Window parity games: an alternative approach toward parity games with time bounds*

Véronique Bruyère[1], Quentin Hautem[1], and Mickael Randour[1,2]

[1] Computer Science Department, Université de Mons (UMONS), Belgium
[2] Computer Science Department, Université libre de Bruxelles (ULB), Belgium

Abstract. Classical objectives in two-player zero-sum games played on graphs often deal with limit behaviors of infinite plays: e.g., *mean-payoff* and *total-payoff* in the quantitative setting, or *parity* in the qualitative one (a canonical way to encode ω-regular properties). Those objectives offer powerful abstraction mechanisms and often yield nice properties such as memoryless determinacy. However, their very nature provides no guarantee on time bounds within which something good can be witnessed. In this work, we consider two approaches toward inclusion of *time bounds* in parity games. The first one, *parity-response* games, is based on the notion of finitary parity games [9] and parity games with costs [18,32]. The second one, *window parity* games, is inspired by window mean-payoff games [6]. We compare the two approaches and show that while they prove to be equivalent in some contexts, window parity games offer a more tractable alternative when the time bound is given as a parameter (P-c. vs. PSPACE-c.). In particular, it provides a conservative approximation of parity games computable in polynomial time. Furthermore, we extend both approaches to the multi-dimension setting. We give the full picture for both types of games with regard to complexity and memory bounds.

1 Introduction

Games on graphs. Two-player games played on directed graphs constitute an important framework for the synthesis of a suitable controller for a reactive system faced to an uncontrollable environment [28]. In this setting, *vertices* of the graph represent states of the system and edges represent transitions between those states. We consider *turn-based two-player* games: each vertex either belongs to the system (the first player, denoted by $\mathcal{P}_1$) or the environment (the second player, denoted by $\mathcal{P}_2$). A game is played by moving an imaginary pebble from vertex to vertex according to existing transitions: the owner of a vertex decides where to move the pebble. The outcome of the game is an infinite sequence of vertices called *play*. The choices of both players depend on their respective *strategy* which can use an arbitrary amount of memory in full generality. In the classical setting, $\mathcal{P}_1$ tries to achieve an *objective* (describing a set of *winning plays*) while $\mathcal{P}_2$ tries to prevent him from succeeding: hence, our games are *zero-sum*. As all the objectives considered in this paper define Borel sets, Martin's theorem [26] guarantees determinacy.

Parity games. Two-player games with ω-regular objectives have been studied extensively in the literature. See for example [30,19] for an introduction. A canonical way to represent games with ω-regular conditions is the class of *parity games*: vertices are assigned a non-negative integer *priority* (or *color*), and the objective asks that among the vertices that are seen infinitely often along a play, the minimal priority be even. Parity games have been under close scrutiny for a long time both due to their importance (e.g., they subsume modal μ-calculus model checking [16]) and their intriguing complexity: they belong to the class of problems in $\mathsf{UP} \cap \mathsf{coUP}$ [22] and despite many efforts (e.g., [33,23,24,29]), whether they belong to P is still an open question. Furthermore, parity games enjoy memoryless determinacy [14,33]. Multi-dimension parity games were studied in [11]: in such games, n-dimension vectors of priorities are associated to each vertex, and the objective is to satisfy the conjunction of all the one-dimension parity objectives. The complexity of solving those games is higher: deciding if $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) has a winning strategy is coNP-complete (resp. NP-complete) and exponential memory is needed for $\mathcal{P}_1$ whereas $\mathcal{P}_2$ remains memoryless [13,20]

* Q. Hautem is supported by a FRIA fellowship, M. Randour is an F.R.S.-FNRS Postdoctoral researcher.

Time bounds. In its classical formulation, the parity objective essentially requires that *for each odd priority seen infinitely often, a smaller even priority should also be seen infinitely often.* An odd priority can be seen as a stimulus that must be answered by seeing a smaller even priority. The parity objective has fundamental qualities. The simplicity of its definition totally abstracts timing issues like "how much time has elapsed between a stimulus and its answer" and is key to memoryless determinacy. This makes it robust to slight changes in the model which could impact more precise formulations (e.g., counting the number of steps between a stimulus and its answer critically depends on the granularity of the game graph).

Nonetheless, it has been recently argued that in a large number of practical applications, *timing does matter* (e.g., [9,25,6]). Indeed, in general it does not suffice to know that a "good behavior" will *eventually* happen, and one wants to ensure that it can actually be witnessed *within a time frame which is acceptable* with regard to the modeled reactive system. For example, consider a computer server having to grant requests to clients. A classical parity objective can encode that requests should eventually be granted. However, it is clear that in a desired controller, requests should not be placed on hold for an arbitrarily long time. In order to accomodate such requirements, various attempts to associate classical game objectives with time bounds have been recently studied. For example, *window mean-payoff* and *window total-payoff* games provide a framework to reason about quantitative games (e.g., modeling quantities such as energy consumption) with time bounds [6]. In the qualitative setting, *finitary parity* games [9,7] and *parity games with costs* [18,32] provide a similar framework for parity games.

Two approaches. While window games and finitary parity games (resp. parity games with costs) share the goal of allowing precise specification of time bounds, their inner mechanisms differ. The aim of our work is three-fold: (*i*) apply the **window mechanism to parity games**, (*ii*) provide a **thorough comparison** with the existing framework of finitary parity games and parity games with costs, (*iii*) **extend both approaches to the multi-dimension setting** (which was left unexplored up to now). Since all those related papers do not use a uniform terminology, we here use the following taxonomy for the two approaches.

- **Window parity (WP).** Intuitively, the *direct fixed WP* objective considers a window of size bounded by $\lambda \in \mathbb{N}_0$ (given as a parameter) sliding over an infinite play and declare this play winning if in all positions, the window is such that the minimal priority within it is even. For *direct bounded WP*, the size of the window is not fixed as a parameter but a play is winning if there exists a bound λ for which the condition holds. We also consider the *fixed WP* and *bounded WP* objectives which are essentially prefix-independent variants of the previous ones. All those objectives are based on the window mechanism introduced in [6] and our work presents the first implementation of this mechanism for parity games.
- **Parity-response (PR).** The *direct fixed PR* objective asks that along a play, any odd priority be followed by a smaller even priority in at most $\lambda \in \mathbb{N}_0$ (given as a parameter) steps. As for the WP setting, we also consider the *direct bounded PR* objective where a play is winning if there exists a bound λ such that the condition holds, along with the respective prefix-independent variants: the *fixed PR* and the *bounded PR* objectives. The *bounded PR* objective was studied for one-dimension games (under the name *finitary parity*) in [9]: deciding the winner is in P and memoryless strategies suffice for $\mathcal{P}_1$ while $\mathcal{P}_2$ needs infinite memory. The *fixed PR* objective for one-dimension games was very recently proved to be PSPACE-complete, with exponential memory bounds for both players [32] (this work is presented in the more general context of *parity games with costs*). Our work provides the first study of the parity-response approach in multi-dimension games.

Our contributions. Given the number of variants studied, we give an overview of our results in Table 1. Our main contributions are as follows.

1. We prove that *bounded WP* and *bounded PR* objectives coincide, even in multi-dimension games (Proposition 7).
2. We establish that *bounded WP* (and thus *bounded PR*) games are P-hard in one-dimension (Theorem 10, P-membership follows from [9]) and that they are $\mathsf{EXPTIME}$-complete in multi-dimension (Theorem 16). The $\mathsf{EXPTIME}$-membership follows from a reduction to a variant of *request-response games* [31] presented

	one-dimension			multi-dimension		
	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.
Fixed WP	**P-c.**	**polynomial**		**EXPTIME-c.**	**exponential**	
Fixed PR	PSPACE-c.	exponential	$\leq$ exponential			
Bounded WP	P-c.	memoryless	infinite		**exponential**	**infinite**
Bounded PR						

Table 1. Complexity of deciding the winner and memory required for winning strategies in window parity (WP) and parity-response (PR) games. All results hold for both the *prefix-independent* and the *direct* (Dir) variants of all the objectives, except for the memory of $\mathcal{P}_2$ in the direct bounded cases: in one-dimension games, linear memory is both sufficient and necessary (for both WP and PR) and in multi-dimension games, exponential memory is both sufficient and necessary. All bounds are tight unless specified by the $\leq$ symbol. New results are in bold.

in [9] under the name of *finitary Streett games* (Lemma 18). The EXPTIME-hardness is proved via a reduction from the membership problem in alternating polynomial-space Turing machines (Lemma 20).

3. We show that in multi-dimension *bounded WP* (and thus *bounded PR*) games, exponential memory is both sufficient and necessary for $\mathcal{P}_1$ while infinite memory is needed for $\mathcal{P}_2$ (Theorem 16).
4. We prove that one-dimension *fixed WP* games provide a **conservative approximation of parity games** (Proposition 7) **computable in polynomial time** (Theorem 14). This is in contrast to the PSPACE-completeness of *fixed PR* games [32] (actually, the proof in [32] is for a more general model but already holds for *fixed PR* games).
5. While *fixed PR* games are PSPACE-complete, we establish two polynomial-time algorithms (Theorem 12) to solve fixed-parameter sub-cases: (i) the bound λ is fixed, or (ii) the number of priorities is fixed.
6. In multi-dimension, we prove that both *fixed PR* (Theorem 23) and *fixed WP* (Theorem 24) games are EXPTIME-complete. Membership relies on different techniques and algorithms for each case while hardness is based on the same reduction as for the *bounded* variants (Lemma 20).
7. In one-dimension games, we also establish that for *fixed WP*, polynomial memory is both sufficient and necessary for both players, whereas exponential memory is required for *fixed PR* [32]. In multi-dimension games, we prove that for both *fixed PR* and *fixed WP*, exponential memory is both sufficient and necessary for both players. The upper bounds follow from the EXPTIME algorithms mentioned above whereas the lower bounds in one-dimension are shown thanks to appropriate families of games (Example 15) and in multi-dimension are obtained through reduction from *generalized reachability games* [17] (Lemma 21).
8. We establish the existence of values of λ such that the *fixed* objectives become equivalent to the *bounded* ones, both in one-dimension (Theorem 10) and in multi-dimension (Theorem 16).

While all the aforementioned results are for the *prefix-independent* variants of our objectives, we also obtain closely related complexities and memory bounds for the *direct* ones (Table 1). We obtain our results using a variety of techniques, sometimes inspired by [9,6]. Our focus is on giving the full picture for the two approaches toward including time bounds in parity games: window parity and parity-response. We sum up the key comparison points in the next paragraph.

Comparison. The *parity-response* and *window parity* approaches turn out to be equivalent in the *bounded* context, i.e., when the question is the existence of a bound $\lambda \in \mathbb{N}_0$ for which the corresponding *fixed* objective holds. Hence, the focus of the comparison is the *fixed* variants. Observe that those variants are of interest for applications where the time bound is part of the specification: parameter λ grants flexibility in the specification as it can be adjusted to specific requirements of the application. Let us review the complexities of the *fixed PR* and *fixed WP* objectives.

In one-dimension games, *fixed PR* is PSPACE-complete whereas *fixed WP provides a framework with similar flavor that enjoys increased tractability*: it is P-complete. Hence, *fixed WP does provide a polynomial-time conservative approximation of parity games* (Proposition 7). Interestingly, the *fixed WP* objective also

permits to approximate the *fixed PR* one in both directions, and in polynomial time: we prove in Proposition 7 that the *fixed PR* objective for time bound λ can be framed by the *fixed WP* objective for two well-chosen values of the time bound λ' and λ''.

In multi-dimension, both *fixed PR* and *fixed WP* games are EXPTIME-complete. Nonetheless, while the algorithm for *fixed PR* requires exponential time in *both* the number of dimensions *and* the number of priorities (which can be as large as the game graph), *solving the fixed WP case only requires exponential time in the number of dimensions*. This distinction may have *impact on practical applications* where, usually, the size of the model (hence the game graph) can be very large while the specification (hence the number of dimensions) is comparatively small. Note that for both objectives, the multi-dimension algorithms are *pseudo*-polynomial in the time bound λ, hence also exponential in the length of its binary encoding.

Finally, let us compare *window parity* games with *window mean-payoff (WMP)* games [6]. First, one could naturally wonder if WP games could be solved by encoding them into WMP games, following a reduction similar in spirit to the one developed by Jurdzinski for classical parity games [22]. This is indeed possible, but leads to increased complexities in comparison to the ad hoc analysis developed in this work. For example, multi-dimension *fixed WP* games would require exponential time in the number of priorities too. Second, observe that *fixed WP* games can be solved in polynomial time whatever the bound $\lambda \in \mathbb{N}_0$ whereas *fixed WMP* games require pseudo-polynomial time, i.e., also polynomial in the bound λ. Finally, multi-dimension *bounded WMP* games are known to be non-primitive-recursive-hard and their decidability is still open [6]. On the contrary, multi-dimension *bounded WP* games are EXPTIME-complete. This suggests that the colossal complexity of *bounded WMP* games is a result of the quantitative nature of mean-payoff mixed with windows, and not an inherent drawback of the window mechanism.

Other related work. This paper extends its preceding conference version [3]. In addition to the aforementioned articles, we mention two papers where logical formalisms dealing with time bounds are studied. In [25], Kupferman et al. introduced Prompt-LTL, which is strongly linked with the finitary conditions discussed above. In [1], Baier et al. also studied an extension of LTL that can express properties based on the window mechanism of [6]. The study of logical fragments corresponding to our framework of window parity games is an interesting question left open for future work.

Outline. Section 2 presents the needed definitions and known results about classical objectives. Section 3 introduces the different objectives studied in this paper and establishes the links between them. Section 4 and Section 5 respectively present our results for one-dimension and multi-dimension games.

2 Preliminaries

Game structures. We consider zero-sum turn-based games played by two players, $\mathcal{P}_1$ and $\mathcal{P}_2$, on a finite directed graph.

Definition 1. *A* game structure *is a tuple* $G = (V_1, V_2, E)$ *where*

- (V, E) *is a finite directed graph, with* $V = V_1 \cup V_2$ *the set of vertices and* $E \subseteq V \times V$ *the set of edges such that for each* $v \in V$*, there exists* $(v, v') \in E$ *for some* $v' \in V$ *(no deadlock),*
- (V_1, V_2) *forms a partition of* V *such that* V_i *is the set of vertices controlled by player* $\mathcal{P}_i$ *with* $i \in \{1, 2\}$*.*

A *play* of G is an infinite sequence of vertices $\pi = v_0 v_1 \ldots \in V^\omega$ such that $(v_k, v_{k+1}) \in E$ for all $k \in \mathbb{N}$. We denote by $\mathsf{Plays}(G)$ the set of plays in G. *Histories* of G are finite sequences $\rho = v_0 \ldots v_k \in V^*$ defined in the same way. Given a play $\pi = v_0 v_1 \ldots$, the history $v_k \ldots v_{k+l}$ is denoted by $\pi[k, k+l]$; in particular, $v_k = \pi[k]$. We also use notation $\pi[k, \infty]$ for the suffix $v_k v_{k+1} \ldots$ of π.

Strategies. A *strategy* σ_i for $\mathcal{P}_i$ is a function $\sigma_i \colon V^*V_i \to V$ assigning to each history $\rho v \in V^*V_i$ a vertex $v' = \sigma_i(\rho v)$ such that $(v, v') \in E$. It is *memoryless* if $\sigma_i(\rho v) = \sigma_i(\rho' v)$ for all histories $\rho v, \rho' v$ ending with the same vertex v, that is, if σ_i is a function $\sigma_i \colon V_i \to V$. It is *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, $\alpha_u \colon M \times V \to M$ is the update function, and $\alpha_n \colon M \times V_i \to V$ is the next-action function. The Moore machine defines a strategy σ_i such that $\sigma_i(\rho v) = \alpha_n(\widehat{\alpha}_u(m_0, \rho), v)$ for all histories $\rho v \in V^*V_i$, where $\widehat{\alpha}_u$ extends α_u to sequences of vertices as expected. The *size* of the strategy is the size $|M|$ of its Moore machine. Note that a strategy is memoryless when $|M| = 1$.

