Information and Computation 272 (2020) 104501

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Robust worst cases for parity games algorithms ☆

Massimo Benerecetti, Daniele Dell'Erba, Fabio Mogavero *

Università degli Studi di Napoli Federico II, Napoli, Italy

ARTICLE INFO

Article history:
Received 28 February 2018
Received in revised form 4 December 2018
Accepted 15 February 2019
Available online 12 December 2019

Keywords:
Parity games
Infinite-duration games on graphs
Formal methods

ABSTRACT

The McNaughton-Zielonka *divide et impera* algorithm is the simplest and most flexible approach available in the literature for determining the winner in a parity game. Despite its theoretical exponential worst-case complexity and the negative reputation as a poorly effective algorithm in practice, it has been shown to rank among the best techniques for solving such games. Also, it proved to be resistant to a lower bound attack, even more than the strategy improvements approaches, but finally Friedmann provided a family of games on which the algorithm requires exponential time. An easy analysis of this family shows that a simple memoization technique can help the algorithm solve the family in polynomial time. The same result can also be achieved by exploiting an approach based on the dominion-decomposition techniques proposed in the literature. These observations raise the question whether a suitable combination of dynamic programming and game-decomposition techniques can improve on the exponential worst case of the original algorithm. In this paper we answer this question negatively, by providing a robustly exponential worst case, showing that no possible intertwining of the above mentioned techniques can help mitigating the exponential nature of the *divide et impera* approaches. The resulting worst case is even more robust than that, since it serves as a lower bound for progress measures based algorithms as well, such as Small Progress Measure and its quasi-polynomial variant recently proposed by Jurdziński and Lazic.

1. Introduction

Parity games [2] are perfect-information two-player turn-based games of infinite duration, usually played on finite directed graphs. Their vertices, labeled by natural numbers called *priorities*, are assigned to one of two players, named *Even* and *Odd* or, simply, 0 and 1, respectively. A play in the game is an infinite sequence of moves between vertices and it is said to be winning for player 0 (*resp.*, 1), if the maximal priority encountered infinitely often along the play is *even* (*resp.*, odd). These games have been extensively studied in the attempt to find efficient solutions to the problem of determining the winner. From a complexity theoretic perspective, this decision problem lies in NPTime ∩ CoNPTime [3], since parity games are *memoryless determined* [2,4–6]. It has been even proved to belong to UPTime ∩ CoUPTime [7] and, very recently, to be solvable in quasi-polynomial time [8]. They are the simplest class of games in a wider family with similar complexities and containing, *e.g.*, *mean payoff games* [9,10], *discounted payoff games* [11], and *simple stochastic games* [12]. In fact, polynomial time reductions exist from parity games to the latter ones. However, despite being the most likely class among those games to admit a polynomial-time solution, the answer to the question whether such a solution exists still remains elusive. The

☆ This work is based on [1], which appeared in GandALF'17.

* Corresponding author.
E-mail address: fabio.mogavero@unina.it (F. Mogavero).

https://doi.org/10.1016/j.ic.2019.104501
0890-5401/

effort devoted to provide efficient solutions stems primarily from the fact that many problems in formal verification and synthesis can be reformulated in terms of solving parity games. Emerson, Jutla, and Sistla [3] have shown that computing winning strategies for these games is linear-time equivalent to solving the modal μCALCULUS model checking problem [13]. Parity games also play a crucial role in automata theory [14,4,15], where they can be applied to solve the complementation problem for alternating automata [16] and the emptiness of the corresponding nondeterministic tree automata [15]. These automata, in turn, can be used to solve the satisfiability and model checking problems for expressive logics, such as the modal [17] and alternating [18,19] μCALCULUS, ATL* [19,20], Strategy Logic [21–25], Substructure Temporal Logic [26,27], and fixed-point extensions of guarded first-order logics [28].

Previous exponential solutions essentially divide into two families. The first one collects procedures that attempt to directly build a winning strategy for one of the two players on the entire game. To such family belongs the *Small Progress Measure* approach by Jurdziński [29], which exploits the connection between the notions of progress measures [30] and winning strategies. A second approach in same vein is the *Strategy Improvement* algorithm by Jurdziński and Vöge [31], based on the idea of iteratively improving an initial, non necessarily winning, strategy. This technique has been then further refined in [32,33].

The second family gathers, instead, the approaches based on decomposing the solution of a game into the analysis of its subgames. To this family belong the so called *divide et impera* approaches led by the *Recursive* algorithm proposed by Zielonka [34], which adapts to parity games an earlier algorithm proposed by McNaughton for Muller games [35]. Intuitively, it decomposes the input game into subgames and solves them recursively. Using the Recursive algorithm as a back-end, and in the attempt to obtain a better upper bound, the *Dominion Decomposition* [36,37] and the *Big Step* [38,39] approaches were devised. Both share the idea of intertwining the recursive calls of the back-end with a preprocessing phase, applied to the current subgame, in search of a sufficiently small dominion for some player $\wp$, *i.e.*, a set of positions from where $\wp$ wins without ever exiting the set. The first technique does so by means of a brute force search, while the second one exploits a suitable variation of the Small Progress Measure procedure. A different direction has been followed recently within the decomposition-based family, that leads to a novel solution technique based on the notions of *quasi-dominions* and *priority promotion* [40–42]. The approach relies on a new procedure that finds dominions of arbitrary size, which proved to be quite efficient in practice and exhibits the best space complexity among the known solution algorithms, even better than the recently introduced quasi-linear space algorithms [43,44]. A related technique has been proposed in [45], where a refined notion of quasi-dominion, called *tangle*, is introduced.

The literature also suggests several heuristics to tune parity game solvers. One of the most successful ones decomposes the game into strongly-connected components (SCCs, for short) and solving it SCC-wise. *SCC-decomposition*, together with some other minor techniques such as removal of self-cycles and priority compression, can significantly improve the solution process, as discussed and empirically demonstrated in [46]. The same authors also show that, against the negative reputation as far as performance is concerned, the Recursive algorithm often stands out as the best solver among those proposed in the literature, particularly when paired with the SCC-decomposition heuristic. Despite having a quite straightforward exponential upper bound, this algorithm has resisted an exponential lower bound for more than ten years, until Friedmann [47] devised an indexed family of games that forces the algorithm to execute a number of recursive calls that grows exponentially with the index. The family is also resilient to the SCC-decomposition technique, since each subgame passed to a recursive call always forms a single SCC. On a closer look, however, the games proposed there force an exponential behavior by requiring the algorithm to repeatedly solve a small number of subgames, actually only a linear number of them. As a consequence, all those games are amenable to a polynomial-time solution, by simply providing the algorithm with a suitable memoization mechanism that prevents it from wasting computational resources on solving already solved subgames. For different reasons, also a dominion decomposition approach can break the lower bound easily, as most of the subgames of a game in the family contain a dominion of constant size.

These observations raise the question whether the Recursive algorithm admits an exponential lower bound robust enough to be resilient to a suitable memoization, SCC-decomposition, and dominion decomposition techniques. The difficulty here is that such a robust worst case should induce an exponential number of different subgames to prevent memoization from being of any help. At the same time, each of those subgames must contain a single SCC and only dominions of sufficiently large size to prevent both SCC-decomposition and dominion decomposition techniques from simplifying the game. In this paper, we answer positively the question, by providing a robust, and harder, worst case family that meets all the above requirements, thereby shading some light on the actual power of aforementioned techniques and sanctioning that no combination of them can indeed help improving the exponential lower bound of the *divide et impera* approaches.

A recent breakthrough [8] by Calude et al. proposes a succinct reduction from parity to reachability games based on a clever encoding of the sequences of priorities a player finds along a play. This allows for a mere quasi-polynomial blow up in the size of the underlying graph and sets the basis of the fixed-parameter tractability *w.r.t.* the number of priorities. The approach has been then considerably refined in [44], where these encodings are modeled as progress measures. A different compact encoding for progress measures, which leads to similar complexity results, as also been proposed in [43]. Using a novel complexity measure for parity games, called the *register-index*, the work in [48] also obtains a quasi-polynomial bound, by showing that parity games with a fixed register-index can be solved in polynomial time and, in addition, their index is always bounded by a logarithmic function of the number of positions. In [49] the authors use the separation approach of [50] to prove that all the above quasi-polynomial techniques are optimal and cannot be further improved. An unrelated quasi-polynomial upper bound has also been obtained in [51] for a refined version of the recursive algorithm. The

idea is based on the observation that, in any parity subgame, at most one of the two players' winning regions can be bigger than $n/2$, with n the number of positions in the subgame. This is, then, exploited to bound the depth of the search for winning regions of each generated subgame in such a way that the total number of recursive calls is quasi-polynomial in the number of positions and of priorities. Despite the theoretical relevance of this new ideas, preliminary experiments [52] suggest that, at least at present, the practical impact of these results do not match the theoretical one, as all exponential algorithms outperform, often by orders of magnitude, the current implementations of the quasi-polynomial ones, which do not scale beyond few hundred vertices. This evaluation is consistent with the fact that the new techniques essentially amount to clever and succinct encodings embedded within a brute force search, which makes matching quasi-polynomial worst cases quite easy to find. We prove, indeed, that our worst case family generates difficult instances for the succinct version [43] of the small progress measure algorithm as well.

The discussion above seems to suggest that the road to a polynomial solution may need to take another direction. Our work is, therefore, intended to evaluate the weaknesses of classic exponential algorithms, in the same vein as [53,54], where the authors study the pitfalls of existing exponential algorithms for graphs isomorphism, in spite of the fact that a quasi-polynomial, but impractical, algorithm exists [55]. We believe that a better understanding of the different issues of the known approaches may lead to progress in the quest for a polynomial algorithm.

2. Parity games

Let us first briefly recall the notation and basic definitions concerning parity games that expert readers can simply skip. We refer to [56][34] for a comprehensive presentation of the subject.

A two-player turn-based *arena* is a tuple $\mathcal{A} = \langle \mathrm{Ps}^0, \mathrm{Ps}^1, Mv\rangle$, with $\mathrm{Ps}^0 \cap \mathrm{Ps}^1 = \emptyset$ and $\mathrm{Ps} \triangleq \mathrm{Ps}^0 \cup \mathrm{Ps}^1$, such that $\langle \mathrm{Ps}, Mv\rangle$ is a finite directed graph. Ps^0 (*resp.*, Ps^1) is the set of positions of player 0 (*resp.*, 1) and $Mv \subseteq \mathrm{Ps} \times \mathrm{Ps}$ is a left-total relation describing all possible moves. A *path* in $\mathrm{V} \subseteq \mathrm{Ps}$ is an infinite sequence $\pi \in \mathrm{Pth}(\mathrm{V})$ of positions in V compatible with the move relation, *i.e.*, $(\pi_i, \pi_{i+1}) \in Mv$, for all $i \in \mathbb{N}$. A positional *strategy* for player $\wp \in \{0,1\}$ (we may, at times, use the symbol $\mathbb{B}$ to denote the set $\{0,1\}$ for convenience) on $\mathrm{V} \subseteq \mathrm{Ps}$ is a function $\sigma_\wp \in \mathrm{Str}^\wp(\mathrm{V}) \subseteq (\mathrm{V} \cap \mathrm{Ps}^\wp) \to \mathrm{V}$, mapping each $\wp$-position $v \in \mathrm{V} \cap \mathrm{Ps}^\wp$ to position $\sigma_\wp(v) \in \mathrm{V}$ compatible with the move relation, *i.e.*, $(v, \sigma_\wp(v)) \in Mv$. By $\mathrm{Str}^\wp(\mathrm{V})$ we denote the set of all $\wp$-strategies on V. A *play* in $\mathrm{V} \subseteq \mathrm{Ps}$ from a position $v \in \mathrm{V}$ *w.r.t.* a pair of strategies $(\sigma_0, \sigma_1) \in \mathrm{Str}^0(\mathrm{V}) \times \mathrm{Str}^1(\mathrm{V})$, called $((\sigma_0, \sigma_1), v)$-*play*, is a path $\pi \in \mathrm{Pth}(\mathrm{V})$ such that $\pi_0 = v$ and, for all $i \in \mathbb{N}$, if $\pi_i \in \mathrm{Ps}^0$, then $\pi_{i+1} = \sigma^0(\pi_i)$ else $\pi_{i+1} = \sigma^1(\pi_i)$.

