoff game if and only if the word ζ is accepted by $\mathcal{M}$. Notice that the construction ensures that if $\mathcal{P}_1$ cheats in the current round, $\mathcal{P}_2$ can make this round losing, as discussed before. Similarly, if $\mathcal{P}_1$ does not cheat but does not reach the accepting state, dimension $2 \cdot p(|\zeta|)+1$ will remain negative when arriving in q_{restart} and $\mathcal{P}_2$ will be able to cycle long enough to make the round losing as the window in the last dimension will remain open for $l_{\max}$ steps. Clearly, $\mathcal{P}_1$ cannot see losing rounds infinitely often otherwise the play is losing. Assume the word ζ is accepted by the machine. Then there is an accepting run tree, and the winning strategy of $\mathcal{P}_1$ is to follow this run tree and always reveal the correct symbol. This way, either $\mathcal{P}_2$ restarts and the round is winning because all dimensions are non-negative, or $\mathcal{P}_2$ does not restart and an accepting state (q_{acc},h) is reached within the maximum allowed window size. Indeed, in the APTM, there is a strategy to reach the accepting state in a number of steps bounded by the size of the configuration graph. In that case, the round is also winning. Conversely, assume that the word ζ is not accepted by the APTM. Consider any strategy λ_1 of $\mathcal{P}_1$. Clearly, $\mathcal{P}_1$ cannot cheat as otherwise, he loses. So assume he does not cheat. Then there is a path in the run tree obtained from playing the strategy λ_1 in $\mathcal{M}$ such that the path never reaches an accepting state. Hence, the strategy λ_2 of $\mathcal{P}_2$ that follows this path in the game G ensures that the sum in dimension $2 \cdot p(|\zeta|)+1$ is always strictly negative, and after waiting till the bound $l_{\max}$ on the window size is met, $\mathcal{P}_2$ has made the round losing and he can restart the game safely. Acting this way infinitely often, $\mathcal{P}_2$ can violate the fixed window objective for $\mathcal{P}_1$. It follows that $\mathcal{P}_1$ wins in G if and only if the word ζ is accepted by the APTM $\mathcal{M}$. □

We now prove EXPTIME-hardness for two dimensions and arbitrary weights via a reduction from countdown games. A countdown game $\mathcal{C}$ consists of a weighted graph $(\mathcal{S},\mathcal{T})$, with $\mathcal{S}$ the set of states and $\mathcal{T} \subseteq \mathcal{S} \times \mathbb{N}_0 \times \mathcal{S}$ the transition relation. Configurations are of the form (s,c), $s \in \mathcal{S}$, $c \in \mathbb{N}$. The game starts in an initial configuration (s_{init},c_0) and transitions from a configuration (s,c) are performed as follows: first $\mathcal{P}_1$ chooses a duration d, $0 < d \leq c$ such that there exists $t = (s,d,s') \in \mathcal{T}$ for some $s' \in \mathcal{S}$, second $\mathcal{P}_2$ chooses a state $s' \in \mathcal{S}$ such that $t = (s,d,s') \in \mathcal{T}$. Then, the game advances to $(s',c-d)$. Terminal configurations are reached whenever no legitimate move is available. If such a configuration is of the form $(s,0)$, $\mathcal{P}_1$ wins the play. Otherwise, $\mathcal{P}_2$ wins the play. Deciding the winner in countdown games given an initial configuration (s_{init},c_0) is EXPTIME-complete [28].

Given a countdown game $\mathcal{C}$ and an initial configuration (s_{init},c_0), we create a game $G = (S_1,S_2,E,k,w)$ with $k = 2$ and a fixed window objective for $l_{\max} = 2 \cdot c_0 + 2$. The two dimensions are used to store the value of the countdown counter and its opposite. Each time a duration d is chosen, an edge of value $(-d,d)$ is taken. The game simulates the moves available in $\mathcal{C}$: a strict alternation between states of $\mathcal{P}_1$ (representing states of $\mathcal{S}$) and states of $\mathcal{P}_2$ (representing transitions available from a state of $\mathcal{S}$ once a duration has been chosen). On states of $\mathcal{P}_1$, we add the possibility to branch to a state s_{restart} of $\mathcal{P}_2$, in which $\mathcal{P}_2$ can either take a zero cycle, or go back to the initial state and force a restart of the game. By placing weights $(0,-c_0)$ on the initial edge, and $(c_0,0)$ on the edge branching to s_{restart}, we ensure

that the only way to win for $\mathcal{P}_1$ is to accumulate a value exactly equal to c_0 in the game before switching to s_{restart}. This is possible if and only if $\mathcal{P}_1$ can reach a configuration of value zero in $\mathcal{C}$.

Lemma 17. *The fixed arbitrary window mean-payoff problem is EXPTIME-hard in multi-dimension games with two dimensions and arbitrary weights.*

Proof. We establish a polynomial-time reduction from the countdown game problem to the fixed arbitrary window problem. Let $\mathcal{C} = (\mathcal{S}, \mathcal{T})$ be a countdown game [28], with initial configuration (s_{init}, c_0). We create a corresponding game $G = (S_1, S_2, E, k, w)$ as follows.

- $S_1 = \mathcal{S}$.
- Let $S^{\mathcal{T}} \subseteq \mathcal{S} \times \mathbb{N}_0$ be the subset of pairs (s,d) such that there exists a transition $(s,d,s') \in \mathcal{T}$. Then, $S_2 = S^{\mathcal{T}} \cup \{s_{\mathsf{restart}}\}$. State s_{restart} is the initial state of game G.
- For each transition $(s,d,s') \in \mathcal{T}$, we add edges $(s,(s,d))$, with $s \in S_1$ and $(s,d) \in S_2$, and $((s,d),s')$, with $s' \in S_1$, to the set of edges E. Edge $(s,(s,d))$ has weight $(-d,d)$ and edge $((s,d),s')$ has weight $(0,0)$.
- For all $s \in S_1$, we add an edge $(s, s_{\mathsf{restart}})$ of weight $(c_0, 0)$.
- From s_{restart}, we add an edge $(s_{\mathsf{restart}}, s_{\mathsf{init}})$ of value $(0, -c_0)$.
- On s_{restart}, we add a self-loop $(s_{\mathsf{restart}}, s_{\mathsf{restart}})$ of weight $(0,0)$.