Given a strategy σ_i of $\mathcal{P}_i$, we say that a play $\pi = v_0 v_1 \dots$ of G is *consistent* with σ_i if $v_{k+1} = \sigma_i(v_0 \dots v_k)$ for all $k \in \mathbb{N}$ such that $v_k \in V_i$. Consistency is naturally extended to histories in a similar fashion. Given an *initial vertex* v_0, and a strategy σ_i of each player $\mathcal{P}_i$, we have a unique play consistent with both strategies. This play is called the *outcome* of the game and is denoted by $\mathsf{Out}(v_0, \sigma_1, \sigma_2)$.

Objectives and winning sets. Let $G = (V_1, V_2, E)$ be a game structure. An *objective for* $\mathcal{P}_1$ is a set of plays $\Omega \subseteq \mathsf{Plays}(G)$. A play π is *winning* for $\mathcal{P}_1$ if $\pi \in \Omega$, and losing otherwise (i.e., winning for $\mathcal{P}_2$). We thus consider *zero-sum* games such that the objective of player $\mathcal{P}_2$ is $\overline{\Omega} = \mathsf{Plays}(G) \setminus \Omega$. In the following, we always take the point of view of $\mathcal{P}_1$ by assuming that Ω is his objective, and we denote by (G, Ω) the corresponding *game*. Given an initial vertex v_0 of a game (G, Ω), a strategy σ_1 for $\mathcal{P}_1$ is *winning* from v_0 if $\mathsf{Out}(v_0, \sigma_1, \sigma_2) \in \Omega$ for all strategies σ_2 of $\mathcal{P}_2$. Vertex v_0 is also called *winning* for $\mathcal{P}_1$ and the *winning set* $\mathsf{Win}_1^G(\Omega)$ is the set of all his winning vertices. Similarly the winning vertices of $\mathcal{P}_2$ are those from which $\mathcal{P}_2$ can ensure to satisfy his objective $\overline{\Omega}$ against all strategies of $\mathcal{P}_1$, and $\mathsf{Win}_2^G(\overline{\Omega})$ is his winning set. If $\mathsf{Win}_1^G(\Omega) \cup \mathsf{Win}_2^G(\overline{\Omega}) = V$, we say that the game is *determined*. It is known that every turn-based game with a Borel objective is determined [26]. This in particular applies to the objectives studied in this paper.

Decision problem. Given a game (G, Ω) and an initial vertex v_0, we want to decide whether $\mathcal{P}_1$ has a winning strategy from v_0 for the objective Ω or not (in which case, $\mathcal{P}_2$ has one for $\overline{\Omega}$). We want to study the complexity class of this decision problem as well as the memory requirements of winning strategies of both players. In this paper, we focus on several variants of the parity objective and we consider two settings: the *one-dimension* case with one objective Ω and the *multi-dimension* case with the intersection of several objectives $\cap_{m=1}^{n} \Omega_m$.

Parity objective. Let G be a game structure. Let π be a play, we define $\mathsf{Inf}(\pi)$ as the set of vertices seen infinitely often in π. Formally, $\mathsf{Inf}(\pi) = \{v \in V \mid \forall k \geq 0, \exists l \geq k, \pi[l] = v\}$. Given a *priority function* $p \colon V \to \{0, 1, \dots, d\}$ that maps every vertex to an integer priority where d is even and $d \leq |V| + 1$ (w.l.o.g.), the *parity objective* $\mathsf{Parity}(p)$ asks that of the vertices that are visited infinitely often, the smallest priority be even. Formally, the parity objective is defined as

$$\mathsf{Parity}(p) = \{\pi \in \mathsf{Plays}(G) \mid \min_{v \in \mathsf{Inf}(\pi)} p(v) \text{ is even}\}.$$

As smallest even priorities have a specific role in parity objectives, we define a partial order $\preceq$ on priorities as follows. For $c, c' \in \{0, \dots, d\}$, we have $c \preceq c'$ if and only if c is even and $c \leq c'$. In this case we say that c is $\preceq$-smaller than c'.

Deciding the winner in parity games is known to be in $\mathsf{UP} \cap \mathsf{coUP}$ [22] and whether a polynomial-time algorithm exists is a long-standing open question (e.g., [33,23,24,29]). Memoryless strategies suffice for both players to win in parity games [14,33]. The multi-dimension case is studied in [11], with n different priority functions p_m, $m \in \{1, \dots, n\}$, and objective $\cap_{m=1}^{n} \Omega_m$ such that each Ω_m is the parity objective defined for p_m. Deciding if $\mathcal{P}_1$ wins in those games is coNP-complete, exponential-memory strategies are necessary and sufficient for $\mathcal{P}_1$, and memoryless strategies suffice for $\mathcal{P}_2$ [4,13,20,27].

Other useful objectives. We recall some useful results for several classical objectives. Let G be a game structure and $U \subseteq V$ be a set of vertices. A *reachability* objective $\mathsf{Reach}(U)$ asks to visit a vertex of U at least once, whereas a *safety* objective $\mathsf{Safe}(U)$ asks to visit no vertex of $V \setminus U$. Deciding the winner in reachability

games and safety games is known to be P-complete with an algorithm in time $O(|V|+|E|)$, and memoryless winning strategies suffice for both objectives and both players [2,19,21]. A *Büchi* objective $\mathsf{Buchi}(U)$ asks to visit a vertex of U infinitely often, whereas a *co-Büchi* objective $\mathsf{CoBuchi}(U)$ asks to visit no vertex of $V \setminus U$ infinitely often. Deciding the winner in Büchi games and co-Büchi games is also P-complete with an algorithm in time $O(|V|^2)$, and memoryless strategies also suffice for both objectives and both players [8,15,19]. Finally, given $U_1, \dots, U_n$ a family of n subsets of V, a *generalized reachability* objective $\mathsf{GenReach}(U_1, ..., U_n) = \cap_{m=1}^{n} \mathsf{Reach}(U_m)$ asks to visit a vertex of U_m at least once, for each $m \in \{1, \dots, n\}$. Deciding the winner in generalized reachability games is PSPACE-complete and exponential-memory strategies are necessary and sufficient for both players [17].

3 Adding time bounds to parity games

In this section we introduce the two approaches discussed in this paper: *window parity* (WP) and *parity-response* (PR) games. In Section 3.1, we formally define the related objectives. Then, Section 3.2 presents an interesting decomposition of winning plays for the WP objective that will prove to be a useful mechanism to establish solving algorithms. Finally, in Section 3.3, we establish inclusions and equivalences between several variants of the WP and PR objectives.

3.1 Window parity and parity-response objectives

As stated in Section 1, the intuition for both approaches is as follows. The parity-response objective asks that every priority be followed by a $\preceq$-smaller priority in a bounded number of steps. In a window parity game, a *window* with a bounded size is sliding along the play, and one asks to find a $\preceq$-smallest[3] priority inside this window, and this for all positions along the play. We derive four variants for each of these objectives, according to whether the bound is given as a parameter or not (*fixed* or *bounded* variant), and whether the objective must be satisfied directly or eventually (*direct* or *undirect* variant). The *undirect* variants are thus prefix-independent.[4] Formally:

Definition 2. *Given a game structure $G = (V_1, V_2, E)$, a priority function $p\colon V \to \{0, 1, \dots, d\}$, and a bound $\lambda \in \mathbb{N}_0$, we define the eight following objectives:*

$$\mathsf{DirFixPR}(\lambda, p) = \{\pi \in \mathsf{Plays}(G) \mid \forall\, j \geq 0,\ \exists\, l \in \{0, \dots, \lambda - 1\},\ p(\pi[j + l]) \preceq p(\pi[j])\},$$
$$\mathsf{DirFixWP}(\lambda, p) = \{\pi \in \mathsf{Plays}(G) \mid \forall\, j \geq 0,\ \exists\, l \in \{0, \dots, \lambda - 1\},\ \forall\, k \in \{0, \dots, l\},\ p(\pi[j + l]) \preceq p(\pi[j + k])\},$$

and given $\mathsf{X} \in \{\mathsf{PR}, \mathsf{WP}\}$,

$$\mathsf{FixX}(\lambda, p) = \{\pi \in \mathsf{Plays}(G) \mid \exists\, i \geq 0,\ \pi[i, \infty] \in \mathsf{DirFixX}(\lambda, p)\},$$
$$\mathsf{DirBndX}(p) = \{\pi \in \mathsf{Plays}(G) \mid \exists\, \lambda \in \mathbb{N}_0,\ \pi \in \mathsf{DirFixX}(\lambda, p)\},$$
$$\mathsf{BndX}(p) = \{\pi \in \mathsf{Plays}(G) \mid \exists\, i \geq 0,\ \pi[i, \infty] \in \mathsf{DirBndX}(p)\}.$$

Thus, in the direct fixed parity-response objective $\mathsf{DirFixPR}(\lambda, p)$, for all positions $j \geq 0$, the priority $p(\pi[j])$ must be followed by a $\preceq$-smaller priority $p(\pi[j + l])$ within at most $\lambda - 1$ steps. The undirect fixed variant $\mathsf{FixPR}(\lambda, p)$ asks this objective to be satisfied eventually (i.e., for all positions $j \geq i$, for some i). The direct bounded variant $\mathsf{DirBndPR}(p)$ (resp. the undirect bounded variant $\mathsf{BndPR}(p)$) asks for the existence of a bound λ for which $\mathsf{DirFixPR}(\lambda, p)$ (resp. $\mathsf{FixPR}(\lambda, p)$) is satisfied.

In the case of window parity objectives, we rather call λ the *window size*. Given a play $\pi = v_0 v_1 \dots$, a *λ-window at position j* is a window of size λ placed along π from position j to $j + \lambda - 1$. The direct fixed window parity objective $\mathsf{DirFixWP}(\lambda, p)$ asks that for all $j \geq 0$, inside the λ-window at position j, one can find a priority $p(\pi[j + l])$ with $l \leq \lambda - 1$ that is the $\preceq$-smallest one in $\pi[j, j + l]$. When the window size λ is clear from the context, we drop the prefix λ and simply talk about windows instead of λ-windows.

[3] Notice the difference: smallest vs. smaller.

[4] An objective Ω is *prefix-independent* if for any play $\pi = \rho\pi'$, it holds that $\pi \in \Omega \iff \pi' \in \Omega$.

Example 3. We illustrate the previous definitions on a simple example of one-player game, where all vertices belong to $\mathcal{P}_1$ (see Figure 1). In this example and in the sequel, the priority $p(v)$ is always put under vertex v, and circle (resp. square) vertices all belong to $\mathcal{P}_1$ (resp. $\mathcal{P}_2$). In the game of Figure 1, there is a unique play

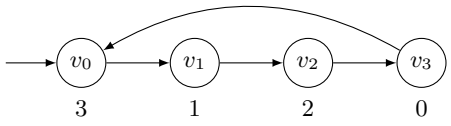

Fig. 1. A simple example of one-player-game: all vertices belong to $\mathcal{P}_1$.

from the initial vertex v_0 equal to $\pi = (v_0v_1v_2v_3)^\omega$. On the one hand, we have that $\pi \in \mathsf{DirFixPR}(\lambda, p)$ for $\lambda = 3$. Indeed, the odd priority 3 (resp. 1) is followed by the even priority 2 (resp. 0) in exactly $\lambda - 1 = 2$ steps, whereas the even priority 2 (resp. 0) is "followed" by itself in 0 steps. Similarly, π also belongs to the three variants $\mathsf{FixPR}(\lambda, p)$, $\mathsf{DirBndPR}(p)$ and $\mathsf{BndPR}(p)$. On the other hand, $\pi \notin \mathsf{DirFixWP}(3, p)$. Indeed, in the 3-window at position 0, there is no $l \in \{0, 1, 2\}$ such that $p(v_l)$ is the $\preceq$-smallest priority in $\pi[0, l]$ because $p(v_0)$ and $p(v_1)$ are odd, and $p(v_2)$ is even but $p(v_2) = 2 \not\preceq 1 = p(v_1)$. However, one can check that $\pi \in \mathsf{DirFixWP}(4, p)$, and it also belongs to $\mathsf{FixWP}(4, p)$, $\mathsf{DirBndWP}(p)$ and $\mathsf{BndWP}(p)$. ◁

Links with finitary and window objectives. Before continuing, we clarify the relation between the objectives introduced in Definition 2 and the objectives studied in [9,6,32].

Remark 4. In [9], Chatterjee et al. study the *bounded parity* (resp. *finitary parity*) objective which asks for the existence of a bound λ such that for all positions $j \geq 0$ (resp. $j \geq i$, for some i) with an odd priority $p(\pi[j])$, the *minimum* number of steps l to see a $\preceq$-smaller priority $p(\pi[j + l])$ does not exceed λ. These two objectives are respectively equal to the $\mathsf{DirBndPR}(p)$ and $\mathsf{BndPR}(p)$ objectives. Indeed asking for a minimum number of steps is not a restriction with respect to our definition, and if $p(\pi[j])$ is even then with $l = 0$ we trivially have $p(\pi[j + l]) \preceq p(\pi[j])$. The $\mathsf{FixPR}(\lambda, p)$ objective can be seen as a particular case of the problem studied in [32], namely the synthesis of so-called optimal strategies in *parity games with costs*, hence $\mathsf{FixPR}(\lambda, p)$ games are in PSPACE. Careful inspection of [32] reveals that the PSPACE-membership also holds for the direct variant $\mathsf{DirFixPR}(\lambda, p)$, and that the PSPACE-hardness proof can easily be adapted to our sub-cases ($\mathsf{FixPR}(\lambda, p)$ and $\mathsf{DirFixPR}(\lambda, p)$).

Remark 5. The WP objective and its variants are inspired by the *window mean-payoff* objective and its variants introduced in [6]. In that paper, the classical mean-payoff objective, that requires the average weight to be non-negative in the long run (i.e., at the limit), is replaced by the need for a non-negative average weight inside every bounded window along the play. Here the classical parity objective $\mathsf{Parity}(p)$, asking the smallest priority seen infinitely often to be even, is replaced by asking the smallest priority to be even in every bounded window.

3.2 Decomposition of plays for the window parity objective

We introduce additional terminology concerning the WP objective. Let $\pi = v_0v_1 \dots$ be a play and consider an *unbounded* window at position j. Essentially, it represents the suffix $\pi[j, \infty]$. We say that this window is *closed* in position $j + k$, for $k \geq 0$, if there exists $l \in \{0, \dots, k\}$ such that $p(\pi[j + l])$ is the $\preceq$-smallest priority in $\pi[j, j + l]$. For the *smallest* such l, we say that the window at position j *closes* in $j + l$. Observe that if $p(\pi[j])$ is even, the window at position j closes immediately. On the opposite, the unbounded window at position j is still *open* in position $j + k$ if no such l exists. Observe that a closed window stays closed forever, and a window can stay open forever if no $\preceq$-smallest priority is ever found in prefixes of $\pi[j, \infty]$.

Now, for the objectives presented in Definition 2, we are especially interested in windows of bounded size. Let $\lambda \in \mathbb{N}_0$. We say that the window at position j is λ-*good* if it closes in a position $j + l$ such that

$l < \lambda$, that is, if it closes in at most $(\lambda - 1)$ steps.[5] If this is the case, then we also say that the history $\pi[j, j+l]$ is λ-good. On the contrary, we say that the window at position j is λ-*bad* if it is still open in position $j + (\lambda - 1)$.[6]

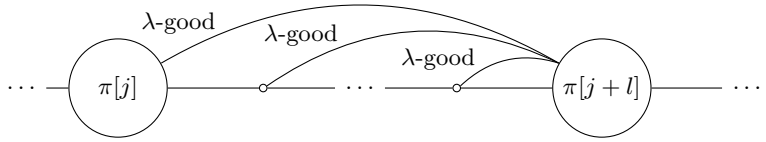

Fig. 2. History $\pi[j, j+l]$ is λ-good implies that for all $j' \in \{j+1, \ldots, j+l\}$, $\pi[j', j+l]$ is also λ-good.