A *parity game* is a tuple $\mathcal{D} = \langle \mathcal{A}, \mathrm{Pr}, \mathsf{pr}\rangle$, where $\mathcal{A}$ is an arena, $\mathrm{Pr} \subset \mathbb{N}$ is a finite set of priorities, and $\mathsf{pr} : \mathrm{Ps} \to \mathrm{Pr}$ is a *priority function* assigning a priority to each position. The priority function can be naturally extended to games and paths as follows: $\mathsf{pr}(\mathcal{D}) \triangleq \max_{v \in \mathrm{Ps}} \mathsf{pr}(v)$; for a path $\pi \in \mathrm{Pth}$, we set $\mathsf{pr}(\pi) \triangleq \limsup_{i \to \infty} \mathsf{pr}(\pi_i)$. A set of positions $\mathrm{V} \subseteq \mathrm{Ps}$ is a $\wp$-*dominion*, with $\wp \in \{0,1\}$, if there exists a $\wp$-strategy $\sigma_\wp \in \mathrm{Str}^\wp(\mathrm{V})$ such that, for all $\overline{\wp}$-strategies $\sigma_{\overline{\wp}} \in \mathrm{Str}^{\overline{\wp}}(\mathrm{V})$ and positions $v \in \mathrm{V}$, the induced $((\sigma_0, \sigma_1), v)$-play π has priority of parity $\wp$, *i.e.*, $\mathsf{pr}(\pi) \equiv_2 \wp$. In other words, $\sigma_\wp$ only induces on V plays whose maximal priority visited infinitely often has parity $\wp$. The *winning region* for player $\wp \in \{0,1\}$ in game $\mathcal{D}$, denoted by $\mathrm{Wn}^\wp_{\mathcal{D}}$, is the maximal set of positions that is also a $\wp$-dominion in $\mathcal{D}$. Since parity games are determined games [4], meaning that from each position one of the two players wins, the two winning regions of a game $\mathcal{D}$ form a partition of its positions, *i.e.*, $\mathrm{Wn}^0_{\mathcal{D}} \cup \mathrm{Wn}^1_{\mathcal{D}} = \mathrm{Ps}_{\mathcal{D}}$. By $\mathcal{D} \setminus \mathrm{V}$ we denote the maximal subgame of $\mathcal{D}$ with set of positions Ps' contained in $\mathrm{Ps} \setminus \mathrm{V}$ and move relation Mv' equal to the restriction of Mv to Ps'. The $\wp$-predecessor of V, in symbols $\mathsf{pre}^\wp(\mathrm{V}) \triangleq \{v \in \mathrm{Ps}^\wp : Mv(v) \cap \mathrm{V} \neq \emptyset\} \cup \{v \in \mathrm{Ps}^{\overline{\wp}} : Mv(v) \subseteq \mathrm{V}\}$, collects the positions from which player $\wp$ can force the game to reach some position in V with a single move. The $\wp$-attractor $\mathsf{atr}^\wp(\mathrm{V})$ generalizes the notion of $\wp$-predecessor $\mathsf{pre}^\wp(\mathrm{V})$ to an arbitrary number of moves. Thus, it corresponds to the least fix-point of that operator. When $\mathrm{V} = \mathsf{pre}^\wp(\mathrm{V})$, player $\wp$ cannot force any position outside V to enter this set. For such a V, the set of positions of the subgame $\mathcal{D} \setminus \mathrm{V}$ is precisely $\mathrm{Ps} \setminus \mathrm{V}$. When confusion cannot arise, we may abuse the notation and write $\mathcal{D}$ to mean its set of positions $\mathrm{Ps}_{\mathcal{D}}$.

3. The worst-case core family

As mentioned above, our goal is to define a new family of parity games that exhibits worst-case behaviors for a number of parity games solvers and, at the same time, is robust with respect to memoization and game decomposition techniques possibly employed by the solvers. We start by focusing on satisfying the memoization robustness requirement, the most difficult one to meet. We define below a simple family that forces the Recursive algorithm to solve an exponential number of different subgames. As we shall see in the following two sections, this family is already robust enough to provide a worst-case for the Recursive algorithm, with or without memoization, and for progress-measure based algorithms such as Small Progress Measure [29] and Succinct Progress Measure [43].

The *core family* $\{\mathcal{D}^k_{\mathrm{C}}\}_{k=1}^{\omega}$ of games is informally defined as follows. For each $k \in \mathbb{N}_+$, game $\mathcal{D}^k_{\mathrm{C}}$ contains $k+1$ gadgets, each one formed by three positions α_i, β_i, and γ_i, for $i \in [0,k]$. The positions β_i and γ_i, in gadget i, share the same priority i and opposite owners, namely player $i \bmod 2$ for β_i and $(i+1) \bmod 2$ for γ_i. The position α_i in the gadget has the same owner as the corresponding β_i. These positions are leading ones, having higher priorities than all the β's and γ's of the other gadgets. Positions within gadget i are connected as follows: α_i can only move to β_i; β_i can only move to γ_i; γ_i can choose either to move to β_i or to stay in γ_i itself. Two adjacent gadgets, of indexes i and $i+1$, are connected by only two moves: one from γ_i to α_{i+1} and one from β_{i+1} to α_i. Fig. 1 depicts game $\mathcal{D}^4_{\mathrm{C}}$. The gray portion in the figure represents the base game $\mathcal{D}^1_{\mathrm{C}}$ of the family, where the priorities of all its α-positions have been increased by 2, so as to comply with the

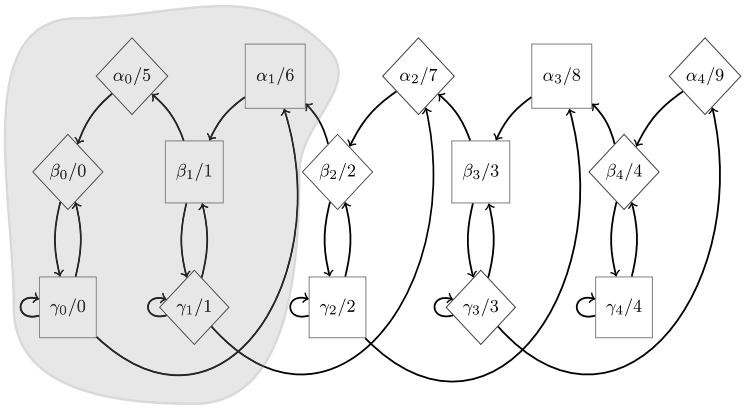

Fig. 1. Game $\mathfrak{D}_{\mathrm{C}}^{4}$ of the core family.

requirement that the α-positions have the higher priorities. Indeed, in general, given an index k, game $\mathfrak{D}_{\mathrm{C}}^{k+1}$ is obtained from $\mathfrak{D}_{\mathrm{C}}^{k}$ by increasing by $2(k \bmod 2)$ units the priorities of each α_i in the gadgets of $\mathfrak{D}_{\mathrm{C}}^{k}$ and adding a new gadget with index $k+1$ connected as in figure. The games in the core family are formally described by the following definition.

Definition 3.1 *(Core family).* The *core family* $\{\mathfrak{D}_{\mathrm{C}}^{k}\}_{k=1}^{\omega}$, where $\mathfrak{D}_{\mathrm{C}}^{k} \triangleq \langle \mathcal{A}, \mathrm{Pr}, \mathrm{pr}\rangle$, $\mathcal{A} \triangleq \langle \mathrm{Ps}^0, \mathrm{Ps}^1, Mv\rangle$, and $\mathrm{Pr} \triangleq [0, 2k+1+k \bmod 2]$, is defined as follows. For any index $k \geq 1$, the set of positions $\mathrm{Ps} \triangleq \{\alpha_i, \beta_i, \gamma_i : 0 \leq i \leq k\}$ of $\mathfrak{D}_{\mathrm{C}}^{k}$ is divided into three categories:

- α_i belongs to player $\wp \triangleq i \bmod 2$, *i.e.*, $\alpha_i \in \mathrm{Ps}^{\wp}$, and has priority $\mathrm{pr}(\alpha_i) \triangleq k+i+1+k \bmod 2$;
- β_i belongs to player $\wp \triangleq i \bmod 2$, *i.e.*, $\beta_i \in \mathrm{Ps}^{\wp}$, and has priority $\mathrm{pr}(\beta_i) \triangleq i$;
- γ_i belongs to player $\overline{\wp} \triangleq (i+1) \bmod 2$, *i.e.*, $\gamma_i \in \mathrm{Ps}^{\overline{\wp}}$, and has priority $\mathrm{pr}(\gamma_i) \triangleq i$.

Moreover, the moves from positions α_i, β_i, and γ_i, with $0 \leq i \leq k$, are prescribed as follows:

- α_i has a unique move to β_i, *i.e.*, $Mv(\alpha_i) = \{\beta_i\}$;
- β_i has one move to γ_i and one to α_{i-1}, if $i > 0$, *i.e.*, $Mv(\beta_i) = \{\gamma_i\} \cup \{\alpha_{i-1} : i > 0\}$;
- γ_i has one move to γ_i itself, one to β_i, and, if $i < k$, one to α_{i+1}, *i.e.*, $Mv(\gamma_i) = \{\beta_i, \gamma_i\} \cup \{\alpha_{i+1} : i < k\}$.

It is not hard to verify that, for any $k \in \mathbb{N}_+$, the game $\mathfrak{D}_{\mathrm{C}}^{k}$ is completely won by player $(k \bmod 2)$ and contains precisely $3(k+1)$ positions and $6k+4$ moves. As we shall see later in detail, the solution of each such games requires the Recursive algorithm to solve an exponential number of different subgames.

These core games form the backbone for a more general framework, consisting of an entire class of game families, with the property that each of them remains resilient to memoization techniques. Essentially, each game in any such family extends a core game. In order to define such a wider class, let us first establish what counts as a suitable extension of a core. Clearly, for a game $\mathfrak{D}$ to be an extension of $\mathfrak{D}_{\mathrm{C}}^{k}$, for some index $k \in \mathbb{N}_+$, it must contain $\mathfrak{D}_{\mathrm{C}}^{k}$ as a subgame. However, in order to prevent the recursive algorithm from disrupting the structure of the core game contained in $\mathfrak{D}$ while processing the game, we have to enforce some additional requirements. In particular, we need the algorithm to behave on $\mathfrak{D}$ virtually in the same way as it does on the core subgame. There is no unique way to ensure that. A simple solution is to require that all positions α_i still have the maximal priorities as in the core. Moreover, we require that no positions α_i and β_i have additional moves in $\mathfrak{D}$ *w.r.t.* those contained in the core. Finally, if γ_i can escape to some position v outside the core, then v does not have a higher priority, it has a move back to γ_i, and belongs to the opponent player *w.r.t.* γ_i. This condition ensures that no γ_i can decide to escape the core without being bounced back immediately by the opponent. The following definition makes the notion of extension precise.

Definition 3.2 *(Core extension).* An arbitrary parity game $\mathfrak{D} \in \mathrm{PG}$ is a *core extension* of $\mathfrak{D}_{\mathrm{C}}^{k}$, for a given index $k \in \mathbb{N}_+$, if the following four conditions hold:

1. $\mathfrak{D} \setminus \mathrm{P} = \mathfrak{D}_{\mathrm{C}}^{k}$, where $\mathrm{P} \triangleq \{v \in \mathfrak{D} : v \notin \mathfrak{D}_{\mathrm{C}}^{k}\}$;
2. $\mathrm{pr}_{\mathfrak{D}}(v) < \mathrm{pr}_{\mathfrak{D}}(\alpha_0)$, for all $v \in \mathrm{P}$;
3. $\{\alpha_i, \beta_i \in \mathfrak{D} : 0 \leq i \leq k\} \cap (Mv(\mathrm{P}) \cup Mv^{-1}(\mathrm{P})) = \emptyset$;
4. $v \in \mathrm{Ps}^{\wp}$, $\gamma_i \in Mv(v)$, and $\mathrm{pr}(v) \leq i$, for all $v \in \mathrm{P} \cap Mv(\gamma_i)$, $i \in [0, k]$, and $\wp \triangleq i \bmod 2$.

We shall denote with $\mathrm{PG}_{\mathrm{C}} \subseteq \mathrm{PG}$ the set of all core extensions of $\mathfrak{D}_{\mathrm{C}}^{k}$, for any index $k \in \mathbb{N}_+$.

Algorithm 1: Recursive algorithm.

signature $\mathsf{sol} : \mathsf{PG} \to_{\mathcal{D}} 2^{\mathsf{Ps}_{\mathcal{D}}} \times 2^{\mathsf{Ps}_{\mathcal{D}}}$
function $\mathsf{sol}(\mathcal{D})$
1 $(\mathcal{D}_{\mathrm{L}}, \wp) \leftarrow \mathsf{f}_{\mathrm{L}}(\mathcal{D})$
2 $(\mathsf{Wn}^0_{\mathrm{L}}, \mathsf{Wn}^1_{\mathrm{L}}) \leftarrow \mathsf{sol}(\mathcal{D}_{\mathrm{L}})$
3 **if** $\mathsf{pre}^{\overline{\wp}}_{\mathcal{D}}(\mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}}) \setminus \mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}} = \emptyset$ **then**
4 $(\mathsf{Wn}^{\wp}, \mathsf{Wn}^{\overline{\wp}}) \leftarrow (\mathsf{Ps}_{\mathcal{D}} \setminus \mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}}, \mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}})$
else
5 $\mathcal{D}_{\mathrm{R}} \leftarrow \mathsf{f}_{\mathrm{R}}(\mathcal{D}, \mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}}, \overline{\wp})$
6 $(\mathsf{Wn}^0_{\mathrm{R}}, \mathsf{Wn}^1_{\mathrm{R}}) \leftarrow \mathsf{sol}(\mathcal{D}_{\mathrm{R}})$
7 $(\mathsf{Wn}^{\wp}, \mathsf{Wn}^{\overline{\wp}}) \leftarrow (\mathsf{Wn}^{\wp}_{\mathrm{R}}, \mathsf{Ps}_{\mathcal{D}} \setminus \mathsf{Wn}^{\wp}_{\mathrm{R}})$
8 **return** $(\mathsf{Wn}^0, \mathsf{Wn}^1)$

Algorithm 2: Left-subgame.

signature $\mathsf{f}_{\mathrm{L}} : \mathsf{PG} \to \mathsf{PG} \times \mathbb{B}$
function $\mathsf{f}_{\mathrm{L}}(\mathcal{D})$
1 $\wp \leftarrow \mathsf{pr}(\mathcal{D}) \bmod 2$
2 $\mathcal{D}^{\star} \leftarrow \mathcal{D} \setminus \mathsf{atr}^{\wp}_{\mathcal{D}}(\mathsf{pr}^{-1}_{\mathcal{D}}(\mathsf{pr}(\mathcal{D})))$
3 **return** $(\mathcal{D}^{\star}, \wp)$

Algorithm 3: Right-subgame.

signature $\mathsf{f}_{\mathrm{R}} : \mathsf{PG} \times_{\mathcal{D}} 2^{\mathsf{Ps}_{\mathcal{D}}} \times \mathbb{B} \to \mathsf{PG}$
function $\mathsf{f}_{\mathrm{R}}(\mathcal{D}, \mathsf{W}, \wp)$
1 $\mathcal{D}^{\star} \leftarrow \mathcal{D} \setminus \mathsf{atr}^{\wp}_{\mathcal{D}}(\mathsf{W})$
2 **return** $\mathcal{D}^{\star}$

We can now define the abstract notion of worst-case family that extends the core family, while still preserving the same essential properties that we are going to prove shortly.