We fix the window size $l_{\max} = 2 \cdot c_0 + 2$, and we claim that $\mathcal{P}_1$ wins the fixed window problem if and only if he wins the countdown game. Recall that to win a countdown game, $\mathcal{P}_1$ must be able to reach a configuration $(s,0)$ in the game $\mathcal{C}$. The key idea to our construction is that in the game G, the only way to avoid seeing infinitely often open windows of size larger than $l_{\max}$ is to accumulate exactly c_0 before restarting, which is equivalent to reaching a configuration of value 0 in $\mathcal{C}$.

Notice that the game G starts by visiting an edge of value $(0, -c_0)$ and afterwards, all edges from states of $\mathcal{P}_1$ have a value $(-d,d)$ corresponding to the duration he chooses in the countdown game. All except the edge he can decide to take to go to s_{restart}, which value is $(c_0, 0)$. Clearly, if $\mathcal{P}_1$ decides to go in s_{restart}, he has to close all windows, as otherwise $\mathcal{P}_2$ can use the self-loop to delay the play long enough and provoke a sufficiently long bad window, which if done repeatedly, induces a losing play. On the other hand, if $\mathcal{P}_1$ decides to never go towards s_{restart}, he will keep accumulating negative values in the first dimension and he is guaranteed to lose. So obviously the behavior of $\mathcal{P}_1$ should be to play as in the countdown game to accumulate exactly c_0 in dimension 2 (and $-c_0$ in dimension 1) before switching to s_{restart}, so that $\mathcal{P}_2$ can do no harm by delaying the play as all windows will be closed. The accumulated value has to be *exactly* c_0 as (a) if it is less than c_0, dimension 2 will remain negative, and (b) if it is more than c_0, dimension 1 will stay negative (i.e., the edge $(s, s_{\mathsf{restart}})$ will not suffice to get it back above zero). Since the minimal increase is of 1 every two edges by construction, the allowed window size $l_{\max}$ is sufficient to enforce such a behavior, if possible. This shows that $\mathcal{P}_1$ wins the fixed window problem from initial state s_{restart} in G if and only if he also wins the countdown game $\mathcal{C}$ from (s_{init}, c_0), as accumulating c_0 in G is equivalent to reaching a configuration of value zero in $\mathcal{C}$. □

For the case of polynomial windows, Lemma 18 proves PSPACE-hardness via a reduction from generalized reachability games [22]. Filling the gap with the EXPTIME-membership given by Corollary 2 is an open problem. The generalized reachability objective is a conjunction of reachability objectives: a winning play has to visit a state of each of a series of k reachability sets. If $\mathcal{P}_1$ has a winning strategy in a generalized reachability game $G^r = (S_1^r, S_2^r, E^r)$, then he has one that guarantees visit of all sets within $k \cdot |S^r|$ steps. We create a modified weighted version of the game, $G = (S_1, S_2, E, k, w)$, such that the weights are k-dimension vectors. The game starts by opening a window in all dimensions and the only way for $\mathcal{P}_1$ to close the window in dimension t, $1 \leq t \leq k$ is to reach a state of the t-th reachability set. We modify the game by giving $\mathcal{P}_2$ the ability to close all open windows and restart the game such that the prefix-independence of the fixed window objective cannot help $\mathcal{P}_1$ to win without reaching the target sets. Then, a play is winning in G for the fixed window objective of size $l_{\max} = 2 \cdot k \cdot |S^r|$ if and only if it is winning for the generalized reachability objective in G^r.

Lemma 18. *The fixed polynomial window mean-payoff problem is PSPACE-hard.*

Proof. We show the PSPACE-hardness by a reduction from the generalized reachability problem [22]. Given a game graph $G^r = (S_1^r, S_2^r, E^r)$, a series of reachability sets $R_t \subseteq S^r$, for $1 \leq t \leq k$, with $k \leq |S^r|$, and an initial state $s^r_{\mathsf{init}} \in S^r$, the generalized reachability problem asks if there exists a strategy of $\mathcal{P}_1$ such that any consistent outcome starting in s^r_{init} visits a state of each set R_t at least once. It is known that if such a strategy exists, then there exists one which ensures reaching all sets in at most $k \cdot |S^r|$ steps.

We build a k-dimension fixed window mean-payoff game $G = (S_1, S_2, E, k, w)$ as follows. We define $S_{\mathsf{branch}} \subset S_2$, a set of additional states belonging to $\mathcal{P}_2$ and of the form $b_{s,s'}$, one for each $(s,s') \in E^r$. Let $S_1 = S_1^r$ and $S_2 = S_2^r \cup S_{\mathsf{branch}} \cup \{s_{\mathsf{restart}}\}$. Let E be the set of edges such that for all $(s,s') \in E^r$, we have that $(s, b_{s,s'}) \in E$, $(b_{s,s'}, s') \in E$, $(b_{s,s'}, s_{\mathsf{restart}}) \in E$, and such that $(s_{\mathsf{restart}}, s^r_{\mathsf{init}}) \in E$. That is, we introduce in all edges of E^r a state of $\mathcal{P}_2$ that let him branch to an added state s_{restart} or continue as in G^r. The new initial state in G is s_{restart}, and there is an edge from s_{restart} to the old initial state s^r_{init}. The weights are as follows: all edges from states $b_{s,s'}$ to s_{restart} have value 1 in all dimensions. The edge from s_{restart} to s^r_{init} has value -1 in all dimensions. All other edges of the game have value zero, except edges entering a state that belongs to a reachability set R_t, which have value 1 in dimension t and 0 in the other dimensions. If a state belongs to several sets, then all corresponding dimensions get a 1.