An interesting property is the following one: if $\pi[j, j+l]$ is λ-good, then $\pi[j', j+l]$ is also λ-good for all $j' \in \{j+1, \ldots, j+l\}$ (see Figure 2). In particular, each window at position $j' \in \{j+1, \ldots, j+l\}$ is closed in position $j + l$.

The next lemma will prove useful later on. Intuitively, it states that any play that is winning for a WP objective can be seen as a succession of λ-good histories which can serve as the basis for a corresponding decomposition.

Lemma 6. *A play π belongs to* $\mathsf{FixWP}(\lambda, p)$ *(resp.* $\mathsf{DirFixWP}(\lambda, p)$*) if and only if there exists an increasing sequence of indices $(k_i)_{i \geq 0}$ (resp. with $k_0 = 0$) such that for each i, $\pi[k_i, k_{i+1} - 1]$ is λ-good.*

When such a sequence $(k_i)_{i \geq 0}$ with $k_0 = 0$ exists for a play π, we say that it is a λ-*good decomposition* of π. When $k_0 \geq 0$, it is called an *eventually λ-good decomposition*.

Proof (of Lemma 6). We only give the proof for the direct variant. Suppose that π belongs to $\mathsf{DirFixWP}(\lambda, p)$. Then by definition of $\mathsf{DirFixWP}(\lambda, p)$, the window at position $k_0 = 0$ is closed in at most $(\lambda - 1)$ steps, i.e., there exists $k_1 > k_0$ such that $\pi[k_0, k_1 - 1]$ is λ-good. Next, as the window at position k_1 is closed in at most $(\lambda - 1)$ steps, there exists $k_2 > k_1$ such that $\pi[k_1, k_2 - 1]$ is λ-good also. This leads to a λ-good decomposition $(k_i)_{i \geq 0}$ of π.

Conversely, if there exists a λ-good decomposition $(k_i)_{i \geq 0}$ of π, then for all $i \geq 0$, $\pi[k_i, k_{i+1} - 1]$ is λ-good, as well as $\pi[k', k_{i+1} - 1]$ for all $k' \in \{k_i + 1, \ldots, k_{i+1} - 1\}$. It follows that π belongs to $\mathsf{DirFixWP}(\lambda, p)$. □

3.3 Relationship between objectives

We now detail the inclusions and equalities between the various objectives introduced in Definition 2 as well as with the parity objective.

Proposition 7. *Let $G = (V_1, V_2, E)$ be a game structure and p be a priority function. Let $\lambda \in \mathbb{N}_0$.*

1. *For all* $\mathsf{X} \in \{\mathsf{FixPR}(\lambda, p), \mathsf{FixWP}(\lambda, p), \mathsf{BndPR}(p), \mathsf{BndWP}(p)\}$, $\mathsf{DirX} \subseteq \mathsf{X}$.
2. *For all* $\mathsf{X} \in \{\mathsf{PR}, \mathsf{WP}\}$, $(\mathsf{Dir})\mathsf{FixX}(\lambda, p) \subseteq (\mathsf{Dir})\mathsf{BndX}(p)$.
3. *For all* $\lambda' > \lambda$, *for all* $\mathsf{X} \in \{\mathsf{FixPR}, \mathsf{FixWP}\}$, $(\mathsf{Dir})\mathsf{X}(\lambda, p) \subseteq (\mathsf{Dir})\mathsf{X}(\lambda', p)$.
4. $(\mathsf{Dir})\mathsf{FixWP}(\lambda, p) \subseteq (\mathsf{Dir})\mathsf{FixPR}(\lambda, p)$.
5. $(\mathsf{Dir})\mathsf{FixPR}(\lambda, p) \subseteq (\mathsf{Dir})\mathsf{FixWP}(\frac{d}{2} \cdot \lambda, p)$.
6. $(\mathsf{Dir})\mathsf{BndPR}(p) = (\mathsf{Dir})\mathsf{BndWP}(p)$.
7. $(\mathsf{Dir})\mathsf{BndWP}(p) \subseteq \mathsf{Parity}(p)$.

[5] This is equivalent to saying that the λ-window at position j is closed.
[6] This is equivalent to saying that the λ-window at position j is open.

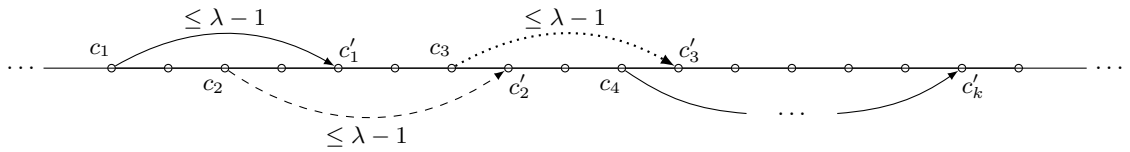

Fig. 3. Illustration of inclusion (Dir)FixPR$(\lambda, p) \subseteq$ (Dir)FixWP$(\frac{d}{2} \cdot \lambda, p)$. Priorities c_i are all odd and decreasing ($c_{i+1} < c_i$), priorities c'_i are even and such that $c'_i \preceq c_i$ but $c'_i \not\preceq c_{i+1}$. Eventually, a priority c'_k is reached such that the history from c_1 to c'_k is λ'-good for some λ'. Furthermore $k \leq \frac{d}{2}$, hence $\lambda' \leq \frac{d}{2} \cdot \lambda$.

Before proving those statements formally, we give an intuitive explanation of Item 5, as it is the most interesting one technically. Assume we have a play $\pi \in$ DirFixPR(λ, p), like the one depicted in Figure 3. Since it satisfies objective DirFixPR(λ, p), we know that each odd priority is followed by a smaller even priority in at most $(\lambda - 1)$ steps. We argue that we can give a λ'-good decomposition of this play for some $\lambda' \leq \frac{d}{2} \cdot \lambda$, hence that it belongs to DirFixWP$(\frac{d}{2} \cdot \lambda, p)$. The key idea is depicted in Figure 3. Let c_1 be an odd priority. It must be followed by a $\preceq$-smaller priority c'_1 in at most $(\lambda - 1)$ steps. If c'_1 is the minimal priority encountered from c_1 to c'_1, then we are done as the corresponding history is λ-good. Assume it is not, then there exists c_2 between c_1 and c'_1 such that c_2 is odd and $c'_1 \not\preceq c_2$. But again, c_2 must be followed by $c'_2 \preceq c_2$ in at most $(\lambda - 1)$ steps. Repeating this argument, we obtain that c_1 is followed by a priority c'_k in strictly less than $\frac{d}{2} \cdot \lambda$ steps[7] (as there are $\frac{d}{2}$ odd priorities and each of them is answered in $(\lambda - 1)$ steps) such that c'_k is even and smaller than all priorities encountered from c_1 to c'_k. Therefore, the corresponding history is λ'-good for $\lambda' \leq \frac{d}{2} \cdot \lambda$. As this argument can be repeated from the vertex following priority c'_k, we obtain a λ'-good decomposition of the play, which implies that it satisfies DirFixWP(λ', p) as claimed.

Proof. The proof is given for the direct variants only, as it is similar for the undirect variants. The first four items immediately follow from the definitions.

We now prove Item 5. Let $\pi \in$ DirFixPR(λ, p) and let us show that $\pi \in$ DirFixWP(λ', p) with $\lambda' = \frac{d}{2} \cdot \lambda$. Given $j_1 \in \mathbb{N}$, we are looking for an integer $l' \in \{0, \dots, \lambda' - 1\}$ such that $p(\pi[j_1 + l'])$ is the $\preceq$-smallest priority in $\pi[j_1, j_1 + l']$. If $p(\pi[j_1])$ is even then take $l' = 0$. Otherwise, as $\pi \in$ DirFixPR(λ, p), (*) there exists $l_1 \in \{0, \dots, \lambda - 1\}$ (that we take minimal) such that $p(\pi[j_1 + l_1]) \preceq p(\pi[j_1])$. If $p(\pi[j_1 + l_1])$ is the $\preceq$-smallest priority in $\pi[j_1, j_1 + l_1]$, then take $l' = l_1$. Otherwise, the smallest priority in $\pi[j_1, j_1 + l_1]$ is odd (by minimality of l_1) and strictly smaller than $p(\pi[j_1])$. Let j_2 be a position in $\pi[j_1, j_1 + l_1]$ of this smallest priority and repeat (*) for j_2. There now exists a minimal $l_2 \in \{0, \dots, \lambda - 1\}$ such that $p(\pi[j_2 + l_2]) \preceq p(\pi[j_2])$. Again, if $p(\pi[j_2 + l_2])$ is the $\preceq$-smallest priority in $\pi[j_2, j_2 + l_2]$, then by definition of j_2, $p(\pi[j_2 + l_2])$ is also the $\preceq$-smallest priority in $\pi[j_1, j_2 + l_2]$. Therefore we can take $l' = j_2 + l_2 - j_1$. Otherwise, the smallest priority $p(\pi[j_3])$ in $\pi[j_2, j_2 + l_2]$ is odd and strictly smaller than $p(\pi[j_2])$, and we now repeat (*) for j_3, also. As $p(\pi[j_1]), p(\pi[j_2]), p(\pi[j_3]) \dots$ is a decreasing sequence of odd priorities with $j_2 - j_1 \leq \lambda - 1$ and $j_3 - j_1 \leq 2(\lambda - 1)$, we end the process with $p(\pi[j_{\frac{d}{2}} + l_{\frac{d}{2}}]) = 0$ in the worst case. It follows that $p(\pi[j_{\frac{d}{2}} + l_{\frac{d}{2}}])$ is the $\preceq$-smallest priority in $\pi[j_1, j_{\frac{d}{2}} + l_{\frac{d}{2}}]$ and we take $l' = j_{\frac{d}{2}} + l_{\frac{d}{2}} - j_1$. Notice that $l' < \frac{d}{2} \cdot \lambda$.

Item 6 is a direct consequence of Items 4 and 5.

In order to prove Item 7, let $\pi \in$ DirBndWP(p) and suppose that $\pi \notin$ Parity(p). Thus among the vertices that are visited infinitely often, the least priority, call it c, is odd. Let $j \in \mathbb{N}$ be such that $p(\pi[j]) = c$ and every vertex in $\pi[j, \infty]$ belongs to Inf(π). It follows that for all $l \geq 0$, $p(\pi[j + l]) \not\preceq p(\pi[j])$. This is in contradiction with π belonging to DirBndWP(p) as the window at position j would never close, and therefore we conclude that $\pi \in$ Parity(p). ◻

From the inclusions $\Omega \subseteq \Omega'$ of Proposition 7, we immediately derive the inclusions $\mathsf{Win}_1^G(\Omega) \subseteq \mathsf{Win}_1^G(\Omega')$. It yields two interesting observations.

[7] Actually, $\frac{d}{2} \cdot (\lambda - 1) + 1$ but we use the simpler bound $\frac{d}{2} \cdot \lambda$ from now on for the sake of readability.

1. By Items 1, 2, 6, and 7, we see that all WP and PR variants provide conservative approximations of the classical parity objective. While the bounded variants will all be shown to be in P (Theorem 10), *for the fixed variants, the WP objective will be the only polynomial-time alternative* (Theorem 14).
2. The *fixed WP objective can actually be used to provide both an under-approximation and an over-approximation of the fixed PR one in polynomial-time* (Theorem 14), as witnessed by Items 4 and 5. This is particularly interesting since the fixed PR objective itself is PSPACE-complete (Theorem 11).

Notice that the inclusions of Proposition 7 are strict in general. This is also the case when one replaces the objectives by the winning sets of $\mathcal{P}_1$ for these objectives. We illustrate this on the following example.

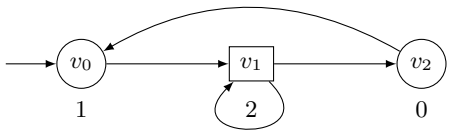

Fig. 4. The initial vertex v_0 is winning for the parity objective but is losing for all variants of objectives WP and PR: $\mathcal{P}_2$ has the possibility to use the self-loop on v_1 to delay for an arbitrarily long time the response 0 to priority 1, and can do so repeatedly using the other loop, thus defeating both direct and undirect variants of the objectives.

Example 8. Recall the game in Figure 1. We showed in Example 3 that the only possible outcome, $\pi = (v_0v_1v_2v_3)^\omega$, belongs to $\mathsf{DirFixPR}(\lambda, p) \setminus \mathsf{DirFixWP}(\lambda, p)$ for $\lambda = 3$. This shows that inclusion of Item 4 in Proposition 7 is strict. As this game is a one-player game with a unique play, it follows that $\mathsf{Win}_1^G(\mathsf{DirFixWP}(3, p)) \subsetneq \mathsf{Win}_1^G(\mathsf{DirFixPR}(3, p))$.

Consider now the game depicted in Figure 4. First, we show that vertex v_0 is winning for $\mathcal{P}_1$ for the parity objective. Indeed, either $\mathcal{P}_2$ eventually loops forever on vertex v_1, or he visits vertex v_2 infinitely often. Then, either priority 2 (of v_1) is the only one seen infinitely, or priority 0 (of v_2) is seen infinitely often, which shows that $v_0 \in \mathsf{Win}_1^G(\mathsf{Parity}(p))$. However, $\mathcal{P}_1$ loses from v_0 for the objective $\mathsf{DirFixPR}(\lambda, p)$ (for all $\lambda \in \mathbb{N}_0$) since $\mathcal{P}_2$ can ensure that priority 1 is never followed by a $\preceq$-smaller priority by always looping on vertex v_1. Thus, $\mathsf{Win}_1^G(\mathsf{DirFixPR}(\lambda, p)) \subsetneq \mathsf{Win}_1^G(\mathsf{Parity}(p))$ for all λ, and thus also $\mathsf{Win}_1^G(\mathsf{DirBndPR}(p)) \subsetneq \mathsf{Win}_1^G(\mathsf{Parity}(p))$. The same strategy permits $\mathcal{P}_2$ to win the game when the objective is $\mathsf{DirFixWP}(\lambda, p)$ for all λ, and thus also when the objective is $\mathsf{DirBndWP}(p)$.

Let us go further with the game in Figure 4 by considering the $\mathsf{BndWP}(p)$ and $\mathsf{BndPR}(p)$ objectives, which are prefix-independent as the parity objective. We can see that v_0 is still losing for $\mathcal{P}_1$ for both objectives. Indeed, consider the infinite-memory strategy σ_2 of $\mathcal{P}_2$ which consists in looping for longer and longer time periods in v_1 before going to v_2. Formally, let n be a counter initialized to 1, the strategy σ_2 loops n times in v_1, increments n and goes to v_2. Such a strategy increases the number of steps before priority 1 is followed by a $\preceq$-smaller priority each "round". Therefore, there exists no bound $\lambda \in \mathbb{N}_0$ for which the outcome belongs to $\mathsf{FixPR}(\lambda, p)$ and, similarly, there is no window size $\lambda \in \mathbb{N}_0$ for which the outcome belongs to $\mathsf{FixWP}(\lambda, p)$. Essentially, $\mathcal{P}_2$ uses his infinite-memory to create ever-increasing delays. This shows that for all $\lambda \in \mathbb{N}_0$, $v_0 \notin \mathsf{Win}_1^G(\mathsf{FixPR}(\lambda, p))$ and $v_0 \notin \mathsf{Win}_1^G(\mathsf{FixWP}(\lambda, p))$. Consequently, we also have that $v_0 \notin \mathsf{Win}_1^G(\mathsf{BndPR}(p))$ and $v_0 \notin \mathsf{Win}_1^G(\mathsf{BndWP}(p))$.