Definition 3.3 *(Worst-case family).* A family of parity games $\{\mathcal{D}^k\}_{k=1}^{\omega}$ is a *worst-case family* if $\mathcal{D}^k$ is a core extension of $\mathcal{D}^k_{\mathrm{C}}$, for every index $k \in \mathbb{N}_+$.

4. The recursive algorithm and the core family

The Recursive procedure, reported in Algorithm 1 and proposed by Zielonka [34] in an equivalent version, solves a parity game $\mathcal{D}$ by decomposing it into two subgames, each of which is, then, solved recursively. Intuitively, the procedure works as follows. Algorithm 1, by means of Algorithm 2, starts by collecting all the positions that are forced to pass through a position with maximal priority $p \triangleq \mathsf{pr}(\mathcal{D})$ in that game. This first step results in computing the set $\mathsf{A} \triangleq \mathsf{atr}^{\wp}_{\mathcal{D}}(\mathsf{pr}^{-1}_{\mathcal{D}}(p))$, *i.e.*, the attractor to the set $\mathsf{pr}^{-1}_{\mathcal{D}}(p)$ of positions with priority p *w.r.t.* player $\wp \triangleq p \bmod 2$. The subgame $\mathcal{D}_{\mathrm{L}}$ is, then, obtained from $\mathcal{D}$ by removing A from it and solved recursively. The result is a partitioning of the positions of $\mathcal{D}_{\mathrm{L}}$ into two winning regions, $\mathsf{Wn}^0_{\mathrm{L}}$ and $\mathsf{Wn}^1_{\mathrm{L}}$, one per player.

In the following, for the sake of succinctness, given two arbitrary sets A and B and a function $\mathsf{f} : \mathsf{A} \to 2^{\mathsf{B}}$, we denote by $\mathsf{A} \to_a \mathsf{f}(a)$ the set of functions $\{\iota : \mathsf{A} \to \mathsf{B} : \forall a \in \mathsf{A}\,.\,\iota(a) \in \mathsf{f}(a)\}$, mapping each element in $a \in \mathsf{A}$ to an element $\iota(a) \in \mathsf{B}$ contained in the set $\mathsf{f}(a)$. Similarly, by $\mathsf{A} \times_a \mathsf{f}(a)$ we indicate the subset $\{(a, b) \in \mathsf{A} \times \mathsf{B} : b \in \mathsf{f}(a)\}$ of the Cartesian product $\mathsf{A} \times \mathsf{B}$, where the elements $a \in \mathsf{A}$ are only associated with the elements of B contained in $\mathsf{f}(a)$.

At this point, the algorithm checks whether the subgame $\mathcal{D}_{\mathrm{L}}$ is completely won by $\wp$ or, more generally, if the adversary $\overline{\wp}$ cannot force any other position in $\mathcal{D}$ into its own winning region $\mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}}$ in one move. In other words, none of the winning positions of the adversary $\overline{\wp}$ can attract something outside that region, *i.e.*, $\mathsf{pre}^{\overline{\wp}}_{\mathcal{D}}(\mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}}) \setminus \mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}} = \emptyset$. If this is the case, the entire game $\mathcal{D}$ is solved. Indeed, the positions of $\mathcal{D}$ winning for $\wp$ are all its positions except, possibly, for those won by $\overline{\wp}$ in subgame $\mathcal{D}_{\mathrm{L}}$ (see Line 4 of Algorithm 1). If, on the other hand, the above condition does not hold, the winning region of $\overline{\wp}$ can be extended with some other positions in $\mathcal{D}$. Let $\mathsf{B} \triangleq \mathsf{atr}^{\overline{\wp}}_{\mathcal{D}}(\mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}})$ be the set collecting all such positions. Observe that all the positions in B are certainly winning for $\overline{\wp}$ in the entire game, as, from each such position, $\overline{\wp}$ can force entering its own winning region $\mathsf{Wn}^{\overline{\wp}}_{\mathrm{L}}$, from which its opponent $\wp$ cannot escape. The residual subgame $\mathcal{D}_{\mathrm{R}}$, obtained by removing B from $\mathcal{D}$, as computed by Algorithm 3, may now contain positions winning for either player, and, therefore, needs to be solved again recursively (see Line 6 of Algorithm 1). All the positions of $\mathcal{D}_{\mathrm{R}}$ that turn out to be winning for $\wp$ in that game, namely $\mathsf{Wn}^{\wp}_{\mathrm{R}}$, are, then, all and only those positions winning for $\wp$ in the entire game $\mathcal{D}$, while the remaining ones are winning for $\overline{\wp}$ (see Line 7 of Algorithm 1).

As shown by Friedmann in [47], the algorithm admits a worst case family of games $\{\mathcal{F}^k\}_{k=1}^{\omega}$ that requires a number of recursive calls exponential in k. The reason is essentially the following. Each game $\mathcal{F}^k$ of that family contains all $\mathcal{F}^j$, with $1 \leq j < k$, as subgames. Each recursive call that receives as input one such subgame $\mathcal{F}^j$ requires to eventually solve both $\mathcal{F}^{j-1}$ and $\mathcal{F}^{j-2}$. As a consequence, the number of recursive calls performed by the algorithm on game $\mathcal{F}^k$ can be put in correspondence with a Fibonacci sequence. This proves that their number grows at least as fast as the sequence of the Fibonacci numbers, namely that their number is $\Omega\big(((1+\sqrt{5})/2)^k\big)$. The very reason that makes this family exponential also makes it amenable to a polynomial-time solution. It suffices to endow the Recursive algorithm with a memoization mechanism that, for each solved game $\mathcal{D}$, records the triple $(\mathcal{D}, \mathrm{Wn}^0_{\mathcal{D}}, \mathrm{Wn}^1_{\mathcal{D}})$. Each recursive call can, then, directly extract the winning regions of a subgame that is already contained in the collection, thus preventing the procedure from solving any subgame more than once. Not only does the resulting procedure make Friedmann worst case vain, but it also speeds up the solution of games significantly, as long as the number of repeated subgames remains relatively small, *e.g.*, linear in the size of the original game, which is often the case in practice.

We shall show that the same trick does not work for the worst-case family defined in the previous section. In order to prove that any game in that family requires an exponential number of different subgames to be solved, we shall characterize a suitable subtree of the recursion tree generated by the algorithm, when called on one of the games in the family. Starting from the root, which contains the original game $\mathcal{D}^k$, we fix specific observation points in the recursion tree that are identified by sequences in the set $w \in \{\mathtt{L}, \mathtt{R}\}^{\leq \lfloor k/2 \rfloor}$, where $\mathtt{L}$ (*resp.*, $\mathtt{R}$) denotes the recursive call on the left (*resp.*, right) subgame. Each sequence w identifies two subgames of $\mathcal{D}^k$, namely $\widehat{\mathcal{D}}^k_w$ and $\mathcal{D}^k_w$, that correspond to the input subgames of two successive nested calls. In the analysis of the recursion tree, we only take into account the left subgame $\mathcal{D}^k_w$ of each $\widehat{\mathcal{D}}^k_w$, thus disregarding its right subtree as it is inessential to the argument. An example of the resulting subgame tree for game $\mathcal{C}^2$ of the core family is depicted in Fig. 2. According to Algorithm 1 on input $\mathcal{C}^2 = \mathcal{D}^2_\varepsilon$, the first (left) recursive call is executed on the subgame obtained by removing the 1-attractor to the positions with maximal priority, in this case α_2, which only contains α_2. Therefore, the left subgame coincides precisely with $\widehat{\mathcal{D}}^2_{\mathtt{L}}$. The second call is executed on the game obtained by removing the 0-attractor in $\mathcal{D}^2_\varepsilon$ to the winning region for player 0 of the left subgame $\widehat{\mathcal{D}}^2_{\mathtt{L}}$. In this case, that winning region is precisely $\{\beta_2, \gamma_2\}$, and its 0-attractor is $\{\alpha_2, \gamma_1, \beta_2, \gamma_2\}$. As consequence, the subgame fed to the right-hand call precisely coincides with $\widehat{\mathcal{D}}^2_{\mathtt{R}}$. The rest of the subtree is generated applying the same reasoning. The following definition generalizes this notion to a game of any worst-case family and characterizes the portion of the recursion tree we are interested in analyzing.

Definition 4.1 *(Induced subgame tree).* Given a worst-case family $\{\mathcal{D}^k\}_{k=1}^{\omega}$, the *induced subgame tree* $\mathbb{G}^k \triangleq \{\mathcal{D}^k_w\}_{w \in \{\mathtt{L},\mathtt{R}\}^{\leq k}} \cup \{\widehat{\mathcal{D}}^k_w\}_{w \in \{\mathtt{L},\mathtt{R}\}^{\leq k+1}}^{w \neq \varepsilon}$ *w.r.t.* an index $k \in \mathbb{N}_+$ is defined inductively on the structure of the sequence $w \in \{\mathtt{L}, \mathtt{R}\}^{\leq \lfloor k/2 \rfloor}$ as follows, where $\mathcal{D}^k_\varepsilon \triangleq \mathcal{D}^k$ and $z \triangleq k - 2|w|$:

1. $\widehat{\mathcal{D}}^k_{w\mathtt{L}} \triangleq \mathcal{D}^k_w \setminus \mathrm{atr}^{\overline{\wp}_k}_{\mathcal{D}^k_w}(\{\alpha_z\})$;
2. $\widehat{\mathcal{D}}^k_{w\mathtt{R}} \triangleq \mathcal{D}^k_w \setminus \mathrm{atr}^{\wp_k}_{\mathcal{D}^k_w}(\mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}})$;
3. $\mathcal{D}^k_w \triangleq \widehat{\mathcal{D}}^k_w \setminus \mathrm{atr}^{\wp_k}_{\widehat{\mathcal{D}}^k_w}(\{\alpha_{z+1}\})$, if $w \neq \epsilon$.

Before proceeding with the proof of the main result of this section, we need some additional properties of the induced subgame tree of any worst-case family. The following lemma, which is essential to the result, states some invariants of the elements contained in the tree induced by a game $\mathcal{D}^k$ that extends the core $\mathcal{C}^k$. In particular, these invariants ensure that all those elements are subgames of $\mathcal{D}^k$ (Items 1 and 4) and that, depending on the identifying sequence w, they contain the required leading positions α_i of the core (Items 2 and 7). In addition, it states two important properties of every left child in the tree, *i.e.*, those elements identified by a sequence w ending with $\mathtt{L}$. Both of them will be instrumental in proving that all the subgames in the tree are indeed different and to assess their number, as we shall see in Lemmas 4.3 and 4.4. The first property ensures that each such game necessarily contains a specific position γ_i, with index i depending on w (Items 3 and 5). The second one (Item 6) characterizes the winning region for player $\wp_k$ of the left-child subgames $\widehat{\mathcal{D}}^k_{w\mathtt{L}}$. It states that, in each such game, the winning positions for player $\wp_k$ contained in the corresponding core $\mathcal{C}^k$ are all its β-positions and γ-positions, with index that is of the same parity as $\wp_k$ and greater than the maximal index x of the leading α-position. Indeed, as soon as the higher positions α_i, with $i \in [x+1, k]$, are removed from the game, each residual corresponding $\gamma_{\wp_k+2j}$, possibly together with its associated $\beta_{\wp_k+2j}$, is necessarily contained in an independent $\wp_k$-dominion.

In the sequel, by $\mathsf{lst}(w)$ we denoted the last position of a non-empty sequence $w \in \{\mathtt{L}, \mathtt{R}\}^+$.