We claim that $\mathcal{P}_1$ has a winning strategy for $\mathsf{FixWMP}_G(\{0\}^k, l_{\max} = 2 \cdot k \cdot |S^r|)$ if and only if he has a winning strategy for the generalized reachability objective in G^r. Consider the game G. The only edge involving negative values is $(s_{\mathsf{restart}}, s^r_{\mathsf{init}})$, which value is $(-1, \ldots, -1)$. Therefore, a losing play for Eq. (4) should see this edge infinitely often, as it is the starting position of all open windows. Notice that on the other hand, going from a state $b_{s,s'}$ to s_{restart} involves an edge of value $(1, \ldots, 1)$, hence if the open window starting in s_{restart} comes back in s_{restart} before hitting its maximal size, the window will close. So the strategy of $\mathcal{P}_2$ should be to wait for $l_{\max} = 2 \cdot k \cdot |S^r|$ steps before forcing a restart. Now, consider a winning strategy λ_1 of $\mathcal{P}_1$ in G. Because of the strategy of $\mathcal{P}_2$, λ_1 has to ensure obtaining $+1$ in all dimensions by only using transitions entering in states of S^r. By construction, this implies that λ_1 enforces a visit of all reachability sets, and thus is winning for the generalized reachability problem. Consider the converse implication. Let λ_1^r be a winning strategy in G^r. There exists such a strategy that ensures seeing all reachability sets (thus closing all windows) in at most $l_{\max} = 2 \cdot k \cdot |S^r|$ steps if $\mathcal{P}_2$ does not branch to s_{restart}. On the other hand, if $\mathcal{P}_2$ does branch before $l_{\max}$ steps, all windows also close, as branching edges have value $(1, \ldots, 1)$. Hence, this strategy is also winning for $\mathsf{FixWMP}_G(\{0\}^k, l_{\max})$. This shows the correctness of the reduction and concludes our proof. □

We conclude our study of the multi-dimension fixed window problem by considering memory bounds. A direct corollary of Lemma 15 is the existence of winning strategies of at most exponential size for both players, as memoryless strategies are sufficient in co-Büchi games [21]. A corollary of the reduction from generalized reachability games to the fixed polynomial window problem used to prove Lemma 18 and the results of [22, Lemma 2] (showing exponential lower bounds on memory for generalized reachability objectives) is that such memory is needed in general, again for both players.

Another example of a family of games in which $\mathcal{P}_1$ requires exponential memory (in the number of dimensions) is given by the family defined in [18, Lemma 8] (Fig. 7), introduced in the context of multi energy games.

Example 1. We define a family of games $(G(K))_{K \geq 1}$ which is an assembly of $k = 2 \cdot K$ gadgets, the first K belonging to $\mathcal{P}_2$, and the remaining K belonging to $\mathcal{P}_1$ (Fig. 7). Precisely, we have $|S_1| = |S_2| = 3 \cdot K$, $|S| = |E| = 6 \cdot K = 3 \cdot k$ (linear in k), $k = 2 \cdot K$, and w defined as:

$$\begin{aligned}
&\forall\, 1 \leq i \leq K, w((\circ, s_i)) = w((\circ, t_i)) = (0, \ldots, 0),\\
&\qquad w((s_i, s_{i,L})) = -w((s_i, s_{i,R})) = w((t_i, t_{i,L})) = -w((t_i, t_{i,R})),\\
&\forall\, 1 \leq j \leq k, w((s_i, s_{i,L}))(j) = \begin{cases} 1 \text{ if } j = 2 \cdot i - 1 \\ -1 \text{ if } j = 2 \cdot i \\ 0 \text{ otherwise} \end{cases},
\end{aligned}$$

where $\circ$ denotes any valid predecessor state.

Essentially, in each state s_i, $\mathcal{P}_2$ can open a window on either dimension $2 \cdot i - 1$ or dimension $2 \cdot i$ by choosing the corresponding edge. In this game, $\mathcal{P}_1$ wins objective $\mathsf{FixWMP}_G(\{0\}^k, l_{\max} = |S|/2)$ only if he is able to make in t_i the opposite choice of $\mathcal{P}_2$ in s_i, as this ensures closure of the corresponding window. This requires a strategy encoded as a

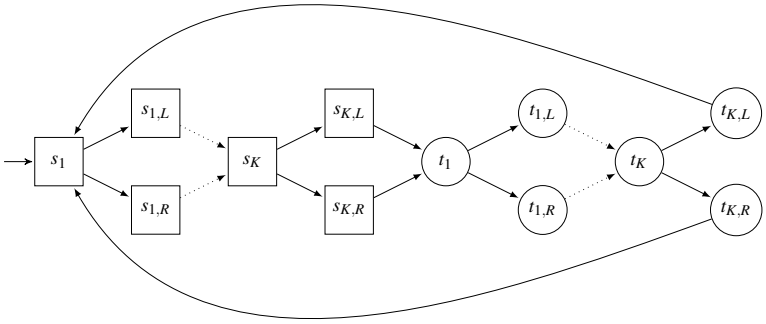

Fig. 7: Family of multi-dimension games requiring exponential memory for $\mathcal{P}_1$, for the fixed window objective.

Moore machine with at least $2^{k/2}$ states. Indeed, if $\mathcal{P}_1$ cannot differentiate between the exponential number of histories from s_i up to t_i, he is not able to enforce closure of the needed windows.

Lemma 19. *In multi-dimension games with a fixed window mean-payoff objective, exponential memory is both sufficient and necessary for both players in general, even for polynomial window sizes.*

Fixed window: summary. We summarize the complexity of the fixed window problem in Theorem 4.

Theorem 4. *In two-player multi-dimension games, the fixed arbitrary window mean-payoff problem is EXPTIME-complete, and the fixed polynomial window mean-payoff problem is PSPACE-hard. For both players, exponential memory is sufficient and is required in general.*

Bounded window. Unlike the one-dimension case, in which it is easier to decide the bounded problem than the fixed arbitrary one (i.e., the problem becomes easier when the fixed window size is sufficiently large), we prove that the complexity of the bounded window problem in multi-weighted games is at least non-primitive recursive.[10] Hence, there is no hope for efficient algorithms on the complete class of two-player multi-weighted games. This result is obtained by reduction from the problem of deciding the existence of an infinite execution in a *marked reset net*, also known as the *termination problem*. A marked reset net [19] is a Petri net with *reset arcs* together with an initial marking of its places. Reset arcs are special arcs that reset a place (i.e., empty it of all its tokens). The termination problem for reset nets is decidable but non-primitive recursive hard (as follows from [38], also discussed in [32]).