Observe that σ_2 uses *infinite memory*. Actually, $\mathcal{P}_2$ cannot win the *undirect bounded* objectives from v_0 for any finite-memory strategy σ_2'. Indeed, in this case, either the outcome π eventually loops on v_1 forever, or each time it visits v_1, π loops on v_1 at most m times (for some $m \in \mathbb{N}$ depending on the finite memory of σ_2') and then goes to v_2. It follows that the delay is *eventually* bounded in both cases and $\pi \in \mathsf{FixPR}(m + 2, p) \subseteq \mathsf{BndPR}(p)$ and $\pi \in \mathsf{FixWP}(m + 2, p) \subseteq \mathsf{BndWP}(p)$. ◁

We close this section by establishing that for the sub-case of games with priorities in $\{0, 1, 2\}$, WP and PR objectives coincide.

Lemma 9. *Let G be a game structure and $p\colon V \to \{0,1,2\}$ be a priority function. For all $\lambda \in \mathbb{N}_0$,*

$$(\mathsf{Dir})\mathsf{FixPR}(\lambda, p) = (\mathsf{Dir})\mathsf{FixWP}(\lambda, p).$$

Proof. Again, we only give the proof for the direct variant. By Proposition 7, we already know that $\mathsf{DirFixWP}(\lambda, p) \subseteq \mathsf{DirFixPR}(\lambda, p)$. To show the other inclusion, let π be a play in $\mathsf{DirFixPR}(\lambda, p)$. Then, for all $j \geq 0$, there exists $l \in \{0, \dots, \lambda - 1\}$ such that $p(\pi[j+l]) \preceq p(\pi[j])$. Now, as $p\colon V \to \{0,1,2\}$, we have that either $p(\pi[j])$ is even (and $l = 0$) or $p(\pi[j]) = 1$ and $p(\pi[j+l]) = 0$. In particular, $p(\pi[j+l])$ is the $\preceq$-smallest priority in $\pi[j, j+l]$ showing that $\pi \in \mathsf{DirFixWP}(\lambda, p)$. ◻

4 One-dimension games

We begin our study of WP and PR objectives with one-dimension games: in this setting, there is a unique priority function p and the objective Ω is a single objective (Dir)FixX or (Dir)BndX for $\mathsf{X} \in \{\mathsf{PR}, \mathsf{WP}\}$. We first address the *bounded* variants in Section 4.1, then turn to the *fixed* ones in Section 4.2.

4.1 Bounded variants

Recall that we know by Proposition 7 that the bounded variants are equivalent. Furthermore, it is already known that games with objective (Dir)BndPR(p) are solvable in polynomial time [9]. The next theorem sums up the complexity landscape for bounded variants and enrich it by proving P-hardness for the associated decision problems. In terms of memory requirements, $\mathcal{P}_1$ can play without memory whereas Example 8 already illustrated that $\mathcal{P}_2$ requires infinite memory in general. The linear memory bound for $\mathcal{P}_2$ and the direct variant was established in [18].

Theorem 10. *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, p be a priority function, and Ω be the objective* DirBndPR(p) *or* DirBndWP(p) *(resp.* BndPR(p) *or* BndWP(p)*).*

1. *Deciding the winner in (G, Ω) from v_0 is* P*-complete with an algorithm in $\mathcal{O}(|V| \cdot |E|)$ (resp. $\mathcal{O}(|V|^2 \cdot |E|)$) time, memoryless strategies are sufficient for $\mathcal{P}_1$, and linear-memory strategies are necessary and sufficient for $\mathcal{P}_2$ (resp. infinite memory is necessary for $\mathcal{P}_2$).*
2. *$\forall\, \lambda \geq |V|$, $\forall\, \lambda' \geq \frac{d}{2} \cdot |V|$, the winning sets for the objectives* BndPR(p), FixPR(λ, p), BndWP(p), *and* FixWP(λ', p) *are all equal.*
3. *The equalities given in Item 2 also hold for the direct variants* (Dir)*.*

Proof. (a) Let us prove Item 1. The DirBndPR(p) (resp. BndPR(p)) objective is studied in [9] and shown to be in P with an algorithm in $\mathcal{O}(|V| \cdot |E|)$ (resp. $\mathcal{O}(|V|^2 \cdot |E|)$) time. Moreover it is shown that memoryless strategies are sufficient for $\mathcal{P}_1$, and linear-memory strategies are necessary and sufficient for $\mathcal{P}_2$ [18] (resp. infinite memory is necessary for $\mathcal{P}_2$). As (Dir)BndPR(p) = (Dir)BndWP(p) by Item 6 of Proposition 7, we have the same complexity results and memory requirements for the (Dir)BndWP(p) objectives.

To complete the proof of Item 1, we have to prove that deciding the winner in $(G, (\mathsf{Dir})\mathsf{BndWP}(p))$ is P-hard. We begin with the undirect variant. Let $G_r = (V_1, V_2, E_r)$ be a game structure, and consider the reachability objective Reach(U) with the target set $U \subseteq V$. From G_r, we build the game structure $G = (V_1, V_2, E)$ by (i) making the target vertices absorbing with a self-loop and (ii) defining the priority function $p\colon V \to \{0,1\}$ as follows: $p(v) = 0$ if $v \in U$, and $p(v) = 1$ otherwise. We claim that $\mathcal{P}_1$ has a winning strategy in G_r from an initial vertex v_0 for the Reach(U) objective if and only if he has a winning strategy for the BndWP(p) objective in G from v_0. Indeed, any outcome that never reaches the target set U in G_r is such that for all $j \in \mathbb{N}$ the window at position j in G stays open forever, i.e., it is λ-bad for any size $\lambda \in \mathbb{N}_0$. Conversely, any outcome that reaches some $v \in U$ in j steps in G_r has a corresponding outcome in G that reaches v in j steps and then loops on v (recall that v is absorbing), that is, from position j all windows of size $\lambda = 1$ are closed. Thus v_0 is winning for the Reach(U) objective in G_r if and only if v_0 is winning for the BndWP(p) objective in G. This concludes the proof of P-hardness in case of BndWP(p) objective since deciding the winner in reachability games is P-complete [2,21].

The same reduction holds for the direct variant $\mathsf{DirBndWP}(p)$. We already know that if $\mathcal{P}_1$ has a winning strategy for $\mathsf{BndWP}(p)$ in G, he has one for $\mathsf{Reach}(U)$ in G_r. Since $\mathsf{DirBndWP}(p) \subseteq \mathsf{BndWP}(p)$ by Proposition 7, Item 1, extending this direction to $\mathsf{DirBndWP}(p)$ is trivial. It remains to consider the converse, that is, if $\mathcal{P}_1$ has a winning strategy for $\mathsf{Reach}(U)$ in G_r, he has one for $\mathsf{DirBndWP}(p)$ in G. Using the arguments from above, we actually see that any winning play $\pi = v_0 v_1 \dots$ in G_r belongs to $\mathsf{DirFixWP}(k+1, p)$ in G where k is the first index such that $v_k \in U$. Hence, all winning plays in G_r belong to $\mathsf{DirBndWP}(p)$ in G.

(b) Let us now proceed with the proof of Items 2 and 3. Recall that $(\mathsf{Dir})\mathsf{BndPR}(p) = (\mathsf{Dir})\mathsf{BndWP}(p)$ by Item 6 of Proposition 7. Let us consider the PR objectives. It is shown in [9] that $\mathsf{Win}_1^G((\mathsf{Dir})\mathsf{BndPR}(p)) = \mathsf{Win}_1^G((\mathsf{Dir})\mathsf{FixPR}(\lambda, p))$ with $\lambda = |V|$. By Items 2 and 3 of Proposition 7, we also have equalities of those sets with $\mathsf{Win}_1^G((\mathsf{Dir})\mathsf{FixPR}(\lambda', p))$ for all $\lambda' \geq |V|$. We get the required equalities for WP objectives by Items 3, 4, 5 and 6 of Proposition 7. □

4.2 Fixed variants

The fixed variants are more interesting: the PR and WP approaches yield different results in this setting. We start with the PR one, for which we provide two polynomial-time algorithms for fixed-parameter sub-cases, hence significantly reducing the complexity of the problem (which is PSPACE-complete in the general case).

Parity-response objectives. Deciding the winner in (Dir)FixPR games was very recently proved to be PSPACE-complete [32]. As mentioned in Remark 4, the proof was actually provided for a more general model, but already holds for both FixPR and DirFixPR games.

Theorem 11 ([32]). *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, p be a priority function, and Ω be the objective $(\mathsf{Dir})\mathsf{FixPR}(\lambda, p)$ for some $\lambda < |V|$. Deciding the winner in (G, Ω) from v_0 is PSPACE-complete. Moreover, exponential memory is both necessary and sufficient for $\mathcal{P}_1$, and exponential memory is sufficient for $\mathcal{P}_2$ while linear memory is necessary.*

Observe that the PSPACE-hardness only holds for time bounds $\lambda < |V|$ since we know by Theorem 10, Items 2 and 3, that for larger values, the objectives are equivalent to the bounded variants, hence the corresponding decision problems lie in P. In addition, we focus on the case $\lambda < |V|$: we show in the next theorem that when we fix either the largest priority d or the bound λ, the complexity collapses to P. We briefly sketch the two algorithms here (we illustrate the memory bounds in the upcoming Example 13).

First, consider the case where d is fixed. We reduce the $\mathsf{FixPR}(\lambda, p)$ (resp. $\mathsf{DirFixPR}(\lambda, p)$) game to a co-Büchi (resp. safety) game on an extended graph where we keep track of additional information in the vertices. Namely, we keep a vector that represents, for each odd priority c, the number of steps since seeing c without seeing any $\preceq$-smaller priority in the meantime. When this number reaches λ for any odd priority, we visit a special "bad vertex" and then reset the counters in the vector and resume the game. Essentially, winning for $\mathsf{FixPR}(\lambda, p)$ (resp. $\mathsf{DirFixPR}(\lambda, p)$) boils down to eventually (resp. completely) avoiding those bad vertices, hence to a co-Büchi (resp. safety) game. This extended game has size $\mathcal{O}(|V| \cdot \lambda^{\frac{d}{2}})$ and can be solved in polynomial time since $\lambda < |V|$ (otherwise we use the algorithm for the *bounded* variants presented in Theorem 10) and d is fixed.

Second, consider the case where λ is fixed (and $< |V|$ for the same reason as before). We also reduce the $\mathsf{FixPR}(\lambda, p)$ (resp. $\mathsf{DirFixPR}(\lambda, p)$) game to a co-Büchi (resp. safety) game, but with a different extended graph. Specifically, we here keep track of the last λ vertices seen in the original game, and we want to avoid vertices of the extended graph that correspond to histories where an odd priority c is not followed by a priority $c' \preceq c$ within $(\lambda - 1)$ steps. Again, this can be expressed as either a co-Büchi or a safety objective depending on whether we are interested in the undirect or the direct variant respectively. The extended game has size $\mathcal{O}(|V|^\lambda)$ hence can be solved in polynomial time since λ is fixed.

Theorem 12. *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, $p \colon V \to \{0, \dots, d\}$ be a priority function, and Ω be the objective $\mathsf{DirFixPR}(\lambda, p)$ (resp. $\mathsf{FixPR}(\lambda, p)$) for some $\lambda \in \mathbb{N}_0$. If either d is fixed or λ is fixed, deciding the winner in (G, Ω) from v_0 is in P. More precisely,*

1. *If $\lambda \geq |V|$, deciding the winner can be done in $\mathcal{O}(|V|\cdot|E|)$ (resp. $\mathcal{O}(|V|^2\cdot|E|)$) time, memoryless strategies are sufficient for $\mathcal{P}_1$, and linear-memory strategies are both necessary and sufficient for $\mathcal{P}_2$ (resp. infinite memory is necessary for $\mathcal{P}_2$).*
2. *If $\lambda < |V|$ and d is fixed, deciding the winner can be done in $\mathcal{O}((|V|+|E|)\cdot\lambda^{\frac{d}{2}})$ (resp. $\mathcal{O}(|V|^2\cdot\lambda^d)$) time, polynomial-memory strategies with $\mathcal{O}(\lambda^{\frac{d}{2}})$ memory are sufficient for both players, and memory is necessary even in one-player games.*
3. *If $\lambda < |V|$ and λ is fixed, deciding the winner can be done in $\mathcal{O}((|V|+|E|)\cdot|V|^{\lambda-1})$ (resp. $\mathcal{O}(|V|^{2\lambda})$) time, polynomial-memory strategies with $\mathcal{O}(|V|^{\lambda-1})$ memory are sufficient for both players, and memory is necessary even in one-player games.*

Proof. (a) The first item directly follows from Theorem 10.

(b) Let us prove Item 2. Let $\lambda < |V|$. We first consider the undirect variant. From G, we construct a game G' that keeps track in its vertices of the current vertex v of G and whether each seen odd priority c has been followed by a $\preceq$-smaller priority within at most $\lambda-1$ steps. To this end, to each odd priority c is associated a counter $l_c \in \{\bot, 0, 1, \dots, \lambda-2\}$ such that during the $\lambda-1$ last steps (i) either $l_c = \bot$ when c has not been seen, or c has been seen and followed by a $\preceq$-smaller priority, (ii) or l_c is the number of steps from c to the current vertex v. If $l_c = \bot$ and a new occurrence of c is seen, l_c is initialized to 0. If $l_c \neq \bot$, if a $\preceq$-smaller priority is seen in at most $\lambda-1$ steps, we reset l_c to $\bot$, otherwise we move to a special vertex. Formally, we define $V' = V \times \{\bot, 0, 1, \dots, \lambda-2\}^{\frac{d}{2}} \cup \beta_V$, where the additional set $\beta_V = \{\beta_v \mid v \in V\}$ is composed of the special vertices.

Let $u = (v, \bar{l}) \in V' \setminus \beta_V$ be such that $v \in V$, $\bar{l}$ is a vector of counters l_c, one for each odd priority c. Given $(v, v') \in E$, we construct the edge $(u, u') \in E'$ such that

$$u' = \begin{cases} \beta_{v'} & \text{if } \exists c,\ l_c = \lambda-2 \text{ and } p(v') \not\preceq c, \\ (v', \bar{l}') & \text{otherwise,} \end{cases}$$

where in the second case

$$l'_c = \begin{cases} l_c + 1 & \text{if } l_c \neq \bot \text{ and } p(v') \not\preceq c, \\ \bot & \text{if } l_c \neq \bot \text{ and } p(v') \preceq c, \\ 0 & \text{if } l_c = \bot \text{ and } p(v') = c, \\ \bot & \text{otherwise.} \end{cases}$$

Notice that in the previous cases, if $l_c \neq \bot$ and $p(v') = c$, we do not define $l'_c = 0$. Indeed, if a new occurrence of c is detected ($p(v') = c$), we have to check that the previous occurrence of c ($l_c \neq \bot$) is followed by a $\preceq$-smaller priority c' within at most $\lambda-1$ steps. If this happens, the last occurrence will automatically be followed by the same $\preceq$-smaller priority c' and it is not necessary to keep track of the second occurence specifically.

For all $v \in V$, we also add the edge $(\beta_v, (v, \bar{l}))$ to E' such that $l_c = \bot$ for all odd priorities c except if $p(v)$ is odd in which case $l_{p(v)} = 0$. In this way, we allow to delay the check performed on each odd priority. If v_0 is an initial vertex of G, then the corresponding initial vertex in G' is $u_0 = (v_0, \bar{l})$ such that $\bar{l}$ is defined as done previously. Finally we define the objective $\Omega' = \mathsf{CoBuchi}(U')$ with $U' = V' \setminus \beta_V$. One can check that $\mathcal{P}_1$ has a winning strategy from v_0 in (G, Ω) if and only if $\mathcal{P}_1$ has a winning strategy from u_0 in (G', Ω'). As $|V'| = \mathcal{O}(|V|\cdot\lambda^{\frac{d}{2}})$ and $|E'| = \mathcal{O}(|E|\cdot\lambda^{\frac{d}{2}})$, the size of the game G' is polynomial in the size of the original game G since $\lambda < |V|$ and d is fixed. As deciding the winner in the co-Büchi game (G', Ω') can be solved in $\mathcal{O}(|V'|^2)$ time [8], deciding the winner in the game (G, Ω) can be solved in $\mathcal{O}((|V|^2\cdot\lambda^d)$ time. Moreover, as memoryless strategies are sufficient for both players to win in (G', Ω'), finite-memory strategies with $\mathcal{O}(\lambda^{\frac{d}{2}})$ memory are sufficient for both players to win in (G, Ω). We show in Example 13 that memory is necessary for both players.