Lemma 4.1. *For any index $k \in \mathbb{N}_+$ and sequence $w \in \{\mathtt{L}, \mathtt{R}\}^{\leq \lfloor k/2 \rfloor + 1}$, let $z \triangleq k - 2|w|$. Then, the following properties hold true:*

- *if $|w| \leq \lfloor k/2 \rfloor$, then*
  1. *$\mathcal{D}^k_w$ is a subgame of $\mathcal{D}^k$;*
  2. *$\alpha_j \in \mathcal{D}^k_w$ iff $j \in [0, z]$;*
  3. *$\gamma_j \in \mathcal{D}^k_w$, for all $j \in [0, z]$, and $\gamma_{z+1} \in \mathcal{D}^k_w$, if $w \neq \varepsilon$ and $\mathsf{lst}(w) = \mathtt{L}$;*

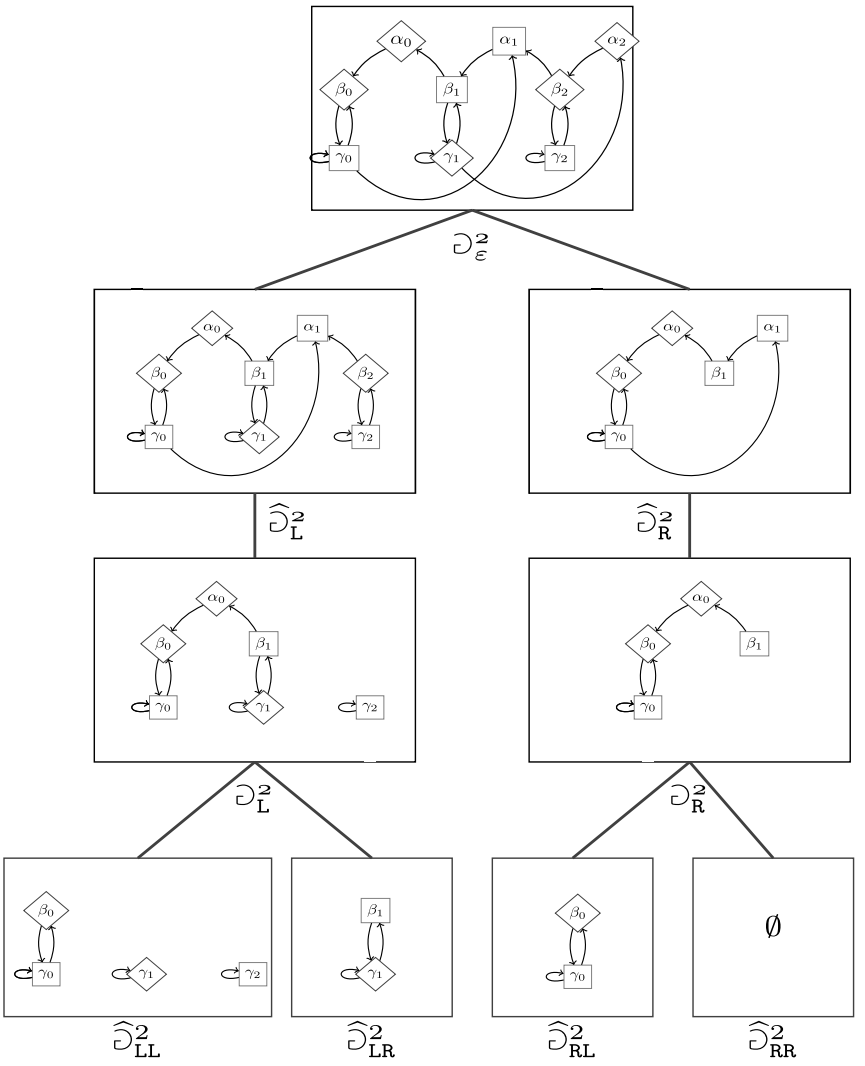

Fig. 2. The induced subgame tree $\mathbb{G}^2$ of ∂^2.

- *if* $|w| > 0$, *then*
  4. $\widehat{\partial}^k_w$ *is a subgame of* ∂^k;
  5. $\gamma_j \in \widehat{\partial}^k_w$, *for all* $j \in [0, z]$, *and, whenever* $\mathsf{lst}(w) = \mathtt{L}$, $\gamma_{z+1} \in \widehat{\partial}^k_w$, *if* $|w| \leq \lfloor k/2 \rfloor$, *and* $\gamma_0 \in \widehat{\partial}^k_w$, *otherwise;*
  6. $\mathrm{Wn}^{\wp_k}_{\widehat{\partial}^k_w} \cap \mathcal{C}^k = \{\beta_{z+2j}, \gamma_{z+2j} \in \widehat{\partial}^k_w : j \in [1, |w|]\}$, *if* $\mathsf{lst}(w) = \mathtt{L}$;
- *if* $0 < |w| \leq \lfloor k/2 \rfloor$, *then*
  7. $\alpha_j \in \widehat{\partial}^k_w$ *iff* $j \in [0, z+1]$.

Proof. The proof proceeds by induction on the structure of the sequence w.

For the base case $w = \varepsilon$, we have that $\partial^k_w = \partial^k$ and $z = k$. Thus, Items 1, 2, and 3 hold due to Item 1 of Definition 3.2 and the structure of the game core $\mathcal{C}^k$ given in Definition 3.1, while Items 4, 5, 6, and 7 are vacuously verified.

For the inductive case, assume that all properties hold for a given sequence $w \in \{\mathtt{L}, \mathtt{R}\}^{\leq \lfloor k/2 \rfloor}$. We show that they hold for $v = wx \in \{w\mathtt{L}, w\mathtt{R}\}$ as well. By Items 1 and 2 of Definition 4.1, we have that $\widehat{\partial}^k_v = \partial^k_w \setminus \mathrm{A}$, where $\mathrm{A} = \mathrm{atr}^{\overline{\wp_k}}_{\partial^k_w}(\{\alpha_z\})$, if $x = \mathtt{L}$, and $\mathrm{A} = \mathrm{atr}^{\wp_k}_{\partial^k_w}(\mathrm{Wn}^{\wp_k}_{\widehat{\partial}^k_{w\mathtt{L}}})$, otherwise. We now proceed by analyzing all seven properties separately.

[Item 4]. Since $\widehat{\partial}^k_v$ is a subgame of ∂^k_w, the property applied to v immediately follows from Item 1 applied to w.

[Item 7]. Item 1, together with Items 1 and 3 of Definition 3.2 and the topology of the game $\mathcal{C}^k$, implies that $\mathrm{A} = \{\alpha_z, \beta_{z+1}\}$, if $x = \mathtt{L}$, and $\mathrm{A} \cap \mathcal{C}^k = \{\alpha_z\} \cup \{\gamma_{z-1} : |w| < \lfloor k/2 \rfloor\} \cup \{\beta_{z+1} : \gamma_{z+1} \notin \partial^k_w\} \cup \{\beta_{z+2j}, \gamma_{z+2j} \in \partial^k_w : j \in [0, |w|]\}$, otherwise. Indeed, if $x = \mathtt{L}$, the position β_{z+1} is the only one that gets attracted by α_z *w.r.t.* player $\overline{\wp_k}$, as the other two positions γ_{z-1} and γ_{z+1}, having a non-forced move to either α_z or β_{z+1}, belong to player $\wp_k$. If, on the other hand, $x = \mathtt{R}$, due to Item 6

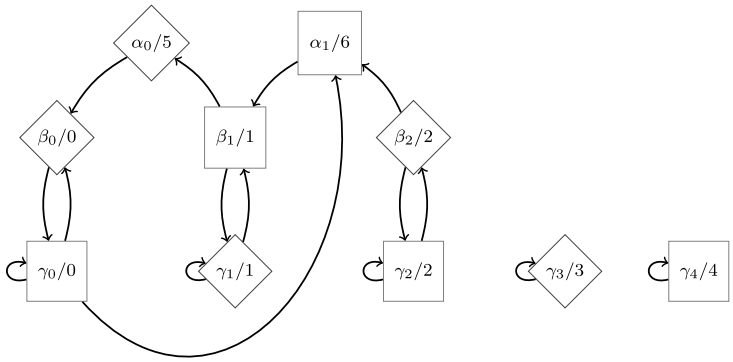

Fig. 3. Game $\widehat{\mathcal{D}}^4_{\mathtt{LL}}$.

for $w\mathtt{L}$, we have that $\mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}} \cap \mathcal{C}^k = \{\beta_{z+2j}, \gamma_{z+2j} \in \widehat{\mathcal{D}}^k_{w\mathtt{L}} : j \in [0, |w|]\}$. Therefore, the only positions of $\mathcal{D}^k_w \cap \mathcal{C}^k$ attracted by $\mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}}$ *w.r.t.* player $\wp_k$ are α_z and, possibly, γ_{z-1} or β_{z+1}. Indeed, α_z has a forced move to β_z. Moreover, if $|w| < k$, then γ_{z-1} is willing to follow α_z, since it belongs to player $\wp_k$. Finally, if γ_{z+1} does not belong to the game, β_{z+1} is forced to reach α_z as well. At this point, the item for v is immediately derived from Item 2 for w, by observing that in both cases α_z is the only α-position that is removed and $k - 2|v| + 1 = k - 2|w| - 1 < z$.

[Item 5]. Similarly to the proof of Item 4], the property of interest applied to v is derived from Item 3 applied to w, by observing that the only γ-positions possibly removed have index at least equal to $z - 1 > k - 2|v| = z - 2$.

We can now focus on the games $\mathcal{D}^k_v$ that, by Definition 4.1, are equal to $\widehat{\mathcal{D}}^k_v \setminus \mathrm{A}$ with $\mathrm{A} = \mathrm{atr}^{\wp_k}_{\widehat{\mathcal{D}}^k_v}(\{\alpha_{z-1}\})$.

[Item 1]. The property applied to v simply derives from Item 4 applied to v.

[Item 2]. Again by Items 1 and 3 of Definition 3.2 and the topology of the game $\mathcal{C}^k$, we have that $\mathrm{A} = \{\alpha_{k-2|v|+1}, \beta_{k-2|v|}\} = \{\alpha_{z-1}, \beta_z\}$, if $x = \mathtt{L}$, and $\mathrm{A} = \{\alpha_{k-2|v|+1}\} = \{\alpha_{z-1}\}$, otherwise. Consequently, the item for v follows from Item 7 for v, since α_{z-1} is the unique α-position that is removed from the game.

[Item 3]. The property applied to v is implied by Item 5 applied to v, as no γ-positions is removed.

[Item 6]. To conclude the proof of the entire theorem, consider the case where $v = w\mathtt{L}$. First observe that, due to Items 1 and 4 of Definition 3.2, the set of positions $\mathrm{D} = \{\beta_{z+2j}, \gamma_{z+2j} \in \widehat{\mathcal{D}}^k_v : j \in [0, |w|]\} \cup \bigcup_{\gamma_{z+2j} \in \widehat{\mathcal{D}}^k_v}^{j \in [0,|w|]} Mv(\gamma_{z+2j}) \cap \widehat{\mathcal{D}}^k_v$ is a $\wp_k$-dominion in the game $\widehat{\mathcal{D}}^k_v$, constituted by the possibly overlapping union of the $\wp_k$-dominions $\mathrm{D}^{\wp_k}_j = \{\beta_{z+2j}, \gamma_{z+2j} \in \widehat{\mathcal{D}}^k_v\} \cup Mv(\gamma_{z+2j}) \cap \widehat{\mathcal{D}}^k_v$, for every index $j \in [0, |w|]$ such that $\gamma_{z+2j} \in \widehat{\mathcal{D}}^k_v$. Indeed, the priority of the positions in $\{\beta_{z+2j}, \gamma_{z+2j} \in \widehat{\mathcal{D}}^k_v\}$ is of the same parity as player $\wp_k$, being equal to $z + 2j$, and the priorities of the positions in $Mv(\gamma_{z+2j}) \cap \widehat{\mathcal{D}}^k_v$ are not greater than $z + 2j$. Moreover, there is no move the opponent $\overline{\wp_k}$ can take from its unique position γ_{z+2j} in order to escape from the set, as there is no position $\alpha_{z+2j+1} \in \widehat{\mathcal{D}}^k_v$. Hence, $\mathrm{D}^{\wp_k}_j$ is, by definition, a $\wp_k$-dominion.

For example, looking at the game $\mathcal{D}^4_{\mathtt{LL}}$ depicted in Fig. 3, it is immediate to see that β_2, γ_2, and γ_4 belong to the winning region of player 0. Observe that $\mathrm{D} \cap \mathcal{C}^k = \{\beta_{k-2|v|+2j}, \gamma_{k-2|v|+2j} \in \widehat{\mathcal{D}}^k_v : j \in [1, |v|]\}$, being $k - 2|v| = z - 2$. Therefore, to prove the thesis, it suffices to show that all the other positions in $\mathcal{C}^k \setminus \mathrm{D}$ belong to the $\overline{\wp_k}$-dominion in the game $\widehat{\mathcal{D}}^k_v$ obtained by the possibly overlapping union of the following $\overline{\wp_k}$-dominions:

- $\mathrm{D}^{\overline{\wp_k}}_j = \{\beta_{z+2j+1}, \gamma_{z+2j+1} \in \widehat{\mathcal{D}}^k_v\} \cup Mv(\gamma_{z+2j+1}) \cap \widehat{\mathcal{D}}^k_v$, for every index $j \in [0, |w|[$ such that $\gamma_{z+2j+1} \in \widehat{\mathcal{D}}^k_v$;
- $\mathrm{H} = \{\alpha_j, \beta_j, \gamma_j : j \in [0, z[\} \cup \bigcup_{j \in [0,z[}^{j \not\equiv_2 z} Mv(\gamma_j) \cap \widehat{\mathcal{D}}^k_v$, where $\bigcup_{j \in [0,z[}^{j \not\equiv_2 z} Mv(\gamma_j) \cap \widehat{\mathcal{D}}^k_v$ takes care of including those position of the core extension that are not in the core but are attracted by the γ-positions in $\{\alpha_j, \beta_j, \gamma_j : j \in [0, z[\}$.

For instance, in the game of Fig. 3, where $v = \mathtt{LL}$ and $\overline{\wp_k} = \overline{\wp_4} = 1$, such a 1-dominion is the union of $\mathrm{D}^1_1 = \{\gamma_3\}$ and $\mathrm{H} = \{\alpha_j, \beta_j, \gamma_j : j \in \{0, 1\}\}$. The proof that $\mathrm{D}^{\overline{\wp_k}}_j$ is a $\overline{\wp_k}$-dominion is, *mutatis mutandis*, the same as the one used to show that $\mathrm{D}^{\wp_k}_j$ is a $\wp_k$-dominion. Therefore, we only focus on the component H. Observe that, since $\alpha_z \notin \widehat{\mathcal{D}}^k_v$, the two positions $\beta_{z-1}, \gamma_{z-1} \in \mathrm{H}$, together with those in $Mv(\gamma_{z-1}) \cap \widehat{\mathcal{D}}^k_v$, form a $\overline{\wp_k}$-dominion X. Again, the proof here is the same as that used for the $\overline{\wp_k}$-dominions $\mathrm{D}^{\overline{\wp_k}}_j$. Consequently, the $\overline{\wp_k}$-attractor $\mathrm{A} = \mathrm{atr}^{\overline{\wp_k}}_{\widehat{\mathcal{D}}^k_v}(\mathrm{X})$ is still a $\overline{\wp_k}$-dominion. Now, by structural induction, we can also see that the remaining positions in $\mathrm{H} \setminus \mathrm{A}$ correspond to another $\overline{\wp_k}$-dominion. Therefore, in $\widehat{\mathcal{D}}^k_v$ player $\wp_k$ has only the possibility to either remain in $\mathrm{H} \setminus \mathrm{A}$ or to pass in A and then remain there forever. Hence, H is a $\overline{\wp_k}$-dominion as required. □

Finally, the next lemma simply establishes that all the subgames contained in the induced subgame tree of Definition 4.1 are indeed generated by the Recursive algorithm when called with input game $\mathcal{D}^k$ of some worst-case family.