Given a reset net $\mathcal{N}$ with an initial marking $\overline{m_0} \in \mathbb{N}^{|P|}$ (where P is the set of places of the net), we build a two-player multi-weighted game G with $k = |P| + 3$ dimensions such that $\mathcal{P}_1$ wins the bounded window objective for threshold $\{0\}^k$ if and only if $\mathcal{N}$ does not have an infinite execution from $\overline{m_0}$.

A high level description of our reduction is as follows. The structure of the game (Fig. 8) is based on the alternance between two gadgets simulating the net (Fig. 9). Edges are labeled by k-dimension weight vectors such that the first $|P|$ dimensions are used to encode the number of tokens in each place. In each gadget, $\mathcal{P}_2$ chooses transitions to simulate an execution of the net. During a faithful simulation, there is always a running open window in all the first $|P|$ dimensions: if place p contains n tokens then the negative sum from the start of the simulation is $-(n+1)$. This is achieved as follows: if a transition t consumes $\mathbf{I}(t)(p)$ tokens from p, then this value is added on the corresponding dimension, and if t produces $\mathbf{O}(t)(p)$ tokens in p, then $\mathbf{O}(t)(p)$ is removed from the corresponding dimension. When a place p is reset, a gadget ensures that dimension p reaches value -1 (the coding of zero tokens). This is thanks to the monotonicity property of reset nets: if $\mathcal{P}_1$ does not simulate a full reset, then the situation gets easier for $\mathcal{P}_2$ as it

[10] That is, there exists no primitive recursive function that computes the answer to the bounded window problem. A well-known example of a decidable but non-primitive recursive function is the Ackermann function [1].

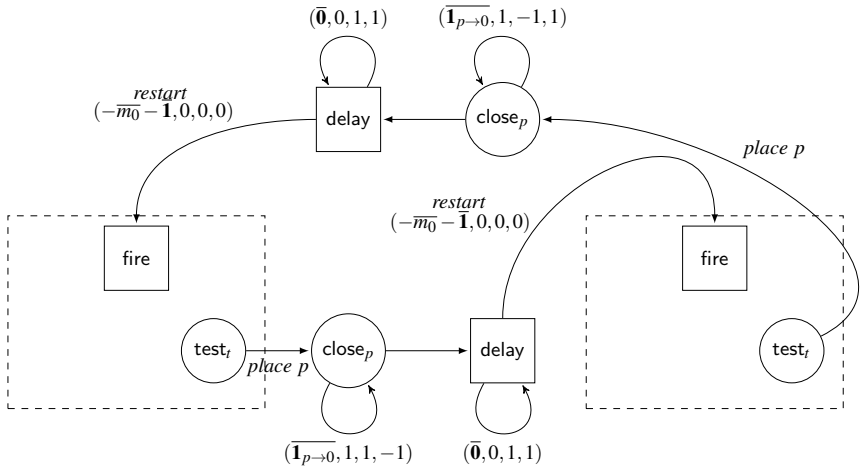

Fig. 8: Careful alternation between gadgets is needed in order for $\mathcal{P}_1$ to win.

leaves him more tokens available. If all executions terminate, $\mathcal{P}_2$ has to choose an unfireable transition at some point, consuming unavailable tokens from some place $p \in P$. If so, the window in dimension p closes. After each transition choice of $\mathcal{P}_2$, $\mathcal{P}_1$ can either continue the simulation or branch out of the gadget to close all windows, except in some dimension p of his choice. Then $\mathcal{P}_2$ can arbitrarily extend any still open window in the first $(|P|+1)$ dimensions and restart the game afterwards. Dimension $(|P|+1)$ prevents $\mathcal{P}_1$ from staying forever in a gadget. If an infinite execution exists, $\mathcal{P}_2$ simulates it and never has to choose an unfireable transition. Hence, when $\mathcal{P}_1$ branches out, the window in some dimension p stays open. The last two dimensions force him to alternate between gadgets so that he cannot take profit of the prefix-independence to win after a faithful simulation. So, $\mathcal{P}_2$ can delay the closing of the open window for longer and longer, thus winning the game.

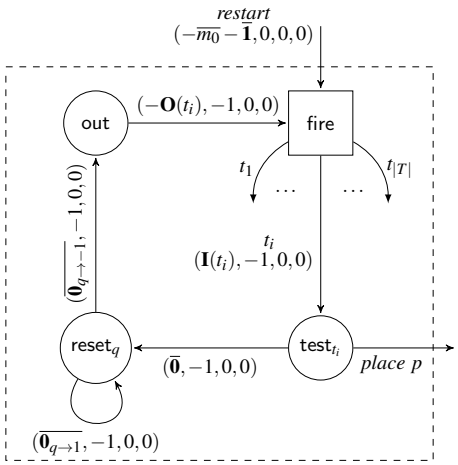

Fig. 9: Gadget simulating an execution of the reset net.

Theorem 5. *In two-player multi-dimension games, the bounded window mean-payoff problem is non-primitive recursive hard.*

Proof. We prove a reduction from the termination problem on reset nets to the bounded window problem on two-player multi-weighted games. The former is known to be non-primitive recursive hard [38,32].

Let $\mathcal{N} = \langle P, T, \mathbf{I}, \mathbf{O}, r \rangle$ be a *reset net* such that

- $P = \{p_1, p_2, \ldots, p_{|P|}\}$ is the set of places;
- $T = \{t_1, t_2, \ldots, t_{|T|}\}$ is the set of transitions;
- $\mathbf{I}\colon T \to \mathbb{N}^{|P|}$ is the input function, such that for each transition $t \in T$, $\mathbf{I}(t)$ is a $|P|$-dimension vector such that for all dimension $p \in \{1, \ldots, |P|\}$, $\mathbf{I}(t)(p)$ specifies the number of tokens from place p consumed by the transition t;[11]
- $\mathbf{O}\colon T \to \mathbb{N}^{|P|}$ is the output function, such that for each transition $t \in T$, $\mathbf{O}(t)$ is a $|P|$-dimension vector such that for all dimension $p \in \{1, \ldots, |P|\}$, $\mathbf{O}(t)(p)$ specifies the number of tokens produced in place p by the transition t;
- $r\colon T \to P$ is the reset function, such that for all transition $t \in T$, $r(t)$ specifies the unique place (w.l.o.g.) which is reset by transition t.