We now turn to the direct variant. The construction of the game G' is rather similar, except that a unique absorbing special vertex β is sufficient, and $\Omega' = \mathsf{Safe}(U')$ with $U' = V' \setminus \{\beta\}$. Then, $\mathcal{P}_1$ wins the game $(G, \mathsf{DirFixWP}(\lambda, p))$ if and only if he wins the new game $(G', \mathsf{Safe}(U'))$. The corresponding algorithm runs in

$\mathcal{O}(|V'| + |E'|) = \mathcal{O}((|V| + |E|) \cdot \lambda^{\frac{d}{2}})$ time. Moreover, the memoryless winning strategies in the safety game (G', Ω') lead to finite-memory strategies with $\mathcal{O}(\lambda^{\frac{d}{2}})$ memory in the original game. The need for memory is also presented in Example 13.

(c) We now proceed to the proof of Item 3. For the undirect variant, we construct from G a game G' that keeps in its vertices the last λ vertices (of G) seen including the current vertex v. Formally, we define $V' = V^\lambda$. For each $u = (w_1, \dots, w_{\lambda-1}, v) \in V'$, and $(v, v') \in E$, we construct the edge $(u, u') \in E'$ such that $u' = (w_2, \dots, w_{\lambda-1}, v, v')$. Given v_0 an initial vertex in G, we let $u_0 = (w, \dots, w, v_0)$ be the corresponding initial vertex of G' such that w is a vertex of V with the highest even priority (w is chosen in a way to have no influence for the objectives considered here). We also define the objective $\Omega' = \mathsf{CoBuchi}(U')$ with $U' = \{(w_1, \dots, w_\lambda) \in V' \mid \exists\, l \in \{0, \dots, \lambda-1\}$ such that $p(w_{l+1}) \preceq p(w_1)\}$. Clearly, $\mathcal{P}_1$ has a winning strategy from v_0 in (G, Ω) if and only if $\mathcal{P}_1$ has a winning strategy from u_0 in (G', Ω'). Note that the size of the game G' is polynomial in the size of the original game G, with $|V'| = \mathcal{O}(|V|^\lambda)$ and $|E'| = \mathcal{O}(|E| \cdot |V|^{\lambda-1})$ since λ is fixed. Therefore deciding the winner in (G, Ω) can be done in $\mathcal{O}(|V|^{2\lambda})$ time and both players have finite-memory winning strategies with $\mathcal{O}(|V|^{\lambda-1})$ memory. We show in Example 13 that both players need memory to win.

For the direct variant, the game G' is identical but with the safety objective $\Omega' = \mathsf{Safe}(U')$. We get an algorithm in $\mathcal{O}((|V| + |E|) \cdot |V|^{\lambda-1})$ time and winning strategies with $\mathcal{O}(|V|^{\lambda-1})$ memory for both players. Example 13 shows that memory is necessary for both players. □

The next example shows that both players need memory in fixed PR games with fixed parameters.

Example 13. Consider the game depicted in Figure 5 where all vertices belong to $\mathcal{P}_1$. We claim that $\mathcal{P}_1$ needs memory to win for (Dir)FixPR(λ, p) with $\lambda = 4$. Indeed, if he plays memoryless by always going to v_1 from v_0, then in the resulting outcome $\pi = (v_0 v_1 v_2)^\omega$ the odd priority 1 of vertex v_2 is followed by no $\preceq$-smaller priority, showing that $\pi \notin \mathsf{FixPR}(4, p)$ (and thus $\pi \notin \mathsf{DirFixPR}(4, p)$). Now, if $\mathcal{P}_1$ always goes to v_3 from v_0 then the priority 3 of vertex v_5 is followed by the $\preceq$-smaller priority 0 of vertex v_4 in 4 steps ($\lambda' = 5$), showing that $\pi = (v_0 v_3 v_4 v_5 v_6)^\omega \notin \mathsf{FixPR}(4, p)$ (hence $\pi \notin \mathsf{DirFixPR}(4, p)$). Thus $\mathcal{P}_1$ has no memoryless winning strategy for (Dir)FixPR$(4, p)$. However, if he alternates between the two cycles, thus producing the outcome $\pi = (v_0 v_1 v_2 v_0 v_3 v_4 v_5 v_6)^\omega$, one can check that π belongs to DirFixPR$(4, p)$, and thus also to FixPR$(4, p)$.

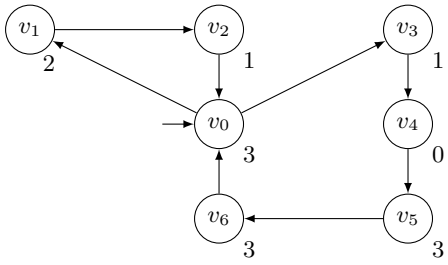

Fig. 5. $\mathcal{P}_1$ needs to alternate between the two simple cycles to win for (Dir)FixPR$(4, p)$.

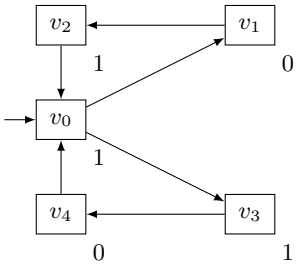

Fig. 6. $\mathcal{P}_2$ needs to alternate between the two simple cycles to win for (Dir)FixPR$(3, p)$.

Let us now focus on the game depicted in Figure 6, where all vertices belong to $\mathcal{P}_2$. We show that $\mathcal{P}_2$ needs memory to win both (Dir)FixPR(λ, p) objectives with $\lambda = 3$. If $\mathcal{P}_2$ plays memoryless by always going to v_1 from v_0, then the resulting play $(v_0 v_1 v_2)^\omega$ belongs to DirFixPR$(3, p) \subseteq$ FixPR$(3, p)$, and similarly if he always goes to v_3 from v_0. He is thus losing for (Dir)FixPR$(3, p)$. However, if $\mathcal{P}_2$ alternates between the two cycles to produce the outcome $\pi = (v_0 v_1 v_2 v_0 v_3 v_4)^\omega$, we have $\pi \notin \mathsf{FixPR}(3, p)$ (hence $\pi \notin \mathsf{DirFixPR}(3, p)$) since priority 1 of vertex v_2 is followed by the $\preceq$-smaller priority 0 of vertex v_4 in 3 steps ($\lambda' = 4$). This way, $\mathcal{P}_2$ wins for (Dir)FixPR$(3, p)$. ◁

Window parity objectives. Whereas deciding the winner in (Dir)FixPR games is PSPACE-complete, we establish in the next theorem that it is P-complete for (Dir)FixWP games. First observe that if $\lambda \geq \frac{d}{2} \cdot |V|$, the problem boils down to solving the bounded variant thanks to Theorem 10. Hence, we focus on the case where $\lambda < \frac{d}{2} \cdot |V|$.

Our algorithm is inspired by the approach developed for *window mean-payoff games* in [6]. It can be sketched as follows. As for the fixed-parameter algorithms for (Dir)FixPR games presented in Theorem 12, we want to reduce the FixWP and DirFixWP games to co-Büchi and safety games respectively, where $\mathcal{P}_1$ wants to avoid "bad vertices" representing a violation of the condition at stake. Here, such a violation represents a λ-*bad window*, i.e., a window for which no even minimum priority is found before λ steps (see the terminology in Section 3.2). Detecting such λ-bad windows can be achieved by considering an extended game structure where we encode additional information for the minimum priority of the current window and the number of steps in this window. A "bad vertex" is visited whenever we reach the end of a λ-window with an odd minimum priority. If an even minimum is found, a λ-good history is detected and the step counter is reset. The extended game has size $\mathcal{O}(|V| \cdot d \cdot \lambda)$, hence polynomial size since $\lambda < \frac{d}{2} \cdot |V|$. Therefore, we can solve it in polynomial time. This is in contrast to window mean-payoff games where the fixed variant requires *pseudo*-polynomial time in general [6].

Upper bounds on the memory are obtained by construction of our reduction and we prove polynomial lower bounds in the upcoming Example 15.

Theorem 14. *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, p be a priority function, and Ω be the objective* DirFixWP(λ, p) *(resp.* FixWP(λ, p)*) for some $\lambda \in \mathbb{N}_0$. Then deciding the winner in (G, Ω) from v_0 is* P*-complete.*

1. *If $\lambda \geq \frac{d}{2} \cdot |V|$, deciding the winner can be done in $\mathcal{O}(|V| \cdot |E|)$ (resp. $\mathcal{O}(|V|^2 \cdot |E|)$) time, memoryless strategies are sufficient for $\mathcal{P}_1$ and linear-memory strategies are necessary and sufficient for $\mathcal{P}_2$ (resp. infinite memory is necessary for $\mathcal{P}_2$).*
2. *If $\lambda < \frac{d}{2} \cdot |V|$, deciding the winner can be done in $\mathcal{O}((|V| + |E|) \cdot d \cdot \lambda)$ (resp. $\mathcal{O}(|V|^2 \cdot d^2 \cdot \lambda^2)$) time, and polynomial-memory strategies with $\mathcal{O}(d \cdot \lambda)$ memory are sufficient for both players. Moreover, polynomial memory is necessary for both players.*

Proof. The reduction from reachability games used for Theorem 10 also suffices to obtain P-hardness for (Dir)FixWP games, so it remains to establish a polynomial-time algorithm and study the memory requirements for the winning strategies. If $\lambda \geq \frac{d}{2} \cdot |V|$, the results of Item 1 follow from Theorem 10. Hence we now suppose that $\lambda < \frac{d}{2} \cdot |V|$.

We begin by studying the *undirect* variant. By Lemma 6, a play belongs to FixWP(λ, p) if and only if it has an eventually λ-good decomposition. Therefore from G, we construct a game G' able to detect λ-good histories. That is, we keep in the vertices of G' the current vertex of G, the minimum priority of the current window and the number of steps performed in the current window. As soon as the minimum priority is even (in at most $\lambda - 1$ steps), a λ-good history has been detected, and the information is reset with a new window. More precisely, we define $V' = V \times \{0, \dots, d\} \times \{0, \dots, \lambda - 1\} \cup \beta_V$, where the additional set $\beta_V = \{\beta_v \mid v \in V\}$ is composed of special vertices for the detection of a λ-bad window. Let $u = (v, c, l) \in V' \setminus \beta_V$ be such that $v \in V$, c is the current minimum priority and l is the current number of steps. Given $(v, v') \in E$, we construct the edge $(u, u') \in E'$ such that

$$u' = \begin{cases} (v', p(v'), 0) & \text{if } c \text{ is even,} \\ (v', \min(c, p(v')), l + 1) & \text{if } c \text{ is odd and } l < \lambda - 1, \\ \beta_{v'} & \text{otherwise.} \end{cases}$$

We also add the edges $(\beta_v, (v, p(v), 0))$ to E' for all $v \in V$. If v_0 is an initial vertex of G, then the corresponding initial vertex in G' is $u_0 = (v_0, p(v_0), 0)$. Finally, we define the objective $\Omega' =$ CoBuchi(U') with $U' = V' \setminus \beta_V$. Observe that a play winning for Ω' corresponds to a play accepting an *eventually λ-good decomposition* (which is easily obtained by looking at the "resets" of the step counter).

Clearly, thanks to Lemma 6, $\mathcal{P}_1$ has a winning strategy from v_0 in (G, Ω) if and only if $\mathcal{P}_1$ has a winning strategy from u_0 in (G', Ω'). As $|V'| = \mathcal{O}(|V| \cdot d \cdot \lambda)$ and $|E'| = \mathcal{O}(|E| \cdot d \cdot \lambda)$, the size of the game G' is polynomial in the size of the original game G (since $\lambda < \frac{d}{2} \cdot |V|$). It follows that deciding the winner in the game (G, Ω) can be solved in $\mathcal{O}(|V|^2 \cdot d^2 \cdot \lambda^2)$ time [8], and that finite-memory strategies with $\mathcal{O}(d \cdot \lambda)$ memory are sufficient for both players to win.

Let us turn to the *direct* variant. By Lemma 6, a play belongs to $\mathsf{DirFixWP}(\lambda, p)$ if and only if it has a λ-good decomposition. The construction of the game G' is thus rather similar, except that a unique absorbing vertex β is sufficient for the detection of a λ-bad window, and $\Omega' = \mathsf{Safe}(U')$ with $U' = V' \setminus \{\beta\}$. Then, $\mathcal{P}_1$ wins the game $(G, \mathsf{DirFixWP}(\lambda, p))$ if and only if he wins the new game $(G', \mathsf{Safe}(U'))$. The corresponding algorithm runs in $\mathcal{O}((|V|+|E|) \cdot d \cdot \lambda)$ time. Moreover, both players have finite-memory strategies with $\mathcal{O}(d \cdot \lambda)$ memory.

The necessity of polynomial memory is established in Example 15. □

In the next example, we illustrate the need for polynomial memory, for both players, in (Dir)FixWP games. We use game families proposed in [7].

Example 15. Consider the game in Figure 7 where the unlabeled vertices have all priority $d = 6$. This example can easily be generalized to any even $d \geq 0$, and observe that the size of the game is $|V| = 2 + \frac{d}{2} \cdot (\frac{d}{2} + 1)$, hence polynomial in d.

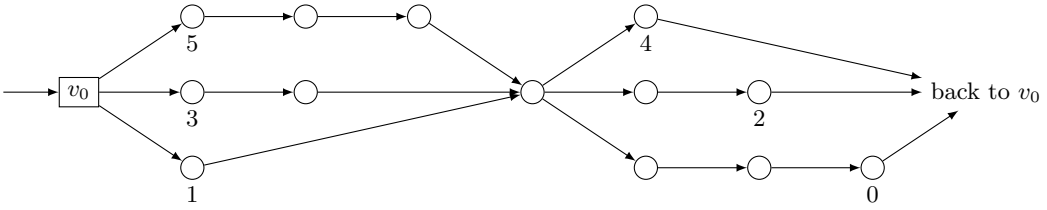

Fig. 7. In order to win for objective (Dir)FixWP$(5, p)$, $\mathcal{P}_1$ needs to answer to priority 5 (resp. 3, 1) by choosing the path with priority 4 (resp. 2, 0). This requires $\mathcal{O}(\frac{d}{2})$ memory (here, $d = 6$).

We claim that $\mathcal{P}_1$ needs memory of size $\frac{d}{2}$ to win the game (G, Ω) with $\Omega = (\mathsf{Dir})\mathsf{FixWP}(\lambda, p)$ with $\lambda = \frac{d}{2} + 2$. Indeed, each time $\mathcal{P}_2$ chooses the path with priority $(d-1)$ (resp. $(d-3)$, …, 1), the only possibility for $\mathcal{P}_1$ is to choose the path with priority $(d-2)$ (resp. $(d-4)$, …, 0) otherwise he creates a λ-bad window. If $\mathcal{P}_1$ uses a finite-memory strategy with less than $\frac{d}{2}$ memory, he has to answer to two different odd priorities with the same choice, and $\mathcal{P}_2$ can take advantage of this to create λ-bad windows at every visit of v_0. Hence, $\mathcal{P}_2$ can prevent $\mathcal{P}_1$ from winning for $\mathsf{FixWP}(\lambda, p)$, a fortiori for $\mathsf{DirFixWP}(\lambda, p)$.

We thus have a family of games polynomial in d and for which $\mathcal{P}_1$ needs polynomial memory to win for objective (Dir)FixWP(λ, p).