Lemma 4.2. *For any index $k \in \mathbb{N}_+$ and sequence $w \in \{\mathtt{L},\mathtt{R}\}^{\leq \lfloor k/2 \rfloor}$, the following properties hold:*

1. $\mathrm{f}_{\mathtt{L}}(\mathcal{D}^k_w) = (\widehat{\mathcal{D}}^k_{w\mathtt{L}}, \overline{\wp_k})$;
2. $\mathrm{f}_{\mathtt{R}}(\mathcal{D}^k_w, \mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}}, \wp_k) = \widehat{\mathcal{D}}^k_{w\mathtt{R}}$;
3. $\mathrm{f}_{\mathtt{L}}(\widehat{\mathcal{D}}^k_w) = (\mathcal{D}^k_w, \wp_k)$, *if* $w \neq \varepsilon$.

Proof. The proof of each item easily follows from the definition of the algorithm and the properties of the game they are applied to.

[Item 1]. Due to Items 1 and 2 of Lemma 4.1 and Items 1 and 2 of Definition 3.2, the maximal priority $\mathrm{pr}(\mathcal{D}^k_w)$ of the game $\mathcal{D}^k_w$ is $2(k-|w|)+1+\wp_k \equiv_2 \overline{\wp_k}$, which is assumed by the position $\alpha_{k-2|w|}$. Consequently, we have $\mathrm{pr}^{-1}(\mathrm{pr}(\mathcal{D}^k_w)) = \{\alpha_{k-2|w|}\}$. Thus, the thesis follows due to Item 1 of Definition 4.1 and the instructions constituting Algorithm 2.

[Item 2]. The thesis is an immediate consequence of the structure of Algorithm 3 and Item 2 of Definition 4.1.

[Item 3]. Due to Items 4 and 7 of Lemma 4.1 and Items 1 and 2 of Definition 3.2, the maximal priority $\mathrm{pr}(\widehat{\mathcal{D}}^k_w)$ of the game $\widehat{\mathcal{D}}^k_w$ is $2(k-|w|)+2+\wp_k \equiv_2 \wp_k$, which is assumed by the position $\alpha_{k-2|w|+1}$. Consequently, we have $\mathrm{pr}^{-1}(\mathrm{pr}(\widehat{\mathcal{D}}^k_w)) = \{\alpha_{k-2|w|+1}\}$. Thus, the thesis follows due to Item 3 of Definition 4.1 and the instructions constituting Algorithm 2. □

We are now ready for the main result of this section, namely that the induced subgame tree contains elements which are all different from each other and whose number is exponential in the index k. We split the result into two lemmas. The first one simply states that any subgame in the left subtree of some $\mathcal{D}^k_w$ is different from any subgame in the right subtree. The idea is that for any subgame in the tree, all subgames of its left subtree contain at least one position, a specific position γ_i, with i depending on w, that is not contained in any subgame of its right subtree.

Lemma 4.3. *For all indexes $k \in \mathbb{N}_+$ and sequences $w, v \in \{\mathtt{L},\mathtt{R}\}^*$, with $\ell \triangleq |w| + |v| \leq \lfloor k/2 \rfloor$ and $z \triangleq k - 2|w| - 1$, the following properties hold:*

1. $\gamma_z \in \mathcal{D}^k_{w\mathtt{L}v}$ *and* $\gamma_z \in \widehat{\mathcal{D}}^k_{w\mathtt{L}v}$, *if* $\ell < \lfloor k/2 \rfloor$, *and* $\gamma_0 \in \widehat{\mathcal{D}}^k_{w\mathtt{L}v}$, *otherwise;*
2. $\gamma_z \notin \mathcal{D}^k_{w\mathtt{R}v}$ *and* $\gamma_z \notin \widehat{\mathcal{D}}^k_{w\mathtt{R}v}$, *if* $\ell < \lfloor k/2 \rfloor$, *and* $\gamma_0 \notin \widehat{\mathcal{D}}^k_{w\mathtt{R}v}$, *otherwise.*

Proof. First observe that, if $\ell < \lfloor k/2 \rfloor$, by Items 3 and 5 of Lemma 4.1, position γ_z belongs to both $\mathcal{D}^k_w$ and $\widehat{\mathcal{D}}^k_w$, since $0 < z < k - 2|w|$.

[Item 1]. We first show that the every position γ_z belongs to all the descendants of $\mathcal{D}^k_w$ in its left subtree. The proof proceeds by induction on the length of the sequence v and recall that, due to Item 2 (*resp.*, 7) of Lemma 4.1, the position with maximal priority in $\mathcal{D}^k_w$ (*resp.*, in $\widehat{\mathcal{D}}^k_w$) is α_z (*resp.*, α_{z+1}). Assume, for the base case, that $|v| = 0$. The thesis becomes $\gamma_z \in \mathcal{D}^k_{w\mathtt{L}}$ and $\gamma_z \in \widehat{\mathcal{D}}^k_{w\mathtt{L}}$. By Item 1 of Definition 4.1, $\widehat{\mathcal{D}}^k_{w\mathtt{L}}$ is defined as $\mathcal{D}^k_w \setminus \mathrm{A}$, where $\mathrm{A} = \mathrm{atr}^{\overline{\wp_k}}_{\mathcal{D}^k_w}(\{\alpha_{z+1}\})$. By Definition 3.2 of core extension (Items 1, 3, and 4), a move entering α_{z+1} may only come from β_{z+2}, if it is present in the subgame, which is always the case unless $w = \varepsilon$, or from γ_z, whose owner is player $\wp_k$ and cannot be attracted. Hence, $\mathrm{A} = \{\alpha_{z+1}, \beta_{z+2}\}$, if $w \neq \varepsilon$, and $\mathrm{A} = \{\alpha_{z+1}\}$, otherwise. Similarly, by Item 3, $\mathcal{D}^k_{w\mathtt{L}}$ is defined as $\widehat{\mathcal{D}}^k_{w\mathtt{L}} \setminus \mathrm{A}$, where $\mathrm{A} = \mathrm{atr}^{\wp_k}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}}(\{\alpha_z\})$. From the same observations as in the previous case, we have that $\mathrm{A} = \{\alpha_z, \beta_{z+1}\}$. Hence, no position γ_i is removed from either game and the thesis immediately follows.

For the inductive case, assume $|v| > 0$, let v' such that $v = v' \cdot x$, with $x \in \{\mathtt{L},\mathtt{R}\}$, and $v \triangleq w\mathtt{L}v'$. By the inductive hypothesis, $\gamma_z \in \widehat{\mathcal{D}}^k_v$ and $\gamma_z \in \mathcal{D}^k_v$. We have two cases, depending on whether $x = \mathtt{L}$ or $x = \mathtt{R}$. Let $r \triangleq k - 2|v|$. If $x = \mathtt{L}$, according to Item 1 of Definition 4.1, $\widehat{\mathcal{D}}^k_{v\mathtt{L}}$ is obtained from $\mathcal{D}^k_v$ by removing $\mathrm{A} = \mathrm{atr}^{\overline{\wp_k}}_{\mathcal{D}^k_v}(\{\alpha_r\}) = \{\alpha_r, \beta_{r+1}\}$. Similarly, by Item 3 of Definition 4.1, $\mathcal{D}^k_{v\mathtt{L}}$ is defined as $\widehat{\mathcal{D}}^k_{v\mathtt{L}} \setminus \mathrm{A}$, where $\mathrm{A} = \mathrm{atr}^{\wp_k}_{\widehat{\mathcal{D}}^k_{v\mathtt{L}}}(\{\alpha_{r-1}\})$, by observing that $|v\mathtt{L}| = |v| + 1$ and, therefore, $k - 2|v\mathtt{L}| + 1 = r - 1$. In both cases the thesis follows immediately.

Let us now consider the case with $x = \mathtt{R}$. According to Item 2 of Definition 4.1, $\widehat{\mathcal{D}}^k_{v\mathtt{R}}$ is obtained by removing the set $\mathrm{A} = \mathrm{atr}^{\wp_k}_{\mathcal{D}^k_v}(\mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{v\mathtt{L}}})$ from $\mathcal{D}^k_v$. Position γ_z has index z congruent to $\overline{\wp_k}$ modulo 2 and cannot belong to $\mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{v\mathtt{L}}}$, which, by Item 6 of Lemma 4.1, only contains, among the positions from the core, those β_i and γ_i, with $i \geq k - 2|w| > z$ and congruent to $\wp_k$ modulo 2. Since, $\widehat{\mathcal{D}}^k_{v\mathtt{L}}$ is a subgame of a core extension, it holds that γ_z is owned by player $\wp_k$ and can only have a move leading to α_{z+1}, which is not in the subgame, or to a position outside the core and owned by player $\overline{\wp_k}$. As a consequence, it cannot end up in $\mathrm{atr}^{\wp_k}_{\mathcal{D}^k_v}(\mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{v\mathtt{L}}})$ and the thesis holds.

Finally, recall that $\mathcal{D}^k_{v\mathtt{R}} = \widehat{\mathcal{D}}^k_{v\mathtt{R}} \setminus \mathrm{A}$, where $\mathrm{A} = \mathrm{atr}^{\wp_k}_{\widehat{\mathcal{D}}^k_{v\mathtt{R}}}(\{\alpha_{\hat{r}}\})$, with $\hat{r} \triangleq k - 2|v\mathtt{R}| + 1$. In the considered subgame, $\alpha_{\hat{r}}$ has incoming moves only from $\beta_{\hat{r}+1}$ and $\gamma_{\hat{r}-1}$. However, $\hat{r} - 1 = k - 2|v\mathtt{R}| < k - 2|w| - 1 = z$. Moreover, index $\hat{r}+1$ is congruent to $\wp_k$ modulo 2, and thus $\beta_{\hat{r}+1}$ is not contained in $\widehat{\mathcal{D}}^k_{v\mathtt{R}}$, being in $\mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{v\mathtt{L}}}$ as shown above. As a consequence, $\mathrm{A} = \{\alpha_{\hat{r}}\}$ and the thesis follows. In addition, when $|w| = \lfloor k/2 \rfloor$, Item 3 of Lemma 4.1 tells us that $\gamma_0 \in \mathcal{D}^k_w$. If $\ell = k$, instead, we have that γ_0 belongs to $\widehat{\mathcal{D}}^k_{w\mathtt{L}}$, due to Item 5 of Lemma 4.1. This ends the proof of the item.

[Item 2]. Observe that, if $|w| < \lfloor k/2 \rfloor$, then $\gamma_z \notin \widehat{\mathcal{D}}^k_{w\mathtt{R}}$. Indeed, position $\gamma_z \in \mathrm{atr}^{\wp_k}_{\mathcal{D}^k_w}(\mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}})$, as shown above. Since this set is removed from $\mathcal{D}^k_w$ to obtain $\widehat{\mathcal{D}}^k_{w\mathtt{R}}$, the thesis holds for $\widehat{\mathcal{D}}^k_{w\mathtt{R}}$. Moreover, every descendant of $\widehat{\mathcal{D}}^k_{w\mathtt{R}}$ in the subgame tree is obtained only by removing positions. As a consequence, none of them can contain position γ_z. In case $|w| = \lfloor k/2 \rfloor$, instead, it suffices to observe that, according to Item 6 of Lemma 4.1, $\gamma_0 \in \mathrm{Wn}^{\wp_k}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}}$, hence it cannot be contained in $\widehat{\mathcal{D}}^k_{w\mathtt{R}}$. □

The result asserting the exponential size of the induced subtrees of any worst-case family is given by the next lemma. This follows by observing that the number of nodes in the induced tree is exponential in k and by showing that the subgames associated with any two nodes in the tree $\mathbb{G}^k$ are indeed different.

Lemma 4.4. *$|\mathbb{G}^k| = 3(2^{\lfloor k/2 \rfloor + 1} - 1)$, for any $k \in \mathbb{N}_+$.*

Proof. To prove that the size of $\mathbb{G}^k$ is as stated, we first need to show that all the elements contained in the subgame trees are different, namely that, for each $w \neq w'$, the subgames $\mathcal{D}^k_w$, $\mathcal{D}^k_{w'}$, $\widehat{\mathcal{D}}^k_w$, and $\widehat{\mathcal{D}}^k_{w'}$ are pairwise different. Let us start by showing that $\mathcal{D}^k_w \neq \widehat{\mathcal{D}}^k_w$, for each $w \neq \varepsilon$. By Item 7 of Lemma 4.1, position $\alpha_{k-2|w|+1} \in \widehat{\mathcal{D}}^k_w$ and, by Item 3 of Definition 4.1, this position is removed from $\widehat{\mathcal{D}}^k_w$ to obtain $\mathcal{D}^k_w$. Hence, those two subgames cannot be equal. Let us consider now two subgames, each associated with one of the sequences w and w'. There are two possible cases: either *(i)* w is a strict prefix of w', *i.e.*, one subgame is a descendant of the other in the subgame tree, or *(ii)* w and w' share a common longest prefix v that is different from both, *i.e.*, the two subgames lie in two distinct subtrees of the subgame associated with v. In case *(i)* we have that $w' = wv$, for some $v \neq \varepsilon$. An easy induction on the length of v can prove that if $\mathcal{D} \in \{\mathcal{D}^k_w, \widehat{\mathcal{D}}^k_w\}$ and $\mathcal{D}' \in \{\mathcal{D}^k_{w'}, \widehat{\mathcal{D}}^k_{w'}\}$ are the subgames associated with w and w', respectively, then the second is a strict subgame of the first, *i.e.*, $\mathcal{D}' \subset \mathcal{D}$. Indeed, Definition 4.1 together with Items 2, 7, and 6 of Lemma 4.1 ensure that, at each step downward along a path in the tree starting from $\mathcal{D}$, whether we proceed on the left or the right branch, at least one position is always removed from the current subgame. In case *(ii)*, instead, Item 1 of Lemma 4.3 tells us that there is at least one position, γ_z in the lemma, contained in all the subgames of the left subtree, while Item 2 states that the same position is not contained in any subgame of the right subtree. Therefore, each of the two subgames associated with w must be different from either of the two subgames associated with w'.