Given an initial marking of the places (i.e., an initial number of tokens in each place) $\overline{m_0} \in \mathbb{N}^{|P|}$, the termination problem asks if there exists an infinite execution of the net, that is, if there exists an infinite sequence of transitions that can be fired from $\overline{m_0}$. A transition t is *fireable* from marking $\overline{m} \in \mathbb{N}^{|P|}$ if for all place $p \in P$, $\mathbf{I}(t)(p) \leq \overline{m}(p)$. An execution terminates if no transition can be fired because the necessary tokens are unavailable. We first note an important *monotonicity* property of reset nets: for all reset net $\mathcal{N} = \langle P, T, \mathbf{I}, \mathbf{O}, r \rangle$, for all markings $\overline{m}, \overline{n} \in \mathbb{N}^{|P|}$, if $\overline{m} \leq \overline{n}$ and $\rho \in T^\omega$ is an infinite sequence of transitions fireable from $\overline{m}$, then ρ is also fireable from $\overline{n}$. This property is used later on.

We claim that given a reset net $\mathcal{N}$ and an initial marking $\overline{m_0}$, we can build in polynomial time a multi-weighted game G in which $\mathcal{P}_1$ has a winning strategy for objective $\mathsf{BndWMP}_G(0)$ if and only if there exists no infinite execution of the net $\mathcal{N}$ from $\overline{m_0}$.

We build the game $G = (S_1, S_2, E, k, w)$ with $k = |P| + 3$ as represented in Fig. 8 and Fig. 9. Unlabeled edges have value zero in all dimensions. For clarity, we define the following $|P|$-dimension integer vectors: $\overline{\mathbf{1}} = (1, \ldots, 1)$ is the unit vector, $\overline{\mathbf{0}} = (0, \ldots, 0)$ is the zero vector, and, for $a, b \in \mathbb{Z}$, $p \in P$, the vector $\overline{\mathbf{a}_{p \to b}}$ represents the vector $(a, \ldots, a, b, a, \ldots, a)$ which has value b in dimension p and a in the other dimensions. The first $|P|$ dimensions of the game are used to encode the tokens present in each place, whereas the last three are used to compel $\mathcal{P}_1$ to act fairly. Our construction will ensure that at all times along a valid execution of the net in a gadget, if a place $p \in P$ possess n tokens, then the running sum of weights over the largest open window has value $(-n-1)$ in dimension p.

The states and edges of the game are built as follows.

- Inside a gadget, we have a state fire belonging to $\mathcal{P}_2$, with $|T|$ outgoing edges corresponding to the $|T|$ transitions of the net. Each transition t is encoded as follows:
  - an edge from fire to a state test_t belonging to $\mathcal{P}_1$, of value $(\mathbf{I}(t), -1, 0, 0)$, such that the running sum is updated to accurately encode the consumption of tokens;
  - in state test_t, $(|P|+1)$ outgoing edges, giving $\mathcal{P}_1$ the possibility to either branch out of the gadget, going to the state close_p corresponding to the dimension p of his choice, or continuing via an edge of value $(\overline{\mathbf{0}}, -1, 0, 0)$ to the reset_q state, a state of $\mathcal{P}_1$ such that $q = r(t)$ is the unique place reset by transition t;
  - a self-loop of value $(\overline{\mathbf{0}_{q \to 1}}, -1, 0, 0)$ on the reset_q state;
  - an edge from reset_q to out_t of value $(\overline{\mathbf{0}_{q \to -1}}, -1, 0, 0)$ which purpose is to ensure that in dimension q, there is a new open window of sum -1 after a full reset (i.e., it encodes that the number of tokens in place q is zero);
  - an edge from out_t back to fire of value $(-\mathbf{O}(t), -1, 0, 0)$, producing tokens according to the output of transition t.
- Branching from the left gadget leads to a state $\mathsf{close}_p^{\text{left}}$ of $\mathcal{P}_1$ with a self-loop of weight $(\overline{\mathbf{1}_{p \to 0}}, 1, 1, -1)$ and an outgoing edge to state $\mathsf{delay}^{\text{left}}$ of $\mathcal{P}_2$.
- State $\mathsf{delay}^{\text{left}}$ possess a self-loop of value $(\overline{\mathbf{0}}, 0, 1, 1)$ and an edge going to the right gadget with value $(-\overline{m_0} - \overline{\mathbf{1}}, 0, 0, 0)$.
- The right gadget is constructed symmetrically, the only change being that the self-loop on states $\mathsf{close}_p^{\text{right}}$ of $\mathcal{P}_1$ now has value $(\overline{\mathbf{1}_{p \to 0}}, 1, -1, 1)$.

[11] For simplicity, we use p to refer to a place $p \in P$ and to the number $i \in \{1, \ldots, |P|\}$ such that $p_i = p$, that is p indistinctly refers to the place and the corresponding dimension in the weight vectors.

The game starts in the left gadget with an initial edge of value $(-\overline{m_0} - \overline{\mathbf{1}}, 0, 0, 0)$ corresponding to the initial marking of the net.

We claim that (i) if there exists no infinite execution $\rho \in T^\omega$ of the net $\mathcal{N}$, then $\mathcal{P}_1$ has a winning strategy in G for the bounded window objective, and (ii) if there exists such an execution, then $\mathcal{P}_2$ has a winning strategy in G. By determinacy, proving both claims will conclude our proof.