We now turn to a second example to illustrate the necessity of polynomial memory for $\mathcal{P}_2$. Consider the family of game structures (parameterized by $n \geq 2$) depicted in Figure 8. From vertex v_0, $\mathcal{P}_2$ can choose one among the n outgoing edges $(v_0, v_{\neq i})$, and from vertex $v_{\neq i}$, $\mathcal{P}_1$ can choose one among the $(n-1)$ outgoing paths $\rho_j = v_1 \dots v_n$ of length n, with $j \neq i$, the vertices of which all have priority 1 except vertex v_j having priority 0. Vertices v_0 and $v_{\neq i}$, $i \in \{1, \dots, n\}$, all have priority 1. These choices of both players alternate infinitely many times. Observe that the size of this game is $|V| = 1 + n \cdot (n+1)$, hence polynomial in n.

We claim that $\mathcal{P}_2$ needs memory of size n to win the game $(G, \overline{\Omega})$ with $\Omega = (\mathsf{Dir})\mathsf{FixWP}(\lambda, p)$ such that $\lambda = n + 3$. (*i*) Let us first explain that he can win with such a memory. Indeed, at each alternation, $\mathcal{P}_2$ records the last choice ρ_i of $\mathcal{P}_1$ and then chooses $(v_0, v_{\neq i})$. At the next alternation, $\mathcal{P}_1$ must choose ρ_j with either $j > i$ or $j < i$. The first case necessarily occurs infinitely often since eventually i equals 1 if $\mathcal{P}_1$ keeps choosing $j < i$. Now, observe that when $\mathcal{P}_1$ chooses $j > i$, the play contains $(n+2)$ consecutive vertices with priority 1, hence a λ-bad window since $\lambda = n + 3$. This shows that $\mathcal{P}_2$ wins for objective $\overline{\mathsf{FixWP}(\lambda, p)}$ and

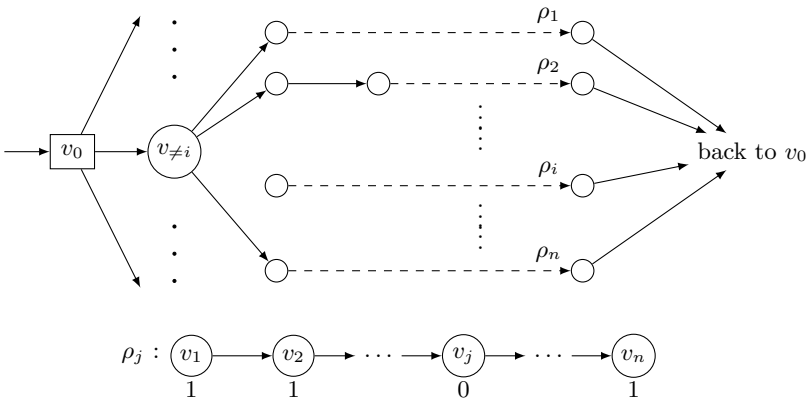

Fig. 8. In order to win for objective $\overline{\mathsf{(Dir)FixWP}(n+3,p)}$, $\mathcal{P}_2$ needs to answer to the choice ρ_i made by $\mathcal{P}_1$ by taking vertex $v_{\neq i}$, hence eventually forcing an $(n+3)$-bad window when $\mathcal{P}_1$ is forced to pick ρ_j with $j > i$. This requires $\mathcal{O}(n)$ memory.

thus for $\overline{\mathsf{DirFixWP}(\lambda,p)}$ too. (*ii*) Let us now show that $\mathcal{P}_2$ is losing with a memory of size less then n. In this case, there is a vertex $v_{\neq i}$ that he will never choose. Thus $\mathcal{P}_1$ can win for objective $\mathsf{DirFixWP}(\lambda,p)$ (a fortiori for $\mathsf{FixWP}(\lambda,p)$) by choosing the path ρ_i at each alternation. Indeed, this will only induce sequences of $(n+1)$ consecutive priorities equal to 1 separated by priority 0, which is fine since $\lambda = n+3$.

We thus have a family of games polynomial in n and for which $\mathcal{P}_2$ needs polynomial memory to win for objective $\overline{\mathsf{(Dir)FixWP}(\lambda,p)}$. ◁

5 Multi-dimension games

We now consider multi-dimension games: in this setting, there are n priority functions $p_1, \ldots, p_n$ and the objective Ω is the *conjunction* of *identical* objectives Ω_m for each "dimension" (i.e., priority function), with Ω_m being $\mathsf{(Dir)FixX}$ or $\mathsf{(Dir)BndX}$ for $\mathsf{X} \in \{\mathsf{PR}, \mathsf{WP}\}$. As in the one-dimension case, we first address the *bounded* variants in Section 5.1, then turn to the *fixed* ones in Section 5.2.

5.1 Bounded variants

Recall that Proposition 7 established the equality of objectives $\mathsf{(Dir)BndWP}(p)$ and $\mathsf{(Dir)BndPR}(p)$ in the one-dimension setting. This equality trivially carries over to the multi-dimension setting, i.e., we have that $\cap_{m=1}^{n}\mathsf{(Dir)BndWP}(p_m) = \cap_{m=1}^{n}\mathsf{(Dir)BndPR}(p_m)$ since the individual objectives (one per priority function) are equal. Hence, it suffices to obtain our results for either WP or PR objectives.

Overview. The next theorem presents an overview of our results. For the sake of readability, its proof is split in several lemmas: we prove EXPTIME-membership and upper bounds on memory in Lemma 18, the equalities of Items 2 and 3 in Lemma 19, EXPTIME-hardness in Lemma 20, and finally, lower bounds on memory in Lemma 21.

Theorem 16. *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, $p_1, \ldots, p_n$ be n priority functions, and Ω be the objective $\cap_{m=1}^{n}\mathsf{DirBndPR}(p_m)$ or $\cap_{m=1}^{n}\mathsf{DirBndWP}(p_m)$ (resp. $\cap_{m=1}^{n}\mathsf{BndPR}(p_m)$ or $\cap_{m=1}^{n}\mathsf{BndWP}(p_m)$). Let $b = |V| \cdot 2^{n\cdot\frac{d}{2}} \cdot n \cdot \frac{d}{2}$.*

1. *Deciding the winner in (G,Ω) from v_0 is* EXPTIME*-complete with an algorithm in $\mathcal{O}(b^2)$ (resp. $\mathcal{O}(|V|\cdot b^2)$) time, and exponential-memory strategies are necessary and sufficient for both players (resp. for $\mathcal{P}_1$ and infinite-memory is necessary for $\mathcal{P}_2$).*

2. $\forall\, \lambda \geq b$, $\forall\, \lambda' \geq b \cdot \frac{d}{2}$, *the winning sets for the following objectives are all equal:* $\cap_{m=1}^{n} \mathsf{BndPR}(p_m)$, $\cap_{m=1}^{n} \mathsf{FixPR}(\lambda, p_m)$, $\cap_{m=1}^{n} \mathsf{BndWP}(p_m)$, *and* $\cap_{m=1}^{n} \mathsf{FixWP}(\lambda', p_m)$.
3. *The equalities given in Item 2 also hold for the direct variants* (Dir).

Before continuing with the proof of these results, let us comment them. First, observe that multi-dimension bounded WP or PR games are EXPTIME-complete whereas multi-dimension parity games are coNP-complete [11]. Hence, in this case, the boundedness requirements specified by WP or PR objectives induce higher complexity, unlike in one-dimension games, where they yield a lower one (P-complete instead of the long-standing UP ∩ coUP barrier of parity games). This implies that, in multi-dimension games, bounded WP or PR objectives cannot be used to approximate efficiently parity objectives, in contrast to one-dimension games. A similar dichotomy was already witnessed for *window mean-payoff games* with regard to approximation of classical *mean-payoff games* [6].

Interestingly, the decidability of multi-dimension bounded window mean-payoff games is still open and they are known to be non-primitive-recursive-hard [6], whereas we prove here EXPTIME-completeness for the parity counterpart of this objective. This suggests that the colossal complexity of bounded window mean-payoff games is a result of the quantitative nature of mean-payoff mixed with windows, and not an inherent drawback of the window mechanism.

Exponential-time algorithm and upper bounds on memory. To prove EXPTIME-membership, we have to introduce related games from the literature. First, let us consider *request-response games* [31,10]. Consider r sets of vertices $Rq_1, \dots, Rq_r$ representing requests and r sets of vertices $Rp_1, \dots, Rp_r$ representing the corresponding responses ($Rq_i, Rp_i \subseteq V$ for all i). The *request-response* objective $\mathsf{RR}((Rq_i, Rp_i)_{i=1}^{r})$ requires that *for all* i, whenever a vertex of Rq_i is visited, then, later on, a vertex of Rp_i is also visited.[8] Observe that by definition, this objective is *direct*, i.e., the condition must hold from the start, not only eventually. Solving these games is EXPTIME-complete with an algorithm in $\mathcal{O}((|V| \cdot 2^r \cdot r)^2)$ time, and exponential memory is both sufficient and necessary for both players [31,10].

In [9], Chatterjee et al. studied *bounded Streett* games, which using our terminology for the sake of consistency, can be equivalently seen as *direct bounded request-response* games. The corresponding objective $\mathsf{DirBndRR}((Rq_i, Rp_i)_{i=1}^{r})$ asks that there exist a bound $b \in \mathbb{N}_0$ such that if a request is visited, then the corresponding response is visited within b steps. Chatterjee et al. proved that such a bound always exists when the ("unbounded") request-response objective RR can be won by $\mathcal{P}_1$, as a by-product of the construction used to solve these games in [31]. Ergo, RR games are equivalent to DirBndRR games.

A prefix-independent variant of the DirBndRR objective is also studied in [9], under the name of *finitary Streett*. Again, to maintain consistency, we call it the *bounded request-response objective* BndRR. It is naturally defined from the direct variant DirBndRR in the same way as all *undirect* variants in this paper (Definition 2).

We sum up the results[9] for all these games in the next theorem.

Theorem 17 ([31,9,10]). *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, and $(Rq_i, Rp_i)_{i=1}^{r}$ be a set of r pairs of requests and responses, such that $Rq_i, Rp_i \subseteq V$. Let $b = |V| \cdot 2^r \cdot r$.*

1. *If $\mathcal{P}_1$ has a winning strategy for $\mathsf{DirBndRR}((Rq_i, Rp_i)_{i=1}^{r})$ (resp. $\mathsf{BndRR}((Rq_i, Rp_i)_{i=1}^{r})$), then he has one that enforces that (resp. eventually) every request is followed by a corresponding response in at most b steps.*
2. *Deciding the winner in the game $(G, \mathsf{DirBndRR}((Rq_i, Rp_i)_{i=1}^{r}))$ from v_0 is* EXPTIME*-complete with an algorithm in $\mathcal{O}(b^2)$ time, and exponential-memory strategies are sufficient for both players and necessary for $\mathcal{P}_1$.*
3. *Deciding the winner in the game $(G, \mathsf{BndRR}((Rq_i, Rp_i)_{i=1}^{r}))$ from v_0 is in* EXPTIME *with an algorithm in $\mathcal{O}(|V| \cdot b^2)$ time, exponential-memory strategies are both sufficient and necessary for $\mathcal{P}_1$, and infinite-memory is necessary for $\mathcal{P}_2$.*

[8] Note that a single response Rp_i suffices to answer all pending requests Rq_i, in the same spirit as for priorities in the *parity-response* objective.

[9] The results presented here are based on the best known complexity in $\mathcal{O}(|V|^2)$ for solving Büchi games [8], and not on the previously best known complexity in $\mathcal{O}(|V| \cdot |E|)$ originally used in [31,9].

We will now establish a polynomial-time reduction from multi-dimension DirBndWP and BndWP games (or equivalently, DirBndPR and BndPR games) to DirBndRR and BndRR games respectively. The crux is to consider $n \cdot \frac{d}{2}$ pairs of requests and responses, so that a request is made when an odd priority occurs and the corresponding response is the occurrence of a $\preceq$-smaller priority. Thanks to Theorem 17, we thus obtain an EXPTIME algorithm for multi-dimension (Dir)BndWP and (Dir)BndPR games.

Lemma 18. *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, $p_1, \dots, p_n$ be n priority functions. Let $b = |V| \cdot 2^{n \cdot \frac{d}{2}} \cdot n \cdot \frac{d}{2}$.*

1. *Let Ω be the objective $\cap_{m=1}^{n} \mathsf{DirBndPR}(p_m)$ or $\cap_{m=1}^{n} \mathsf{DirBndWP}(p_m)$. Deciding the winner from v_0 in the game (G, Ω) can be done in $\mathcal{O}(b^2)$ time and exponential-memory strategies are sufficient for both players.*
2. *Let Ω be the objective $\cap_{m=1}^{n} \mathsf{BndPR}(p_m)$ or $\cap_{m=1}^{n} \mathsf{BndWP}(p_m)$. Deciding the winner from v_0 in the game (G, Ω) can be done in $\mathcal{O}(|V| \cdot b^2)$ time, exponential-memory strategies are sufficient for $\mathcal{P}_1$ and infinite-memory is necessary for $\mathcal{P}_2$.*

Proof. In this proof, we only consider the objective $\Omega = \cap_{m=1}^{n} \mathsf{DirBndPR}(p_m)$ (resp. $\cap_{m=1}^{n} \mathsf{BndPR}(p_m)$) as $\cap_{m=1}^{n} \mathsf{(Dir)BndPR}(p_m) = \cap_{m=1}^{n} \mathsf{(Dir)BndWP}(p_m)$ by Proposition 7 Item 6.

We prove this lemma by encoding the objective Ω as a DirBndRR (resp. BndRR) objective. Intuitively, for any dimension, one wants to response to an odd priority by a $\preceq$-smaller priority. Formally, we define for each dimension m and each *odd* priority c the following sets of vertices: $Rq_{m,c} = \{v \in V \mid p_m(v) = c\}$ and $Rp_{m,c} = \{v \in V \mid p_m(v) \preceq_m c\} = \{v \in V \mid p_m(v) \in \{0, 2, \dots, c-1\}\}$. Clearly, $\mathcal{P}_1$ has a winning strategy for the objective Ω from v_0 if and only if $\mathcal{P}_1$ has a winning strategy from v_0 for the DirBndRR (resp. BndRR) objective considering the $n \cdot \frac{d}{2}$ pairs $(Rq_{m,c}, Rp_{m,c})$. Applying Theorem 17 with $r = n \cdot \frac{d}{2}$ we get the complexity and memory results stated in Lemma 18, except the necessity of infinite-memory for $\mathcal{P}_2$. Recall that the latter property already holds in one-dimension BndPR and BndWP games (see Theorem 10). ◻

Equalities between objectives. The next lemma gives the last two items of Theorem 16. The key ingredient is the bound given in Theorem 17 for (Dir)BndRR games, and by extension, for (Dir)BndPR and (Dir)BndWP games thanks to the reduction established in Lemma 18. The rest follows the same lines as in the one-dimension case, i.e., it builds upon the inclusions and equalities presented in Proposition 7.

Lemma 19. *Let $b = |V| \cdot 2^{n \cdot \frac{d}{2}} \cdot n \cdot \frac{d}{2}$. For all $\lambda \geq b$, $\lambda' \geq b \cdot \frac{d}{2}$, the winning sets for the following objectives are all equal: $\cap_{m=1}^{n} \mathsf{BndPR}(p_m)$, $\cap_{m=1}^{n} \mathsf{FixPR}(\lambda, p_m)$, $\cap_{m=1}^{n} \mathsf{BndWP}(p_m)$, and $\cap_{m=1}^{n} \mathsf{FixWP}(\lambda', p_m)$. The same equalities also hold for the direct variants* (Dir).