Finally, to prove the statement of the lemma, it suffices to observe that the number of sequences of length at most $\lfloor k/2 \rfloor$ over the alphabet $\{\mathtt{L}, \mathtt{R}\}$ are precisely $2^{\lfloor k/2 \rfloor + 1} - 1$ and with each such sequence w a subgame $\mathcal{D}^k_w$ is associated. As a consequence, the set $\{\mathcal{D}^k_w\}_{w \in \{\mathtt{L},\mathtt{R}\}^{\leq k}}$ contains $2^{\lfloor k/2 \rfloor + 1} - 1$ different elements. Moreover, each such subgame has two children in $\{\widehat{\mathcal{D}}^k_w\}^{w \neq \varepsilon}_{w \in \{\mathtt{L},\mathtt{R}\}^{\leq k+1}}$. We can, then, conclude that the size of $\mathbb{G}^k$ is precisely $3(2^{\lfloor k/2 \rfloor + 1} - 1)$. □

As a consequence of Lemma 4.4, we can obtain a stronger lower bound for the execution time of the Recursive algorithm. Indeed, the result holds regardless of whether the algorithm is coupled with a memoization technique.

Theorem 4.1 *(Exponential worst case). The number of distinct recursive calls executed by the Recursive algorithm, with or without memoization, on a game with n positions is $\Omega(2^{\frac{n}{6}})$ in the worst case.*

Proof. To prove the theorem, it suffices to consider a game $\mathcal{C}^k$ belonging to the core family. Indeed, Lemma 4.2 states that the induced subgame tree of $\mathcal{C}^k$ is a subset of the recursion tree induced by the Recursive algorithm executed on that game. Therefore, according to Lemma 4.4, the algorithm performs at least $3(2^{\lfloor k/2 \rfloor + 1} - 1)$ calls, each on a different subgame. By Definition 3.1, game $\mathcal{C}^k$ has $n = 3(k+1)$ positions and, therefore, we have $\lfloor k/2 \rfloor = \lfloor \frac{n-3}{6} \rfloor$. As a consequence, the number of recursive calls is bounded from below by $3(2^{\lfloor \frac{n+3}{6} \rfloor} - 1) = \Omega(2^{\frac{n}{6}})$. □

5. Progress measure-based algorithms and the core family

Progress measures are decorations of a graph, whose local consistency usually guarantees, from an high-level point of view, some global property of the graph itself [30]. Fixed a player $\wp \in \{0, 1\}$, the original progress measure algorithm [29] and its quasi-polynomial time variation [43] exploit this idea, by decorating every position $v \in \mathrm{Ps}$ of a game $\mathcal{D}$ with an element $\iota(v) \in \mathrm{M}^\wp$, called *measure*, from an ordered set $(\mathrm{M}^\wp, \sqsubseteq)$. These decorations ι, called *$\wp$-measure functions*, naturally inherit the order $\sqsubseteq$ from $\mathrm{M}^\wp$. Measures are used to compare, via a suitable truncated ordering $\sqsubseteq_{\mathrm{pr}(v)}$, the relevance of the priority $\mathrm{pr}(v)$ of position $v \in \mathrm{Ps}$ *w.r.t.* those associated with its successors $v' \in Mv(v)$. These comparisons characterize, in this context, the local consistency property among positions mentioned above. It can be proved that a minimal measure function ι_ω satisfying such a property always exists and uniquely identifies those positions in Ps belonging to the winning region $\mathrm{Wn}^\wp$ of player $\wp$ [29]. The two algorithms share the same structure as the ordered measure set $(\mathrm{M}^\wp, \sqsubseteq)$. In more detail, given $\mathrm{D} \triangleq \{d \in \mathrm{Pr} : d \equiv_2 \wp\}$ the set of priorities congruent to the parity of player $\wp$ and $(\mathrm{N}, \preceq)$ an ordered set, the support is $\mathrm{M}^\wp \subseteq \{\top\} \cup \mathrm{D} \to \mathrm{N}$, where, $\sqsubseteq$ is the lexicographic ordering *w.r.t.* $\preceq$ on $\mathrm{M}^\wp \setminus \{\top\}$ that also satisfies $m \sqsubseteq \top$, for every $m \in \mathrm{M}^\wp$, and $\top \in \mathrm{M}^\wp$. The priority-sensitive comparison $\sqsubseteq_p$, with $p \in \mathrm{Pr}$, is derived from $\sqsubseteq$ and a truncation operation defined as follows: $m\!\downarrow_p \triangleq m \restriction \{d \in \mathrm{D} : d \geq p\}$, if $m \neq \top$, and $m\!\downarrow_p \triangleq \top$, otherwise, for all $m \in \mathrm{M}^\wp$. The truncated ordering $\sqsubseteq_p$ can, then,

be obtained as $m_1 \sqsubseteq_p m_2$ if $m_1{\downarrow_p} \sqsubseteq m_2{\downarrow_p}$, for all $m_1, m_2 \in \mathrm{M}^{\wp}$. Given a $\wp$-measure function $\iota\colon \mathrm{Ps} \to \mathrm{M}^{\wp}$, *i.e.*, a function that assigns a measure in $\mathrm{M}^{\wp}$ to each position of the game, the notion of *successor measure* of ι *w.r.t.* a move $(v_1, v_2) \in Mv$ and an ordering relation $\lhd \in \{\sqsubset, \sqsubseteq\}$ can be defined as the element $\mathsf{suc}_{\lhd}(\iota)((v_1, v_2)) \triangleq \min_{\sqsubseteq_{\mathsf{pr}(v_1)}}\{m \in \mathrm{M}^{\wp} : \iota(v_2) \lhd m\}$. Notice that, if $\iota(v_2) = \top$, then $\mathsf{suc}_{\lhd}(\iota)((v_1, v_2)) = \top$, since $\{m \in \mathrm{M}^{\wp} : \iota(v_2) \lhd m\} = \emptyset$ in this case. Thanks to the Knaster-Tarski theorem, the minimal measure function ι_ω satisfying the local consistency property can be obtained as the least fixed point $\iota_\omega = \mu X.\mathsf{lift}^{\wp}(X)$ of the monotone *lifting operator* $\mathsf{lift}^{\wp}\colon (\mathrm{Ps} \to \mathrm{M}^{\wp}) \to (\mathrm{Ps} \to \mathrm{M}^{\wp})$ defined via the two auxiliary functions $\mathsf{lift}_{\lhd}\colon (\mathrm{Ps} \to \mathrm{M}^{\wp}) \to (Mv \to \mathrm{M}^{\wp})$ and $\mathsf{lift}^{\wp}\colon (\mathrm{Ps} \to \mathrm{M}^{\wp}) \to (Mv \to \mathrm{M}^{\wp})$ according to the following:

Definition 5.1.

1. $\mathsf{lift}_{\lhd}(\iota)((v_1, v_2)) \triangleq \max_{\sqsubseteq_{\mathsf{pr}(v[1])}}]\{\iota(v_1), \mathsf{suc}_{\lhd}(\iota)((v_1, v_2))\}$;
2. $\mathsf{lift}^{\wp}(\iota)((v_1, v_2)) \triangleq \begin{cases} \mathsf{lift}_{\sqsubset_{\mathsf{pr}(v[1])}}](\iota)((v_1, v_2)), & \text{if } \mathsf{pr}(v_1) \equiv_2 \wp; \\ \mathsf{lift}_{\sqsubseteq_{\mathsf{pr}(v[1])}}](\iota)((v_1, v_2)), & \text{otherwise;} \end{cases}$
3. $\mathsf{lift}^{\wp}(\iota)(v) \triangleq \begin{cases} \max_{\sqsubseteq_{\mathsf{pr}(v[1])}}]\{\mathsf{lift}^{\wp}(\iota)((v, v')) : (v, v') \in Mv\}, & \text{if } v \in \mathrm{Ps}_{\wp}; \\ \min_{\sqsubseteq_{\mathsf{pr}(v[1])}}]\{\mathsf{lift}^{\wp}(\iota)((v, v')) : (v, v') \in Mv\}, & \text{otherwise.} \end{cases}$

In their works, the authors of [29] and [43] proved that the set of winning positions $\mathrm{Wn}^{\wp}$ of player $\wp$ in a parity game coincides with the set $\{v \in \mathrm{Ps} : \iota_\omega(v) = \top\}$ of positions with measure $\top$ in the fixpoint measure function, while the remaining ones $\mathrm{Wn}^{\overline{\wp}} = \mathrm{Ps} \setminus \mathrm{Wn}^{\wp}$ are winning for player $\overline{\wp}$.

Thanks to the monotonicity of the successor function, and independently of the definition of the ordered set $(\mathrm{N}, \preceq)$, we can prove that, in a game belonging to any extension of the Core family, the lift of a $\wp$-measure function ι always increments by the least possible quantity the measure associated with position $\gamma_{\wp}$, as stated by the following lemma.

Lemma 5.1. *Let $\mathfrak{I}^k$ be an element in a worst case family $\{\mathfrak{I}^k\}_{k\geq 1}^{\omega}$ and $\wp \in \{0, 1\}$ a player. Then, $\mathsf{lift}^{\wp}(\iota)(\gamma_{\wp}) \in \{\iota(\gamma_{\wp}), \mathsf{suc}_{\sqsubset}(\iota)((\gamma_{\wp}, \gamma_{\wp}))\}$, for any measure function $\iota \in \mathrm{M}^{\wp}$.*

Proof. By Definition 3.1, the position $\gamma_{\wp}$ has priority $\mathsf{pr}(\gamma_{\wp}) = \wp$, but player $\overline{\wp}$. Therefore, due to Definition 5.1, we have that $\mathsf{lift}^{\wp}(\iota)(\gamma_{\wp}) = \min_{\sqsubseteq_{\mathsf{pr}(\gamma_{\wp})}}\{\mathsf{lift}^{\wp}(\iota)((\gamma_{\wp}, v)) : (\gamma_{\wp}, v) \in Mv\}$. Now, if there is successor v of $\gamma_{\wp}$ with $\iota(v) \sqsubset_{\wp} \iota(\gamma_{\wp})$, it holds that $\mathsf{lift}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, v)) = \max_{\sqsubseteq_{\mathsf{pr}(\gamma_{\wp})}}\{\iota(\gamma_{\wp}), \mathsf{suc}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, v))\} = \iota(\gamma_{\wp})$, since $\iota(v) \sqsubset_{\wp} \iota(\gamma_{\wp})$ implies $\mathsf{suc}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, v)) \sqsubseteq_{\mathsf{pr}(\gamma_{\wp})} \iota(\gamma_{\wp})$. Otherwise, $\iota(\gamma_{\wp}) \sqsubseteq_{\mathsf{pr}(\gamma_{\wp})} \iota(v)$, for every possible successor v. Hence, $\mathsf{lift}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, v)) = \max_{\sqsubseteq_{\mathsf{pr}(\gamma_{\wp})}}\{\iota(\gamma_{\wp}), \mathsf{suc}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, v))\} = \mathsf{suc}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, \gamma_{\wp}))$. Observe that, in the last equality we are exploiting the monotonicity of the successor function, *i.e.*, $\mathsf{suc}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, v_1)) \sqsubseteq_{\mathsf{pr}(\gamma_{\wp})} \mathsf{suc}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, v_2))$, if $\iota(v_1) \sqsubseteq_{\mathsf{pr}(\gamma_{\wp})} \iota(v_2)$. At this point, since $\wp \in \{0, 1\}$, it holds that $\gamma_{\wp}$ has the minimal priority of parity $\wp$ in the game, thus, $\sqsubset_{\mathsf{pr}(\gamma_{\wp})} = \sqsubset$. As a consequence, we have that $\mathsf{lift}_{\sqsubset_{\wp}}(\iota)((\gamma_{\wp}, v)) = \mathsf{suc}_{\sqsubset}(\iota)((\gamma_{\wp}, \gamma_{\wp}))$. □

As mentioned above, the Small Progress Measure algorithm and its succinct version only differ in the definitions of the underlying ordered set $(\mathrm{N}, \preceq)$ and of the measure set $\mathrm{M}^{\wp}$. For Small Progress Measure, we have that $(\mathrm{N}, \preceq) \triangleq (\mathbb{N}, \leq)$ and, for all $m \in \mathrm{M}^{\wp} \setminus \{\top\}$ and $d \in \mathrm{D}$, it holds that $m(d) \leq n_d$, where n_d denotes the number of positions with priority d in the game. For the Succinct Progress Measure algorithm, instead, we have that $\mathrm{N} \triangleq \{0, 1\}^*$, $\preceq$ is the lexicographic ordering on bit sequences *w.r.t.* the ordering $0 < \epsilon < 1$, and, for all $m \in \mathrm{M}^{\wp} \setminus \{\top\}$, it holds that $\sum_{d \in \mathrm{D}} |m(d)| \leq \lceil \log_2 n \rceil$, where n is the total number of positions in the game.