Case (i). Assume that there exists no infinite execution $\rho \in T^\omega$ of the net. Then there exists a bound $b \in \mathbb{N}$ on the length of any valid execution. Hence, $\mathcal{P}_2$ can only simulate the net faithfully for b steps, so after at most $(b+1)$ steps, he needs to use an unfireable transition. That is, the next chosen transition requires more tokens than available in some place $p \in P$. We define a winning strategy $\lambda_1 \in \Lambda_1$ of $\mathcal{P}_1$ in G as follows:

1. In a state test_t, if the last transition t was valid (i.e., all first $|P|$ dimensions have a negative running sum), go to the corresponding reset_q state. Otherwise, there exists a dimension p in which the sum has become non-negative and all windows are closed: exit the gadget and go to the corresponding state close_p.
2. In a state reset_q, cycle until the sum in dimension q takes value 0, then go to state out_t.
3. In a state close_p, take the loop exactly $f(b)$ times before going to state delay, where $f\colon \mathbb{N} \to \mathbb{N}$ is a well-chosen function that we define below (hence $f(b)$ is constant along the play).

We claim that it is possible to define $f(b)$ sufficiently large to ensure that this strategy is winning. Let $M \in \mathbb{N}$ be the largest number of tokens produced as output of any transition of the net, on any place. We consider the value of the negative sum in any of the first $(|P|+1)$ dimensions at the moment when $\mathcal{P}_1$ decides to exit the gadget according to the strategy λ_1. Notice that for any dimension $p \in \{1, \ldots, |P|\}$, this sum is bounded by $x = (-\overline{m_0}(p) - 1 - b \cdot M)$. Hence, the number of loops taken on any visit of state reset_q is bounded by x. The sum in dimension $(|P|+1)$ is thus bounded by $(b \cdot (4+x)+1)$, which we define as $f(b)$. The last two dimensions are not modified inside a gadget. Now clearly, looping in state close_p for $f(b)$ steps is sufficient to close all windows in all dimensions corresponding to places (recall that dimension p is closed by $\mathcal{P}_2$ cheating on place p), as well as in dimension $(|P|+1)$. However, this loop opens a window in one of the last two dimensions (the last for the left gadget, and the second to last for the right gadget). As the delay state of $\mathcal{P}_2$ has a positive effect in those dimensions, if $\mathcal{P}_2$ decides to delay the play for $f(b)$ steps, all windows will be closed. If he does not delay, the play will proceed to the next gadget, in which $\mathcal{P}_2$ is also forced to cheat before $(b+1)$ transitions. Hence after looping for $f(b)$ steps in the corresponding close_p state, the open window will close (and another will open in the other dimension which will in turn be closed after the next gadget). By keeping this behavior, $\mathcal{P}_1$ can thus enforce that any open window along the play will close in at most $(4 \cdot f(b) + 4)$ steps. Thus the outcome is winning for the bounded window objective.

Case (ii). Assume that there exists an infinite execution $\rho \in T^\omega$ of the net. We define a winning strategy $\lambda_2 \in \Lambda_2$ of $\mathcal{P}_2$ as follows. The strategy is played in rounds, with the initial round being round 1.

1. Every time a gadget is entered, start playing in state fire according to the infinite execution ρ, that is, choose transitions in order to obtain the same trace.
2. When a state delay is visited during round n, take the self-loop n times then continue to state fire and start round $n+1$.

Notice that this strategy requires infinite memory. We claim that any consistent outcome of the game is winning for $\mathcal{P}_2$, that is, it does not belong to $\mathsf{BndWMP}_G(0)$. First, $\mathcal{P}_1$ cannot stay forever in a gadget, thanks to dimension $(|P|+1)$: he has to branch at some point otherwise the play is lost. Second, if in state reset_q, $\mathcal{P}_1$ decides to cycle for less than necessary for a full reset, the situation gets better for $\mathcal{P}_2$ by the monotonicity property of the reset net (as $\mathcal{P}_2$ gets to continue with more tokens than expected). Notice that $\mathcal{P}_1$ cannot accumulate positive values in the sum, as the next edge will restart a new window and all accumulation will be forgotten with regard to the objective. Third, if $\mathcal{P}_1$ branches and exits the gadget to go to some state close_p, then all dimensions corresponding to places, including dimension p, have a running open window (dimension p has a strictly negative value since $\mathcal{P}_2$ does not cheat). Hence, no matter how long $\mathcal{P}_1$ chooses the self-loop, the window in dimension p will stay open (and $\mathcal{P}_1$ cannot stay here forever because of the last two dimensions). Fourth, when the play reaches a state delay with an open window in dimension $p \in \{1, \ldots, |P|\}$, the strategy λ_2 prescribes that $\mathcal{P}_2$ will loop for longer and longer periods of time, thus enforcing open windows of constantly growing length. As a consequence, any consistent outcome is such that the bounded window objective is not satisfied, which proves our point and further concludes our proof. □

Remark 3. Theorem 5 establishes that the bounded window mean-payoff problem is non-primitive recursive hard, and the decidability of the problem remains open. Note that Theorem 5 also implies that $\mathcal{P}_1$ may require a window size of non-primitive recursive length to win a multi-dimension bounded window mean-payoff game (in contrast to the pseudo-polynomial bound of the one-dimension case given in Corollary 1). The main motivation to study window objectives as a strengthening and approximation of the original objectives is to ensure the objectives in every sliding window of reasonable size. A prohibitively large window size of non-primitive recursive length suggests that the decidability of the bounded window problem is purely of theoretical interest, and the fixed window problem is the more relevant question.

4.4 On direct objectives

Through this paper, we have studied the prefix-independent versions of the objectives defined in Sec. 4.1. In this section, we briefly argue that similar complexity results are obtained for the *direct* variants (Table 2), by slight modifications of the presented proofs. Notice that memory requirements however change, as it is now sufficient to force one sufficiently long (for the fixed problem) or never closing (for the bounded problem) window to make an outcome losing.