Proof. By Item 1 of Theorem 17 and the reduction established in Lemma 18, the winning set for the objective $\cap_{m=1}^{n} \mathsf{BndPR}(p_m)$ is equal to the winning set for the objective $\cap_{m=1}^{n} \mathsf{FixPR}(\lambda, p_m)$ with $\lambda = |V| \cdot 2^{n \frac{d}{2}} \cdot n \cdot \frac{d}{2}$. Now, as in the one dimension case, the other equalities follow from Proposition 7 (Items 3, 4, 5, 6). We get the equalities for the direct variant with the same proof. ◻

Lower bound on complexity. To prove the EXPTIME-hardness of objective (Dir)BndWP (and equivalently, of objective (Dir)BndPR), we establish a reduction from the *membership problem for alternating polynomial-space Turing machines (APTMs)* [5]. Our proof is adapted from the reduction presented in [6, Lemma 23] in the related context of window mean-payoff games. Similar techniques have been used for request-response games [10]. Since technical details are similar to [6, Lemma 23], we only include here a high-level sketch of the reduction. The main change is the way we open and close windows: whereas weights were used for window mean-payoff games, we need here to emulate the same actions with adapted priorities. Interestingly, our proof also shows EXPTIME-hardness of the fixed variants, (Dir)FixWP and (Dir)FixPR. Furthermore, the hardness already holds with only three priorities ($d = 2$).

Lemma 20. *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, $p_1, \dots, p_n$ be n priority functions. Let Ω be an objective $\Omega = \cap_{m=1}^{n} \Omega_m$ such that $\forall\, m$, $\Omega_m =$ (Dir)BndPR(p_m) (resp.* (Dir)BndWP(p_m), (Dir)FixWP(λ, p_m), (Dir)FixPR(λ, p_m)*). Deciding the winner from v_0 in the game (G, Ω) is* EXPTIME*-hard even if for all $m \in \{1, \dots, n\}$, $p_m \colon V \to \{0, 1, 2\}$.*

Proof (Sketch). Given an APTM $\mathcal{M}$ and a word $\zeta \in \{0,1\}^*$, such that the tape contains at most $f(|\zeta|)$ cells, where f is a polynomial function, the membership problem asks to decide if $\mathcal{M}$ accepts ζ. It is well-known to be EXPTIME-hard [5]. We first show how to reduce this problem to deciding if $\mathcal{P}_1$ has a winning strategy in a game G with an objective $\Omega = \cap_{m=1}^{n} \mathsf{BndWP}(p_m)$. We discuss the other objectives later as the corresponding results are easily obtained with the same skeleton.

We build the 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 $h \in \{1, 2, \ldots, f(|\zeta|)\}$, we have two dimensions, $(h, 0)$ and $(h, 1)$. The game starts in a vertex q_{in} that initializes those dimensions in order to encode the contents of the APTM tape: if the cell h contains 1 (resp. 0), then vertex q_{in} has priority 1 on dimension $(h, 1)$ (resp. 0) and priority 0 on dimension $(h, 0)$ (resp. 1). In terms of windows, this means that when we start the game, we open a window in the dimensions that correspond to the actual contents of the tape. The goal for $\mathcal{P}_1$ is now to correctly simulate the operation of the APTM by disclosing the correct symbols at each step.

The gadget used to simulate one step of the APTM is presented in Figure 9. When $\mathcal{P}_1$ reaches the vertex (q, h), he must disclose the symbol under the tape head: he can either claim that it contains a 0 and go to $(q, h, 0)_{\text{check}}$, which has priority 0 in dimension $(h, 0)$, or claim that the cell contains 1 and go to $(q, h, 1)_{\text{check}}$, which has priority 0 in dimension $(h, 1)$. Priorities in all other dimensions are always set to 2 (hence they do not open nor close any window). Intuitively, disclosing the correct symbol permits to close the open window on dimension (h, i) whereas lying about the symbol leaves this window open.

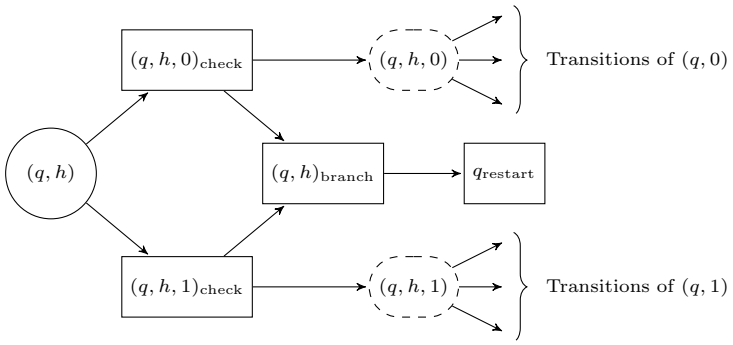

Fig. 9. Gadget ensuring a correct simulation of the APTM on tape cell h.

The job of $\mathcal{P}_2$ is to ensure that $\mathcal{P}_1$ was faithful: in vertex $(q, h, i)_{\text{check}}$, $\mathcal{P}_2$ has the choice to either let the simulation continue by going to the vertex (q, h, i) (where transitions of the APTM are encoded), or to go to $(q, h)_{\text{branch}}$ where the priority is 0 in all dimensions except for $(h, 0)$ and $(h, 1)$. This means that if $\mathcal{P}_2$ decides to branch, all windows will be closed if $\mathcal{P}_1$ was faithful, whereas an open window will remain on dimension (h, i) if $\mathcal{P}_1$ lied.

After $(q, h)_{\text{branch}}$, the game goes to q_{restart}, a vertex where the priority is 2 in all dimensions and that contains a self-loop: this implies that if $\mathcal{P}_1$ lied, $\mathcal{P}_2$ can take profit of this vertex to create an arbitrarily large open window. From q_{restart}, $\mathcal{P}_2$ additionally has the possibility to restart the game by going back to q_{in} where the contents of the tape are encoded again for a new round of simulation.

Finally, the game also contains a vertex q_{acc} that represents the accepting state of the APTM. To force $\mathcal{P}_1$ to go toward this vertex, we use an additional dimension that sees priority 1 at the beginning of the game (thus opening a window) and for which q_{acc} and $(q, h)_{\text{branch}}$ are the only vertices with priority 0 (hence the only ones able to close the open window). After visiting q_{acc}, the game goes to q_{restart}. Observe that the total number of dimensions used in our game is $(2 \cdot f(|\zeta|) + 1)$.

Recall that we consider the objective $\Omega = \cap_{m=1}^{n} \mathsf{BndWP}(p_m)$. Let $|\mathcal{C}|$ represent the size of the configuration graph of the APTM $\mathcal{M}$. It is known that $|\mathcal{C}|$ is a bound on the length of an accepting run in $\mathcal{M}$. In our

game, this induces that if q_{acc} can be reached, it can be reached in strictly less than $\lambda = 3 \cdot |\mathcal{C}| + 3$ steps. We claim that $\mathcal{P}_1$ has a winning strategy in our game if and only if the APTM $\mathcal{M}$ accepts the word ζ.

First, assume that ζ is accepted by $\mathcal{M}$. Then, $\mathcal{P}_1$ has a winning strategy consisting in always revealing the correct symbol. Indeed, either $\mathcal{P}_2$ never branches to a vertex $(q, h)_{\text{branch}}$ and the play reaches q_{acc} and then q_{restart} with all windows closed in strictly less than λ steps; or $\mathcal{P}_2$ decides to branch to $(q, h)_{\text{branch}}$ but since $\mathcal{P}_1$ never lied, all windows are closed when the play reaches q_{restart} (again in strictly less than λ steps). In both cases, the game can continue for another round of simulation in the same manner, yielding a play that is necessarily winning for Ω since it accepts a λ-good decomposition in each dimension.

Second, assume that ζ is not accepted by $\mathcal{M}$. If $\mathcal{P}_1$ never cheats, then $\mathcal{P}_2$ can use the strategy that emulates the path in the run tree of $\mathcal{M}$ that never reaches the accepting state (this path exists since ζ is not accepted). Thus, the window in the last dimension can be kept open for an arbitrarily long time in order to make the round losing for $\mathcal{P}_1$. Following this, $\mathcal{P}_2$ can restart the game and repeat this strategy. Note that in order to prevent $\mathcal{P}_1$ from winning for objective $\Omega = \cap_{m=1}^{n} \mathsf{BndWP}(p_m)$, it is necessary for $\mathcal{P}_2$ to keep restarting continuously and to increase the size of the open window at each round. The reason is essentially identical to the one in Example 8: (i) $\mathcal{P}_2$ must repeat losing rounds forever in order to counteract the prefix-independence of Ω, (ii) rounds must be of increasing length to prevent the existence of a bound on the size of the windows. Now, if $\mathcal{P}_1$ does cheat, $\mathcal{P}_2$ can branch to $(q, h)_{\text{branch}}$ and reach q_{restart} with an open window, here again creating an arbitrarily long open window, resulting in a losing round. Again, this can be repeated forever while increasing the size of open windows. Hence in both cases, $\mathcal{P}_1$ has no winning strategy in G, which proves the correctness of our reduction.

It remains to discuss the adaptations needed for the other objectives. First observe that the right-to-left direction of the equivalence (i.e., case ζ not accepted) holds for any other objective X claimed in Lemma 20 since $X \subseteq \cap_{m=1}^{n} \mathsf{BndWP}(p_m)$ by the various inclusions of Proposition 7. For the other direction (i.e., case ζ accepted), observe that the strategy of $\mathcal{P}_1$ that consists in always disclosing the correct symbol ensures that all windows close within $(\lambda - 1)$ steps (with the λ defined above): hence this strategy is also winning for $\cap_{m=1}^{n} \mathsf{FixWP}(\lambda, p_m)$. The hardness also holds for the direct variants $\mathsf{DirBndWP}$ and $\mathsf{DirFixWP}$ as this strategy never produces any λ-bad window. Finally, the inclusions of Proposition 7 suffice to see that this strategy is also winning for all PR variants (with the same λ for the fixed variants). Hence the proof holds for all claimed objectives. □

Lower bounds on memory. The last missing pieces to the proof of Theorem 16 are the exponential lower bounds on memory. Recall that for $\mathcal{P}_2$ in *undirect* bounded WP or PR games, we already proved that infinite memory is necessary in Example 8. The next lemma covers all remaining cases and establishes lower bounds matching the upper bounds granted by Lemma 18. To achieve this, we prove a polynomial-time reduction from *generalized reachability games* [17] to multi-dimension (Dir)BndWP and (Dir)BndPR games. Since the former games are known to require exponential memory for both players, the reduction yields the desired lower bounds. A similar reduction is presented for window mean-payoff games in [6]. Interestingly, the same technique also works for multi-dimension (Dir)FixWP and (Dir)FixPR games.

Let us sketch the reduction from $\mathsf{GenReach}$ to multi-dimension FixWP games (the other cases are similar). Intuitively, if the generalized reachability objective asks to visits n different target sets, we will use n dimensions. We create a modified version of the game structure such that, at the start of the game, we see priority 1 in all dimensions, hence opening a window, and such that the only way to close the window in dimension $m \in \{1, \dots, n\}$ is to visit the m-th target set. We also modify the game by giving $\mathcal{P}_2$ the possibility to close all open windows and restart the game by opening new ones: this is necessary to ensure that the prefix-independence of objective FixWP cannot help $\mathcal{P}_1$ to win without visiting all target sets. Finally, we use the fact that if $\mathcal{P}_1$ has a winning strategy in a $\mathsf{GenReach}$ game with n targets, then he has one that wins in strictly less than $n \cdot |V|$ steps (i.e., edges), to define an appropriate window size $\lambda = 2 \cdot n \cdot |V|$ for which the reduction to objective $\cap_{m=1}^{n} \mathsf{FixWP}(\lambda, p_m)$ on our modified game structure holds. As the reduction in Lemma 20, we only need three priorities here ($d = 2$).

Lemma 21. *Both players need exponential memory to win in games (G, Ω) where $\Omega = \cap_{m=1}^{n} \Omega_m$ such that $\forall\, m$, $\Omega_m = (\mathsf{Dir})\mathsf{BndPR}(p_m)$ (resp. $(\mathsf{Dir})\mathsf{BndWP}(p_m)$, $(\mathsf{Dir})\mathsf{FixWP}(\lambda, p_m)$, $(\mathsf{Dir})\mathsf{FixPR}(\lambda, p_m)$) even if for all $m \in \{1, \dots, n\}$, $p_m \colon V \to \{0, 1, 2\}$.*

Proof. We use a polynomial reduction from generalized reachability games for which it is known that both players need exponential memory to win [17]. We begin with the case $\Omega = \cap_{m=1}^{n} \mathsf{FixWP}(\lambda, p_m)$ and discuss the other cases at the end of the proof.

Let $G^r = (V_1^r, V_2^r, E^r)$ be a game structure, $U_1, \dots, U_n \subseteq V^r$ be target sets, and v_0 be the initial vertex. Let us recall that in the game (G^r, Ω^r) where $\Omega^r = \mathsf{GenReach}(U_1, \dots, U_n)$, if $\mathcal{P}_1$ has a winning strategy from v_0 then there exists one which ensures visiting all sets U_m in strictly less than $n \cdot |V^r|$ steps [17].

We build a game structure $G = (V_1, V_2, E)$ as follows. We define $V = V_1^r$ and $V_2 = V_2^r \cup V_{\text{branch}} \cup \{v_{\text{restart}}\}$ such that $V_{\text{branch}} = \{b_{v,v'} \mid (v, v') \in E^r\}$ and v_{restart} is a new vertex that is the initial vertex in G. We define E as the set of edges such that $(v, b_{v,v'}), (b_{v,v'}, v'), (b_{v,v'}, v_{\text{restart}}) \in E$ for all $(v, v') \in E^r$, and $(v_{\text{restart}}, v_0) \in E$. That is, we split each edge of E^r into two edges such that $\mathcal{P}_2$ can decide in the central vertex $b_{v,v'}$ to branch to v_{restart} or to continue as in G^r, and there is an edge from the new initial vertex v_{restart} of G to the old initial vertex v_0 of G^r. We define n priority functions as follows: for each $m \in \{1, \dots, n\}$, we let $p_m(v_{\text{restart}}) = 0$, $p_m(b_{v,v'}) = 2$ for all $b_{v,v'} \in V_{\text{branch}}$, $p_m(v_0) = 0$ if $v_0 \in U_m$ and 1 otherwise, and for all the other vertices v of V, we let $p_m(v) = 0$ if $v \in U_m$ and 2 otherwise. Notice that each vertex has only even priorities except v_0 which has odd priority $p_m(v_0) = 1$ whenever $v_0 \notin U_m$.

We claim that $\mathcal{P}_1$ has a winning strategy for objective $\Omega = \cap_{m=1}^{n} \mathsf{FixWP}(\lambda, p_m)$ in G with $\lambda = 2 \cdot n \cdot |V^r|$ if and only if he has a winning strategy for the generalized reachability objective Ω^r in G^r.

We first prove the left-to-right implication. Consider a winning strategy σ_1 of $\mathcal{P}_1$ in (G, Ω) from v_{restart}. By definition σ_1 ensures victory against any strategy of $\mathcal{P}_2$. In particular, it must win against a strategy σ_2 that plays in rounds, such that at each round, it chooses edges $(b_{v,v'}, v')$ during the first $(\lambda - 1)$ steps of the round, then chooses edge $(b_{v,v'}, v_{\text{restart}})$ and starts a new round. Consequently, strategy σ_1 must eventually be able to close all windows in $(\lambda - 1)$ steps without using the priorities 0 from v_{restart}, otherwise $\mathcal{P}_2$ can create infinitely-many λ-bad windows and win the game. The only way to achieve this is to visit all target sets U_m within $(\lambda - 1)$ steps. Consider the point from which σ_1 closes all λ-windows (it exists since σ_1 wins for Ω). Observe that from this point on, this strategy can be mimicked in G^r as during the first $(\lambda - 1)$ steps of the round, state v_{restart} is not visited, hence there is a bijection between histories in both games. By construction, the mimicking strategy σ_1^r ensures that all target sets are visited in at most $(n \cdot |V^r| - 1)$ steps (as we get rid of the additional vertices of V). Hence it is winning for Ω^r.