We can prove that any worst case family $\{\mathfrak{I}^k\}_{k\geq 1}^{\omega}$ contains difficult instances for the both these algorithms as well. In both cases, this is proved by exploiting Lemma 5.1 and by showing that position $\gamma_{\wp}$ is lifted an exponential (*resp.*, a quasi-polynomial) number of times before a fixpoint is reached.

Lemma 5.2 *(Small progress measure exponential worst case). The number of lifts executed by the Small Progress Measure algorithm for player $\wp \in \{0, 1\}$ on the $(2k + \wp)$-th element of a worst case family is $\Omega\big(6^k\big)$.*

Proof. Given a worst-case family $\{\mathfrak{I}^k\}_{k\geq 1}^{\omega}$, let us fix a player $\wp$ and the subfamily $\{\mathfrak{I}^{2k+\wp}\}_{k\geq 1}^{\omega}$. Moreover, observe that, each game in the subfamily is completely won by player $\wp$. The Small Progress Measure algorithm on a game $\mathfrak{I}^r$ with $r = 2k + \wp$ for player $\wp$ computes the least fixpoint $\iota_\omega = \mu\iota.\mathsf{lift}^{\wp}(\iota)$ of $\mathsf{lift}^{\wp}$, which assigns each position v of the game to $\top$, *i.e.*, $\iota_\omega(v) = \top$. To do this, the algorithm iteratively applies the lift operator, *i.e.*, $\iota_{i+1} = \mathsf{lift}^{\wp}(\iota_i)$, starting from the initial measure function ι_0, which assigns the smallest measure in $\mathrm{M}^{\wp}$. By Lemma 5.1, we know that $\mathsf{lift}^{\wp}(\iota)(\gamma_{\wp}) \in \{\iota(\gamma_{\wp}), \mathsf{suc}_{\sqsubset}(\iota)((\gamma_{\wp}, \gamma_{\wp}))\}$, for every ι. In addition, the priority of $\gamma_{\wp}$ is either 0 or 1, depending on $\wp$. As a consequence, $\sqsubseteq_{\mathsf{pr}(\gamma_{\wp})} = \sqsubseteq$, in other words, no truncation occurs. Therefore, the sequence $\iota_0, \iota_1, \ldots, \iota_h = \iota_\omega$ of lifts performed by the algorithm induces a non-decreasing chain $m_0 = \iota_0(\gamma_{\wp}), m_1 = \iota_1(\gamma_{\wp}), \ldots, m_h = \iota_h(\gamma_{\wp})$ of measures for $\gamma_{\wp}$, which contains a maximal strictly increasing sub-chain $m_{j_0}, m_{j_1}, \ldots, m_{j_z}$, where *(i)* $m_{j_0} = m_0$, *(ii)* $m_{j_{l+1}}$ is the minimal measure greater than m_{j_l}, *i.e.*, $m_{j_{l+1}} = \min_{\sqsubseteq}\{m \in \mathrm{M}^{\wp} : m_{j_l} \sqsubset m\}$, for $l \in [0, z]$, and *(iii)* $m_{j_z} = \top$. Therefore, z is equal to the height of the total ordered set $\mathrm{M}^{\wp}$, which corresponds

to its cardinality. For the game ∂^r, a measure $m \in \mathrm{M}^{\wp} \setminus \{\top\}$ is isomorphic to a $(r+1)$-tuple $\langle b_{k+\wp-1}, \ldots, b_0, t_k, \ldots, t_0 \rangle$ of numbers, with $b_i \in \{0,1\}$ and $t_i \in \mathrm{T} \supseteq \{0,1,2\}$, for some set $\mathrm{T} \subseteq \mathbb{N}$ [29]. Indeed, there is a single position for each one of the $k+\wp$ priorities of parity $\wp$ in the range $[r+\wp+1, 2r+\wp+1]$, while there are at least two positions for the $k+1$ priorities of the same parity in $[0,r]$. The cardinality of $\mathrm{M}^{\wp}$ is, therefore, at least $2^{(k+\wp)}3^{(k+1)} \geq 6^k$. □

Theorem 5.1 *(Small progress measure exponential worst case). The number of lifts executed by the Small Progress Measure algorithm on a game with n positions is* $\Omega\big(6^{\frac{n}{6}}\big)$ *in the worst case.*

Proof. Consider the core family $\{\partial_{\mathrm{C}}^k\}_{k\geq 1}^{\omega}$ and its $\partial_{\mathrm{C}}^{2k+\wp}$ element containing $n = 3(2k+\wp+1)$ positions. By Lemma 5.2, we have that Small Progress Measure performs $\Omega\big(6^k\big)$ lifts, where $k = \frac{n-3\wp-3}{6}$. Consequently, the number of lifts is $\Omega\big(6^{\frac{n}{6}}\big)$. □

Lemma 5.3 *(Succinct progress measure quasi-polynomial worst case). The number of lifts executed by the Succinct Progress Measure algorithm for player* $\wp \in \{0,1\}$ *on the* $(2k+\wp)$*-th element of a worst case family is* $\Omega\big((6k)^{\log k - \log\lceil \log 6k\rceil}\big)$.

Proof. Due to the previously described framework, it is clear that the Succinct Progress Measure algorithm is structurally identical to the Small Progress Measure one, being the measure set the only difference between the two approaches. As a consequence, the number of lifts required by the former algorithm on an element ∂^r with index $r = 2k+\wp$ of a worst-case family $\{\partial^k\}_{k\geq 1}^{\omega}$ is equal to the dimension of its measure set $\mathrm{M}^{\wp}$ as well. Now, every measure $m \in \mathrm{M}^{\wp} \setminus \{\top\}$ is isomorphic to a $(r+1)$-tuple $\langle s_0, \ldots, s_r \rangle$ of binary strings [43], with $s_i \in \{0,1\}^*$ and $\sum_{i=0}^{r} |s_i| = \lceil \log n \rceil$, where $n \geq n^\star \triangleq 3(r+1)$ is the number of positions in ∂^r. For any $h \in [0, \lceil \log n \rceil]$, there are 2^h binary strings $s \in \{0,1\}^h$ of length h and, for each one of them, exactly $\binom{h+r}{h}$ ways to be spit into $(r+1)$ substrings $s_0, \ldots, s_r$. Consequently, the size of the measure set $\mathrm{M}^{\wp}$ is $\sum_{h=0}^{\lceil \log n \rceil} 2^h \binom{h+r}{h} \geq n^\star \binom{\lceil \log n^\star \rceil + r}{\lceil \log n^\star \rceil} \geq n^\star \left(\frac{\lceil \log n^\star \rceil + r}{\lceil \log n^\star \rceil}\right)^{\lceil \log n^\star \rceil} \geq n^\star \left(\frac{r}{\lceil \log n^\star \rceil}\right)^{\lceil \log n^\star \rceil} \geq n^\star \left(\frac{r}{\lceil \log n^\star \rceil}\right)^{\log n^\star}$. Since $r \geq \frac{n^\star}{6}$, for any $k \in \mathbb{N}_+$, we have

$$
\begin{aligned}
|\mathrm{M}^{\wp}| &\geq n^\star \left(\frac{n^\star}{6\lceil \log n^\star \rceil}\right)^{\log n^\star} = n^\star \left(\frac{n^\star}{n^{\star \log_{n^\star}(6\lceil \log n^\star \rceil)}}\right)^{\log n^\star} \\
&= n^{\star (\log n^\star)(1 - \log_{n^\star}(6\lceil \log n^\star \rceil)) + 1} = n^{\star \log n^\star - \log(6\lceil \log n^\star \rceil)) + 1} \\
&= n^{\star \log n^\star - \log\lceil \log n^\star \rceil - \log 6 + 1}.
\end{aligned}
$$

Finally, since $n^\star > 6k$, we have that

$$
\begin{aligned}
|\mathrm{M}^{\wp}| &\geq (6k)^{\log 6k - \log\lceil \log 6k \rceil - \log 6 + 1} = (6k)^{\log k - \log\lceil \log 6k \rceil + 1} \\
&= \Omega\big((6k)^{\log k - \log\lceil \log 6k \rceil}\big). \quad \square
\end{aligned}
$$

Theorem 5.2 *(Succinct progress measure quasi-polynomial worst case). The number of lifts executed by the Succinct Progress Measure algorithm, on a game with n positions is* $\Omega\big((n-6)^{\log \frac{n}{6} - \log\lceil \log n \rceil}\big)$ *in the worst case.*

Proof. Consider the core family $\{\partial_{\mathrm{C}}^k\}_{k\geq 1}^{\omega}$ and its $\partial_{\mathrm{C}}^{2k+\wp}$ element containing $n = 3(2k+\wp+1)$ positions. By Lemma 5.3, we have that Succinct Progress Measure performs $\Omega\big((6k)^{\log k - \log\lceil \log 6k \rceil}\big)$ lifts, where $k = \frac{n-3\wp-3}{6}$. Consequently, the number of lifts is $\Omega\big((n-6)^{\log \frac{n}{6} - \log\lceil \log n \rceil}\big)$. □

The quasi-polynomial time algorithm proposed in [44] is yet another instance of a progress measure based approach and can easily be recast in the above framework, by a suitable definition of the measure set $\mathrm{M}^{\wp}$, the truncation operation $m\downarrow_p$, and the successor function $\mathrm{suc}_{\lhd}$. Unlike the two algorithm discussed above, however, the resulting successor function does not enjoy the monotonicity property, as pointed out in [44]. As a consequence, Lemma 5.1 does not hold for this approach. Nonetheless, we conjecture that the number of lifts for position $\gamma_{\wp}$ is still quasi-polynomial in the index of the game and a result analogous to Theorem 5.2 holds in this case as well.

6. SCC-decomposition resilient games

The previous sections provide a class of parametric families of parity games over which a dynamic-programming approach cannot help improving the asymptotic exponential behavior of the classic Recursive algorithm and that serves an exponential, for SMP, or a quasi-polynomial, for Succinct SMP, lower bound for the progress measures based algorithms.

The core family, however, is not robust enough to resist to game decomposition techniques such as, SCC-decomposition. In particular, it is not hard to observe that for the Recursive algorithm a SCC-decomposition of the underlying game graph, if applied by each recursive call as described in [46], would disrupt the recursive structure of the core family $\{\partial_{\mathrm{C}}^k\}_{k=1}^{\omega}$ and,

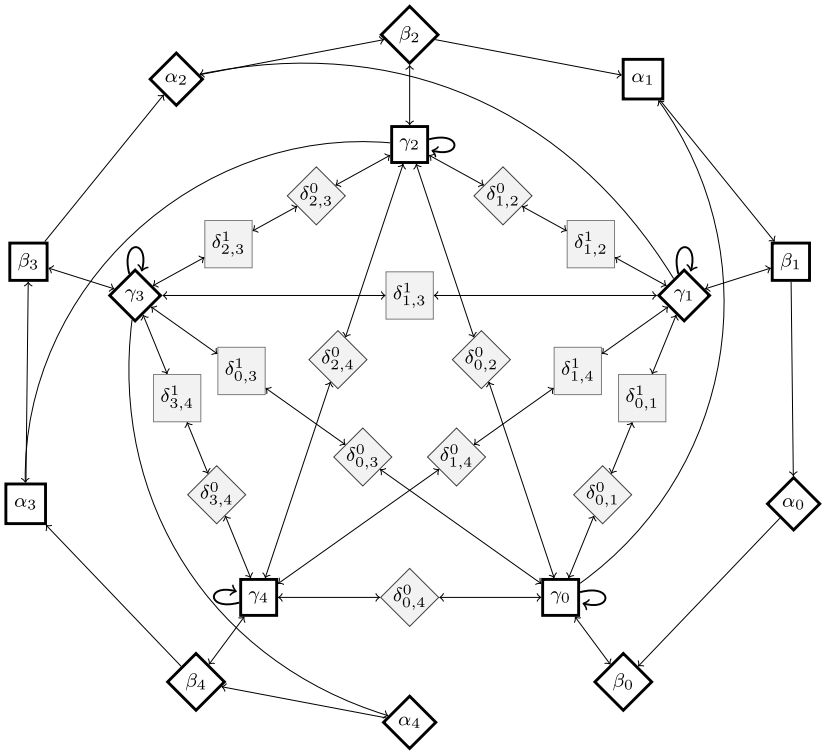

Fig. 4. Game $\mathfrak{D}^4_{\mathrm{S}}$ of the SCC family.

consequently, break the exponential worst-case. This is due to the fact that the subgames $\mathfrak{D}^k_w$ in the induced subgame tree get decomposed into distinct SCCs, which can then be solved as independent subgames and memoized. In other words, the Recursive algorithm extended with memoization and SCC-decomposition can easily solve the core family. A concrete instance of this behavior can be observed by looking at the two games $\widehat{\mathfrak{D}}^1_{\mathrm{LL}}$ and $\widehat{\mathfrak{D}}^1_{\mathrm{RL}}$ of Fig. 2. The first one is formed by three distinct components, one of which exactly corresponds to $\widehat{\mathfrak{D}}^1_{\mathrm{RL}}$. Therefore, a solution of these components immediately implies that the leaves in the induced subgame tree could not be considered distinct games *w.r.t.* to the behavior of the combined algorithm anymore. This behavior can, however, be prevented by introducing a suitable extension of the core family, which complies with the requirements of Definition 3.2. The basic idea is to connect all the pairs of positions γ_i and γ_j together in a clique-like fashion, by means of additional positions, denoted $\delta^{\wp}_{\{i,j\}}$ with $\wp \in \{0,1\}$, whose owners $\wp$ are chosen so as to preserve the exponential behavior on the underlying core family. With more detail, if $i \equiv_2 j$, there is a unique connecting position $\delta^{\wp}_{\{i,j\}}$ of parity $\wp \equiv_2 i$, the opposite of that of γ_i and γ_j. If, on the other hand, $i \not\equiv_2 j$, two mutually connected positions, $\{\delta^0_{\{i,j\}}, \delta^1_{\{i,j\}}\}$, separate γ_i and γ_j. Fig. 4 depicts the extension $\mathfrak{D}^4_{\mathrm{S}}$ of the core game $\mathfrak{D}^4_{\mathrm{C}}$. The grey-filled positions in the figure enforce mutual connection among the nodes of the core in any subgame of the subgame tree, while the remaining positions are exactly those of the Core family and connected among them in the same way. The complete formalization of the new family follows.