<table>
<tr><th></th><th colspan="3">one-dimension</th><th colspan="3">k-dimension</th></tr>
<tr><th></th><th>complexity</th><th>$\mathcal{P}_1$ mem.</th><th>$\mathcal{P}_2$ mem.</th><th>complexity</th><th>$\mathcal{P}_1$ mem.</th><th>$\mathcal{P}_2$ mem.</th></tr>
<tr><td>direct fixed polynomial window</td><td>P-c.</td><td colspan="2" rowspan="2">mem. req. $\leq$ linear($|S| \cdot l_{\max}$)</td><td>PSPACE-h.
EXP-easy</td><td colspan="2" rowspan="2">exponential</td></tr>
<tr><td>direct fixed arbitrary window</td><td>P($|S|, V, l_{\max}$)</td><td>EXP-c.</td></tr>
<tr><td>direct bounded window problem</td><td>NP$\cap$coNP</td><td>mem-less</td><td>linear</td><td>NPR-h.</td><td>-</td><td>-</td></tr>
</table>

Table 2: Complexities and memory requirements for the direct objectives. Differences with the prefix-independent objectives are in bold.

One-dimension direct fixed window problem. The polynomial algorithm in the size of the game and the size of the window is given by Lemma 4. For polynomial windows, we obtain P-hardness using the proof of Lemma 7 and window size $l_{\max} = 2 \cdot |S|$, as if $\mathcal{P}_1$ can win the reachability game, he has a strategy to do it in at most $|S|$ steps. Lemma 6 extends to direct objectives, and provides linear upper bounds on memory with the same arguments. In particular, the provided examples of games require memory for both players when the direct fixed window objective is considered.

One-dimension direct bounded window problem. We obtain an NP$\cap$coNP algorithm for the direct bounded problem by simplifying BoundedProblem (Lemma 10) as follows: $\mathsf{BoundedProblem}(G) = S \setminus \mathsf{UnbOpenWindow}(G)$. Indeed, as the objective is no longer prefix-independent, it is sufficient for $\mathcal{P}_2$ to force one window that never closes to make the play losing. Hence, the attractor of the set $S \setminus L$ in algorithm BoundedProblem cannot be declared winning for $\mathcal{P}_1$. While memoryless strategies still suffice for $\mathcal{P}_1$ (applying the arguments of Lemma 10), winning strategies for $\mathcal{P}_2$ do not need infinite memory anymore, but at most linear memory. Indeed, a winning strategy of $\mathcal{P}_2$ is the one described in the proof of Lemma 10, but without taking rounds into account (i.e., the play stays forever in round one). To illustrate that memoryless strategies still do not suffice for $\mathcal{P}_2$, consider a variation of Fig. 5, with the initial state being s_2. Clearly, $\mathcal{P}_2$ must first take the cycle to s_1 then loop forever on s_2 to ensure a never closing window. Corollary 1 extends in the direct case and gives the same bound on the window size. Finally, the reduction of mean-payoff games developed in Lemma 14 carries over to the direct bounded window objective, as the game with shifted weights is such that the mean-payoff is strictly positive. In which case, the supremum total-payoff is infinite and Lemma 2 applies, implying the result.

Multi-dimension direct fixed window problem. The following results extend to the direct case.

– *EXPTIME algorithm.* Lemma 15 presents a reduction from fixed window games to exponentially larger co-Büchi games. It is easy to obtain a similar reduction from direct fixed window games by considering a safety objective for $\mathcal{P}_1$ (i.e., reachability for the set of bad states for $\mathcal{P}_2$). This also implies an exponential-time algorithm.

– *EXPTIME-hardness of the arbitrary window problem for weights* $\{-1,0,1\}$ *and arbitrary dimensions*. The reduction of the membership problem for polynomial space alternating Turing machines immediately yields the result for the direct objective. Indeed, the strategies proposed in the proof stay winning for this objective. Note that actually the strategy of $\mathcal{P}_2$ may be simpler, as he may cycle forever on s_{restart} after branching to punish an unfaithful symbol disclosure by keeping a window indefinitely open.
– *EXPTIME-hardness of the arbitrary window problem for two dimensions and arbitrary weights*. The reduction from countdown games established in Lemma 17 extends straightforwardly to direct objectives, and $\mathcal{P}_2$ can use a simpler winning strategy consisting in looping forever in its zero cycle.
– *PSPACE-hardness of the polynomial window problem*. The reduction of generalized reachablity games also holds without modification for the direct fixed polynomial window objective.
– *Exponential memory bounds*. Exponential upper bounds follow from the modified Lemma 15, using safety games. Lower bounds witnessed by Lemma 19 are also verified in the presented game as well as from the reduction of generalized reachablity games.

Multi-dimension direct bounded window problem. Non-primitive recursive hardness (Theorem 5) extends to the direct objective with a simpler construction. Indeed, it is sufficient to consider the game using only the first $(|P|+1)$ dimensions, and consisting of only one gadget, with the branching out of the gadget now going to an absorbing state with a self-loop of weight $\overline{\mathbf{1}_{p\to 0}}$ such that when $\mathcal{P}_1$ decides to branch, all windows get closed eventually, except in the dimension p of his choice, for which the window is only closed if $\mathcal{P}_2$ cheats and stays open forever otherwise.

5 Discussion

Conclusion. The strong relation between mean-payoff and total-payoff in single dimension breaks in multi-weighted games as the total-payoff threshold problem becomes undecidable. We introduced the concept of window objectives, which provide conservative approximations with timing guarantees. We believe that window objectives are interesting on the standpoint of *expressiveness*, as they permit to consider quantitative objectives in a time frame context. Furthermore, window objectives constitute an attractive alternative in terms of *tractability*.

We provided algorithms and optimal complexity bounds for one-dimension games. We notably showed that the fixed window variant can be solved in *polynomial time*, which is not known to be the case for the mean-payoff and total-payoff objectives [41,27,24,5].

In multi-dimensions, fixed window games hold an interesting position. While the associated decision problem is easier to solve than the mean-payoff threshold problem in one-dimension (P instead of NP $\cap$ coNP), it becomes comparatively harder in multi-dimension (PSPACE-hard even for polynomial windows instead of coNP). However, it remains EXPTIME-complete for arbitrary windows, in contrast to the total-payoff which becomes undecidable. In terms of complexity, the problem stands in an interesting middle ground between mean-payoff and total-payoff objectives. For the specific case of polynomial windows, there remains a gap between our exponential-time algorithm and the PSPACE lower bound. Whether we can obtain PSPACE-membership or EXPTIME-hardness for the fixed polynomial window problem in multi-dimension games is an open question.