Second, we prove the right-to-left implication. Consider now a winning strategy σ_1^r of $\mathcal{P}_1$ in (G^r, Ω^r) from v_0 that ensures reaching all sets U_m in strictly less than $n \cdot |V^r|$ steps (w.l.o.g.). We define the strategy σ_1 of $\mathcal{P}_1$ in (G, Ω) that mimics σ_1^r and each time v_{restart} is reached restarts mimicking σ_1^r from v_0. Let π be a play consistent with σ_1. For all dimension $m \in \{1, \dots, n\}$, for all position $j \geq 0$, we must prove that there exists $l \in \{0, 1, \dots, \lambda - 1\}$ such that $\pi[j + l]$ is the $\preceq$-smallest priority in $\pi[j, j + l]$. Since we only have priorities in $\{0, 1, 2\}$ and v_0 is the only vertex which can have priority 1, this boils down to checking that each position j such that $\pi[j] = v_0$ is followed within $(\lambda - 1)$ steps by a position $(j + l)$ such that $p_m(\pi[j + l]) = 0$. This is obviously the case if there exists $l \leq \lambda - 1$ such that $\pi[j + l] = v_{\text{restart}}$. Otherwise, as σ_1 mimics σ_1^r and each edge of E^r has been split in G into two edges, $\pi[j] = v_0$ must be followed by a vertex $v \in U_m$ in strictly less than $2 \cdot n \cdot |V^r| = \lambda$ steps, by definition of σ_1^r. Hence, there exists $l < \lambda$ such that $p_m(\pi[j + l]) = 0$, which proves that $\pi \in \Omega$ and establishes the right-to-left implication.

Finally, we discuss why this reduction also holds for the other objectives mentioned in Lemma 21. First, the same proof holds for the direct variant DirFixWP: the left-to-right implication is obvious since this objective is included in FixWP (Proposition 7, Item 1), and the right-to-left implication actually yields a strategy σ_1 which never produces any λ-bad window, so it also wins for DirFixWP. Second, the proof also holds for the bounded variants BndWP and DirBndWP. Indeed, for the left-to-right implication, the fact that strategy σ_1 must be able to close all windows (hence reach all target sets) suffice (whatever the number of steps taken), whereas the right-to-left implication is obvious since the fixed variants are included in the bounded ones by Proposition 7, Item 2. Third, all corresponding PR variants can be obtained through the

equality of the bounded variants for PR and WP (Proposition 7, Item 6), and the equality of the fixed variants for PR and WP in the particular case where $d = 2$ (Lemma 9), as in this reduction. □

For the reader's interest, we complement Lemma 21 with an example illustrating the need for exponential memory for $\mathcal{P}_1$ in FixWP games (it also works for the other objectives of Lemma 21).

Example 22. Consider the family of game structures depicted in Figure 10. This family is parameterized by $n \in \mathbb{N}_0$ and is inspired by a similar one proposed in [12] for a different context (i.e., energy games). For each of these games structures, the number of vertices is linear in n ($|V| = 6n$) and we define $2n$ priority functions in the following way: for all $i \in \{1, \dots, n\}$ and for all $m \in \{1, \dots, 2n\}$, $p_m(v_i) = p_m(u_i) = 2$,

$$p_m(v_{i,L}) = \begin{cases} 1 & \text{if } m = 2i-1 \\ 2 & \text{otherwise} \end{cases}, \qquad p_m(v_{i,R}) = \begin{cases} 1 & \text{if } m = 2i \\ 2 & \text{otherwise} \end{cases},$$

$$p_m(u_{i,L}) = \begin{cases} 0 & \text{if } m = 2i-1 \\ 2 & \text{otherwise} \end{cases}, \qquad p_m(u_{i,R}) = \begin{cases} 0 & \text{if } m = 2i \\ 2 & \text{otherwise} \end{cases}.$$

Let $\Omega = \cap_{m=1}^{2n} \mathsf{FixWP}(3n, p_m)$ be the objective of $\mathcal{P}_1$. In order to prevent $(3n)$-bad windows, $\mathcal{P}_1$ has to choose $u_{i,L}$ (resp. $u_{i,R}$) whenever $\mathcal{P}_2$ chooses $v_{i,L}$ (resp. $v_{i,R}$). Hence in order to prevent outcomes with infinitely-many $(3n)$-bad windows, $\mathcal{P}_1$ must be able to record 2^n different histories from v_1 to u_1. This obviously requires exponential memory in n, hence in the size of the game. ◁

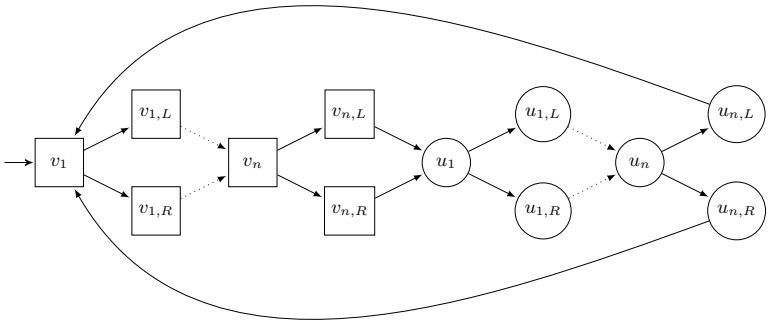

Fig. 10. Family of multi-dimension games requiring exponential memory for $\mathcal{P}_1$ for objective FixWP with $\lambda = 3n$.

5.2 Fixed variants

As in the one-dimension setting, some differences arise for the fixed variants. While both PR and WP variants prove to be EXPTIME-complete, the situation is slightly better for WP variants as we obtain an algorithm exponential in both the number of dimensions n and the binary encoding of λ, whereas for PR we also get an additional exponential in the largest priority d (which can be as large as the game structure). This distinction may be of importance in practical applications. We study the PR objective in Theorem 23 and the WP one in Theorem 24.

Parity-response objectives. To establish an exponential-time algorithm for multi-dimension DirFixPR (resp. FixPR) games, we reduce those games to safety (resp. co-Büchi) games on an exponentially-larger game structure. Our reduction is in the same spirit[10] as the one for Theorem 12, Item 2 (i.e., case d fixed in

[10] Note that the other algorithm suggested in Theorem 12, Item 3 and exponential in λ is not interesting here, since λ can be exponential before the fixed variant becomes equivalent to the bounded one (Lemma 19), hence this algorithm would take *doubly*-exponential time.

the one-dimension setting). That is, the extended structure encodes for each odd priority in each dimension, the number of steps since seeing the odd priority without seeing a $\preceq$-smaller priority in the meantime. The complexity and memory lower bounds follow from Lemma 20 and Lemma 21.

Theorem 23. *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, $p_1, \ldots, p_n$ be n priority functions, and Ω be the objective $\cap_{m=1}^{n} \mathsf{DirFixPR}(\lambda, p_m)$ (resp. $\cap_{m=1}^{n} \mathsf{FixPR}(\lambda, p_m)$) for $\lambda \in \mathbb{N}_0$. Deciding the winner in (G, Ω) from v_0 is* EXPTIME*-complete with an algorithm in $\mathcal{O}((|V|+|E|)\cdot\lambda^{\frac{d}{2}\cdot n})$ (resp. $\mathcal{O}(|V|^2\cdot\lambda^{d\cdot n})$) time, and exponential-memory strategies with $\mathcal{O}(\lambda^{\frac{d}{2}\cdot n})$ memory are sufficient for both players. Moreover, exponential-memory strategies are necessary for both players.*

Proof. In order to prove EXPTIME-membership and that exponential-memory strategies are sufficient, we use the same construction as in the proof of Theorem 12, Item 2, except that we have to deal with multiple dimensions instead of one. More precisely, for the undirect variant FixPR, we construct a new game structure G' from G such that the set V' of vertices is equal to $V \times \{\bot, 0, 1, \ldots, \lambda - 2\}^{\frac{d}{2}\cdot n} \cup \beta_V$ with $\beta_V = \{\beta_v \mid v \in V\}$, and the set E' of edges is defined as in the one-dimension case with the different dimensions considered component-wise. We define $U' = V' \setminus \beta_V$ and $\Omega' = \mathsf{CoBuchi}(U')$. With this construction, winning in the initial game (G, Ω) is equivalent to winning in the new game (G', Ω'). For the direct variant DirFixPR, the construction is similar except that β_V is reduced to a unique absorbing vertex β and $\Omega' = \mathsf{Safe}(U')$. In both constructions, the size of the game G' is exponential in the size of the original game G, with $\mathcal{O}(|V'|) = \mathcal{O}(|V| \cdot \lambda^{\frac{d}{2}\cdot n})$ and $\mathcal{O}(|E'|) = \mathcal{O}(|E| \cdot \lambda^{\frac{d}{2}\cdot n})$. The stated complexities and memory requirements follow.

Finally, EXPTIME-hardness of the decision problem follows from Lemma 20 and the necessity of exponential memory for both players follows from Lemma 21. □

Window parity objectives. To conclude our study of PR and WP objectives, it remains to establish an exponential-time algorithm for multi-dimension DirFixWP (resp. FixWP) games. Again, we reduce those games to safety (resp. co-Büchi) games on an exponentially-larger game structure. Our reduction is here based on the one used in the one-dimension setting (Theorem 14). That is, the extended structure encodes, for each dimension, the minimum priority of the current window, and the number of steps in that window. The complexity and memory lower bounds follow from Lemma 20 and Lemma 21.

Theorem 24. *Let $G = (V_1, V_2, E)$ be a game structure, v_0 be an initial vertex, $p_1, \ldots, p_n$ be n priority functions, and Ω be the objective $\cap_{m=1}^{n} \mathsf{DirFixWP}(\lambda, p_m)$ (resp. $\cap_{m=1}^{n} \mathsf{FixWP}(\lambda, p_m)$) for $\lambda \in \mathbb{N}_0$. Deciding the winner in (G, Ω) from v_0 is* EXPTIME*-complete with an algorithm in $\mathcal{O}((|V| + |E|) \cdot (d \cdot \lambda)^n)$ (resp. $\mathcal{O}(|V|^2 \cdot (d \cdot \lambda)^{2\cdot n})$) time, and exponential-memory strategies are sufficient for both players with $\mathcal{O}((d \cdot \lambda)^n)$ memory. Moreover, exponential-memory strategies are necessary for both players.*

Proof. To establish EXPTIME-membership and that exponential-memory strategies are sufficient, we proceed as in the one-dimension case (see the proof of Theorem 14). For the undirect variant FixWP, we construct from G a new game structure G' such that $V' = V \times (\{0, \ldots, d\} \times \{0, \ldots, \lambda-1\})^n \cup \beta_V$ with $\beta_V = \{\beta_v \mid v \in V\}$, and the set E' of edges is defined as in the one-dimension case with the different dimensions considered component-wise. Again $U' = V' \setminus \beta_V$ and $\Omega' = \mathsf{CoBuchi}(U')$. For the direct variant DirFixWP, β_V is replaced by $\{\beta\}$ and $\Omega' = \mathsf{Safe}(U')$. In both cases, the size of G' is exponential in the size of G, with $\mathcal{O}(|V'|) = \mathcal{O}(|V| \cdot (d \cdot \lambda)^n)$ and $\mathcal{O}(|E'|) = \mathcal{O}(|E| \cdot (d \cdot \lambda)^n)$. The stated complexities and memory requirements follow. To conclude, EXPTIME-hardness follows from Lemma 20 and the necessity of exponential memory follows from Lemma 21. □

Acknowledgments. We express our gratitude to Jean-François Raskin (Université libre de Bruxelles, Belgium) and Martin Zimmermann (Saarland University, Germany) for insightful discussions.

References

1. C. Baier, J. Klein, S. Klüppelholz, and S. Wunderlich. Weight monitoring with linear temporal logic: complexity and decidability. In *Proc. of CSL-LICS*, pages 11:1–11:10. ACM, 2014.

2. C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Trans. Database Syst.*, 5(3):241–259, 1980.
3. V. Bruyère, Q. Hautem, and M. Randour. Window parity games: an alternative approach toward parity games with time bounds. In *Proc. of GandALF*, EPTCS 226, pages 135–148, 2016.
4. N. Buhrke, H. Lescow, and J. Vöge. Strategy construction in infinite games with Streett and Rabin chain winning conditions. In *Proc. of TACAS*, LNCS 1055, pages 207–224. Springer, 1996.
5. A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.
6. K. Chatterjee, L. Doyen, M. Randour, and J.-F. Raskin. Looking at mean-payoff and total-payoff through windows. *Information and Computation*, 242:25–52, 2015.
7. K. Chatterjee and N. Fijalkow. Infinite-state games with finitary conditions. In *Proc. of CSL*, LIPIcs 23, pages 181–196. Schloss Dagstuhl - LZI, 2013.
8. K. Chatterjee and M. Henzinger. Efficient and dynamic algorithms for alternating Büchi games and maximal end-component decomposition. *J. ACM*, 61(3):15:1–15:40, 2014.
9. K. Chatterjee, T. A. Henzinger, and F. Horn. Finitary winning in ω-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009.
10. K. Chatterjee, T. A. Henzinger, and F. Horn. The complexity of request-response games. In *Proc. of LATA*, LNCS 6638, pages 227–237. Springer, 2011.
11. K. Chatterjee, T. A. Henzinger, and N. Piterman. Generalized parity games. In *Proc. of FOSSACS*, LNCS 4423, pages 153–167. Springer, 2007.
12. K. Chatterjee, M. Randour, and J.-F. Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3-4):129–163, 2014.
13. S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games? In *Proc. of LICS*, pages 99–110. IEEE Computer Society, 1997.
14. E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *Proc. of FOCS*, pages 328–337, 1988.
15. E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proc. of FOCS*, pages 368–377. IEEE Computer Society, 1991.
16. E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In *Proc. of CAV*, LNCS 697, pages 385–396. Springer, 1993.
17. N. Fijalkow and F. Horn. The surprizing complexity of generalized reachability games. *CoRR*, abs/1010.2420, 2010.
18. N. Fijalkow and M. Zimmermann. Parity and Streett games with costs. *LMCS*, 10(2), 2014.
19. E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, LNCS 2500. Springer, 2002.
20. F. Horn. Streett games on finite graphs. In GDV'05, 2005.
21. N. Immerman. Number of quantifiers is better than number of tape cells. *J. Comput. Syst. Sci.*, 22(3):384–406, 1981.
22. M. Jurdzinski. Deciding the winner in parity games is in UP $\cap$ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.
23. M. Jurdzinski. Small progress measures for solving parity games. In *Proc. of STACS*, LNCS 1770, pages 290–301. Springer, 2000.
24. M. Jurdzinski, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. Comput.*, 38(4):1519–1532, 2008.
25. O. Kupferman, N. Piterman, and M. Y. Vardi. From liveness to promptness. *FMSD*, 34(2):83–103, 2009.
26. D. A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.
27. N. Piterman and A. Pnueli. Faster solutions of Rabin and Streett games. In *Proc. of LICS*, pages 275–284. IEEE Computer Society, 2006.
28. M. Randour. Automated synthesis of reliable and efficient systems through game theory: A case study. In *Proc. of ECCS 2012*, Springer Proceedings in Complexity XVII, pages 731–738. Springer, 2013.
29. S. Schewe. Solving parity games in big steps. In *Proc. of FSTTCS*, LNCS 4855, pages 449–460. Springer, 2007.
30. W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.
31. N. Wallmeier, P. Hütten, and W. Thomas. Symbolic synthesis of finite-state controllers for request-response specifications. In *Proc. of CIAA*, LNCS 2759, pages 11–22. Springer, 2003.
32. A. Weinert and M. Zimmermann. Easy to win, hard to master: Optimal strategies in parity games with costs. In *Proc. of CSL*, LIPIcs 62, pages 31:1–31:17. Schloss Dagstuhl - LZI, 2016.
33. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.