We also established a prohibitive lower bound on the complexity of multi-dimension bounded window games: they are at least non-primitive recursive hard. It would still be of theoretical interest to know if those games are decidable or not. Techniques used for the undecidability proof of multi-dimension total-payoff games (Thm. 1) cannot be extended easily to the bounded window setting. In particular, our reduction to two-counter machines requires to "memorize" sums of weights both negatively and positively. In the window context, such sums can only be memorized negatively (i.e., while windows stay open), as positive windows are closed and forgotten immediately (this corresponds to so-called resets in Sect. 4.3).

Future work. We mention two interesting questions to investigate. First, in the multi-dimension setting, our definitions of window objectives (Sect. 4.1) are asynchronous: windows on different dimensions are not required to close simultaneously. *Synchronous variants* may be interesting to study but some useful properties are lost in that setting, such as the inductive property on windows. Hence our techniques cannot be extended straightforwardly. Second, conjunction of window objectives with a parity objective would be interesting to consider. Indeed, a similar notion of time bounds on liveness properties was studied by Chatterjee et al. through the concept of *finitary* winning [16,17]. Combining a similar approach with our window objectives seems natural.

References

1. Wilhelm Ackermann. Zum hilbertschen aufbau der reellen zahlen. *Mathematische Annalen*, 99(1):118–133, 1928.
2. Rajeev Alur and Thomas A. Henzinger. Finitary fairness. *ACM Trans. Program. Lang. Syst.*, 20(6):1171–1194, 1998.
3. Henrik Björklund and Sergei G. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155:210–229, 2007.
4. Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin. Synthesis from LTL specifications with mean-payoff objectives. In *Proc. of TACAS*, LNCS 7795, pages 169–184. Springer, 2013.
5. Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011.
6. Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In *Proc. of STACS*, LIPIcs 25, pages 199–213. Schloss Dagstuhl - LZI, 2014.
7. Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In *Proc. of EMSOFT*, LNCS 2855, pages 117–133. Springer, 2003.
8. Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.
9. Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Generalized mean-payoff and energy games. In *Proc. of FSTTCS*, LIPIcs 8, pages 505–516. Schloss Dagstuhl - LZI, 2010.
10. Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. In *Proc. of ATVA*, LNCS 8172, pages 118–132. Springer, 2013.
11. Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. *Information and Computation*, 242:25–52, 2015.
12. Krishnendu Chatterjee and Nathanaël Fijalkow. Infinite-state games with finitary conditions. In *In Proc. of CSL*, LIPIcs 8, pages 181–196. Schloss Dagstuhl - LZI, 2013.
13. Krishnendu Chatterjee and Monika Henzinger. Efficient and dynamic algorithms for alternating Büchi games and maximal end-component decomposition. *J. ACM*, 61(3):15, 2014.
14. Krishnendu Chatterjee, Monika Henzinger, Sebastian Krinninger, Veronika Loitzenbauer, and Michael A. Raskin. Approximating the minimum cycle mean. *Theor. Comput. Sci.*, 547:104–116, 2014.
15. Krishnendu Chatterjee, Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Polynomial-time algorithms for energy games with special weight structures. *Algorithmica*, 70(3):457–492, 2014.
16. Krishnendu Chatterjee and Thomas A. Henzinger. Finitary winning in omega-regular games. In *Proc. of TACAS*, LNCS 3920, pages 257–271. Springer, 2006.
17. Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1):1–27, 2009.
18. Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3-4):129–163, 2014.
19. Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset nets between decidability and undecidability. In *Proc. of ICALP*, LNCS 1443, pages 103–115. Springer, 1998.
20. Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *Int. Journal of Game Theory*, 8(2):109–113, 1979.
21. E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. of FOCS*, pages 368–377. IEEE Computer Society, 1991.
22. Nathanaël Fijalkow and Florian Horn. The surprizing complexity of generalized reachability games. *CoRR*, abs/1010.2420:1–15, 2010.
23. Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.
24. Thomas Gawlitza and Helmut Seidl. Games through nested fixpoints. In *Proc. of CAV*, LNCS 5643, pages 291–305. Springer, 2009.
25. Hugo Gimbert and Wieslaw Zielonka. When can you play positionally? In *Proc. of MFCS*, LNCS 3153, pages 686–697. Springer, 2004.
26. Vladimir A. Gurvich, Alexander V. Karzanov, and L.G. Khachivan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *USSR Computational Mathematics and Mathematical Physics*, 28(5):85–91, 1988.
27. Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.
28. Marcin Jurdziński, Jeremy Sproston, and François Laroussinie. Model checking probabilistic timed automata with one or two clocks. *Logical Methods in Computer Science*, 4(3):1–28, 2008.
29. Richard M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 1978.
30. Alexander V. Karzanov and Vasilij N. Lebedev. Cyclical games with prohibitions. *Math. Program.*, 60:277–293, 1993.

31. Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.
32. Ranko Lazic, Tom Newcomb, Joël Ouaknine, A. W. Roscoe, and James Worrell. Nets with tokens which carry data. *Fundam. Inform.*, 88(3):251–274, 2008.
33. Yuri M. Lifshits and Dmitri S. Pavlov. Potential theory for mean payoff games. *Journal of Mathematical Sciences*, 145(3):4967–4974, 2007.
34. Thomas M. Liggett. and Steven A. Lippman. Stochastic games with perfect information and time average payoff. *Siam Review*, 11(4):604–607, 1969.
35. Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.
36. Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *The Annals of Mathematics*, 74(3):437–455, 1961.
37. Nicolai N. Pisaruk. Mean cost cyclical games. *Mathematics of Operations Research*, 24(4):817–828, 1999.
38. Philippe Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Inf. Process. Lett.*, 83(5):251–261, 2002.
39. Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Rabinovich, and Jean-François Raskin. The complexity of multi-mean-payoff and multi-energy games. *CoRR*, abs/1209.3234:1–27, 2012.
40. Yaron Velner and Alexander Rabinovich. Church synthesis problem for noisy input. In *Proc. of FOSSACS*, LNCS 6604, pages 275–289. Springer, 2011.
41. Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.