Definition 6.1 *(SCC family).* The *SCC family* $\{\mathfrak{D}^k_{\mathrm{S}}\}^{\omega}_{k=1}$, where $\mathfrak{D}^k_{\mathrm{S}} \triangleq \langle \mathcal{A}, \mathrm{Pr}, \mathrm{pr}\rangle$, $\mathcal{A} \triangleq \langle \mathrm{Ps}^0, \mathrm{Ps}^1, Mv\rangle$, $\mathrm{Pr} \triangleq [0, 2k+1+\wp]$, and $\mathrm{Ps} \triangleq \mathfrak{D}^k_{\mathrm{C}} \cup \mathrm{P}$, is defined, for any index $k \geq 1$, as follows:

1. $\mathrm{P} \triangleq \{\delta^{\wp}_{\{i,j\}} : \wp \in \{0,1\} \wedge i, j \in [0,k] \wedge i \neq j \wedge (i \equiv_2 j \rightarrow \wp \equiv_2 i)\}$;
2. $(\gamma_i, \delta^{\wp}_{\mathrm{I}}), (\delta^{\wp}_{\mathrm{I}}, \gamma_i) \in Mv$ *iff* $i \in \mathrm{I}$ and $i \equiv_2 \wp$, for $i \in [0,k]$ and $\delta^{\wp}_{\mathrm{I}} \in \mathrm{P}$;
3. $(\delta^{\wp}_{\mathrm{I}}, \delta^{\overline{\wp}}_{\mathrm{I}}) \in Mv$ *iff* $i \not\equiv_2 j$, for $\delta^{\wp}_{\mathrm{I}} \in \mathrm{P}$ and $\mathrm{I} = \{i, j\}$;
4. $\delta^{\wp}_{\mathrm{I}} \in \mathrm{Ps}^{\wp}$ and $\mathrm{pr}(\delta^{\wp}_{\mathrm{I}}) \triangleq 0$, for $\delta^{\wp}_{\mathrm{I}} \in \mathrm{P}$;
5. $\mathfrak{D}^k_{\mathrm{S}} \setminus \mathrm{P} = \mathfrak{D}^k_{\mathrm{C}}$.

Intuitively, in Item 1, P denotes the set of additional positions of $\mathfrak{D}^k_{\mathrm{S}}$ *w.r.t.* to the core family game $\mathfrak{D}^k_{\mathrm{C}}$, which is, indeed, a proper subgame, as stated in Item 5. Item 2, instead, formalizes the moves connecting the additional $\delta^{\wp}_{\mathrm{I}}$ positions with the γ_i of the core, while Item 3 describes the mutual connection between the δ positions that share the same doubleton

as indexes I. Finally, Item 4 associates each $\delta_{\mathrm{I}}^{\wp}$ with its corresponding owner $\wp$ and priority 0. The following lemma proves that such a parity-game family is indeed a worst-case family.

Lemma 6.1. *The SCC family $\{\partial_{\mathrm{S}}^k\}_{k=1}^{\omega}$ is a worst-case family.*

Proof. To prove that the *SCC family* of Definition 6.1 is a worst-case family, we need to show that each game ∂_{S}^k is a core extension of ∂_{C}^k, *i.e.*, that it complies with the Definition 3.2. Items 1 and 5 of Definition 6.1 imply Item 1 of Definition 3.2, since the set P does not contain any position of the core and, in addition, ∂_{C}^k is a subgame of ∂_{S}^k, as all the positions and moves of the core are contained in ∂_{S}^k. Item 2 of Definition 3.2 follows from Item 4 of Definition 6.1. Indeed, by Definition 3.1, $\mathrm{pr}(\alpha_i) \triangleq k + i + \wp + 1$, for $i \in_{0,k}$ and $k \geq 1$, while all the additional positions in P have priority 0. By Items 2 and 3 of Definition 6.1, there are no moves connecting positions δ_{I} with positions α_i or β_i, for any $i \in [0,k]$, hence Item 3 of Definition 3.2 is satisfied. Finally, we need to show that whenever a γ_i has a move to a position v in P, then v does not have higher priority, belongs to the opponent of γ_i, and has a move back to γ_i (Item 4 of Definition 3.2). This property is enforced by Items 2 and 4 of Definition 6.1. Indeed, by the latter, position $\delta_{\mathrm{I}}^{\wp}$ is owned by player $\wp$. Moreover, by the former, γ_i, whose owner is $(i+1) \bmod 2$, can only have a move to $\delta_{\{i,j\}}^{\wp}$ if $\delta_{\{i,j\}}^{\wp}$ has a move back to γ_i and $\wp = i \bmod 2$. Hence, the two positions belong to opposite players. Since, in addition, all positions δ have priority 0, the requirement is satisfied. □

Due to the clique-like structure of the new family, it is not hard to see that every game in the induced subgame tree forms a single SCC. This guarantees that the intertwining of SCC-decomposition and memoization cannot prevent an exponential worst-case behavior of the Recursive algorithm on this family.

Lemma 6.2. *Each game in the induced subgame tree $\mathbb{G}^k$ of the SCC family $\{\partial_{\mathrm{S}}^k\}_{k=1}^{\omega}$, for an arbitrary index $k \in \mathbb{N}_+$, forms a single SCC.*

Proof. Let $\partial_w^k \in \mathbb{G}^k$ (*resp.*, $\widehat{\partial}_w^k \in \mathbb{G}^k$) be a game in the induced subgame tree. By induction on the structure of the string w, it is not hard to see that, for all indexes $i, j \in [0,k]$ with $i \neq j$, it holds that $\gamma_i, \gamma_j \in \partial_w^k$ (*resp.*, $\gamma_i, \gamma_j \in \widehat{\partial}_w^k$) iff the positions $\delta_{\{i,j\}}^{\wp} \in \mathrm{P}$, with $\wp \in \{0,1\}$, belong to ∂_w^k (*resp.*, $\widehat{\partial}_w^k$), as well. For the base case $\partial_{\varepsilon}^k = \partial^k$, the thesis trivially follows from Definition 6.1. For the inductive case $\widehat{\partial}_{wx}^k = \partial_w^k \setminus \mathrm{A}$ (*resp.*, $\partial_{wx}^k = \widehat{\partial}_w^k \setminus \mathrm{A}$), let us assume, as inductive hypothesis, that the statement holds for ∂_w^k (*resp.*, $\widehat{\partial}_w^k$). By Definition 4.1, the set A is computed as the attractor to some set of positions B such that either *(i)* $\gamma_i, \gamma_j, \delta_{\{i,j\}}^{\wp} \notin \mathrm{A}$ or *(ii)* $\delta_{\{i,j\}}^{\wp} \in \mathrm{A}$ and at least one between γ_i and γ_j belongs to A, for all $i, j \in [0,k]$. Case *(i)* arises when $\mathrm{B} = \{\alpha_{k-2|w|}\}$ (*resp.*, $\mathrm{B} = \{\alpha_{k-2|w|+1}\}$), while Case *(ii)* when $\mathrm{B} = \mathrm{Wn}_0(\widehat{\partial}_{w\mathbb{L}}^k)$. Consequently, the required property on $\widehat{\partial}_{wx}^k$ (*resp.*, ∂_{wx}^k) immediately follows from the inductive hypothesis on w, since, if a position $\delta_{\{i,j\}}^{\wp}$ is removed from the game, also one between γ_i and γ_j is removed as well and *vice versa*. Now, let ∂ be an arbitrary game in $\mathbb{G}^k$. Thanks to the topology of the games in the SCC family and to the property on $\gamma_i, \gamma_j \in \partial_w^k$ (*resp.*, $\gamma_i, \gamma_j \in \widehat{\partial}_w^k$), it is easy to see that all positions in $\mathrm{X} = \{\beta_i, \gamma_i, \delta_{\mathrm{I}}^{\wp} \in \partial\}$ form a strongly connected subgame. Indeed, two positions β_i and γ_i are mutually reachable due to the two moves $(\beta_i, \gamma_i), (\gamma_i, \beta_i) \in M v$. Moreover, two arbitrary positions γ_i and γ_j, with $i \neq j$, are mutually reachable via the positions $\delta_{\{i,j\}}^{\wp}$. Also, there are no isolated positions $\delta_{\{i,j\}}^{\wp}$. Finally, to prove that ∂ is indeed a single SCC, it remains just to show that the positions in $\mathrm{Y} = \{\alpha_i \in \partial\}$ can reach and can be reached by those in X. The first part is implied by Item 1 of Lemma 4.1, since every α_i has only a move to β_i, which needs to belong to ∂ in order for this to be a game. Now, due to the same observation, all positions α_i, but possibly the last one α_m with maximal index m in ∂, are reachable by β_{i+1}. Finally, α_m can be reached by γ_{m-1}, which necessarily belongs to ∂ due to Item 3 of the same lemma. □

Putting everything together, we obtain the following structural, although non asymptotic, strengthening of Theorem 4.1.

Theorem 6.1 *(SCC-decomposition worst case). The number of distinct recursive calls executed by the Recursive algorithm with SCC decomposition on a game with n positions is $\Omega\big(2^{\sqrt{n/3}}\big)$ in the worst case.*

Proof. Due to Lemma 6.1, the *SCC family* is an exponential worst-case family. As shown in the proof of Theorem 4.1, the Recursive algorithm performs at least $3(2^{\lfloor k/2 \rfloor + 1} - 1)$ calls to solve a game of $\{\partial^k\}_{k=1}^{\omega}$, and therefore, of $\{\partial_{\mathrm{S}}^k\}_{k=1}^{\omega}$. As consequence of Lemma 6.2, the number of calls cannot be affected by an SCC-decomposition technique, since there is an exponential number of subgames, the ones in the induced subgame tree of ∂_{S}^k, each of which forms a single SCC. Moreover, a core game ∂_{C}^k has $3(k+1)$ positions, while the number of additional positions P in the extension ∂_{C}^k is given by $\frac{3k^2+\wp}{4} + k$, which follows from Definition 6.1. Indeed, for every pair of positions γ_i, γ_j, with $i, j \in [0,k]$ and $i \neq j$, there is a single position $\delta_{\{i,j\}}^{\wp}$, if $i \equiv_2 j$, and two such positions, otherwise. Hence, ∂_{S}^k has $n \triangleq \frac{3k^2+\wp}{4} + 4k + 3$ positions, from which we obtain that $k = \frac{\sqrt{4(3n+7)-3\wp}-8}{3} = \frac{\lceil\sqrt{4(3n+7)-3}\rceil-8}{3} = \frac{\lceil\sqrt{12n+25}\rceil-8}{3}$. As a consequence, the number of recursive calls is bounded from below by $3\left(2^{\lfloor \frac{\lceil\sqrt{12n+25}\rceil-2}{6} \rfloor}\right) = \Omega\big(2^{\sqrt{n/3}}\big)$. □

7. Dominion-decomposition resilient games

A deeper analysis of the SCC family reveals that the size of the smallest dominion for player 0 in Game $\mathcal{D}_S^k$ (there are no dominions for player 1, being the game completely won by its opponent) is of size $k+1$. This observation, together with the fact that the game has $n = \frac{3k^2+\wp}{4} + 4k + 3 < (k+1)^2$ positions, immediately implies that the proposal of [36,37] of a brute force search for dominions of size at most $\lceil\sqrt{n}\rceil < k+1$ cannot help improving the solution process on these games. We can prove an even stronger result, since this kind of search cannot reduce the running time in any of the subgames of the induced tree. The reason is that the smallest dominion in each such subgame that contains at least a position in the core has size linear *w.r.t.* k, as reported by the following lemma.

Lemma 7.1. *For all $k \in \mathbb{N}_+$, let $\mathcal{D}$ be a subgame in the induced subgame tree $\mathbb{G}^k$ of $\mathcal{D}_S^k$, and $D \subseteq \mathcal{D}$ the smallest dominion such that $D \cap \mathcal{D}_C^k \neq \emptyset$. Then, $|D| \geq \lfloor k/2 \rfloor + 1$.*

Proof. We start by proving that any such dominion D must necessarily contain at least a position γ_i, from some $i \in \mathbb{N}_+$. Assume, by contradiction, that it does not. Then, $D \subseteq \{\alpha_i, \beta_i \in \mathcal{D} : i \in [0,k]\} \cup P$. We can prove that D is not a game. By assumption, D must contain at least one position α_i or β_i, otherwise $D \cap \mathcal{D}_C^k = \emptyset$. Let j be the smallest index such that $\alpha_j \in D$ or $\beta_j \in D$. Then, at least one of those two positions has only moves leading outside D. Hence D is not a game and, *a fortiori*, cannot be a dominion.

Therefore, D must contain at least a position γ_i. By Definition 6.1, γ_i is connected in $\mathcal{D}_S^k$ to precisely k positions $\delta_{\{i,j\}}^{\wp}$, where $j \in [0,k]$, $j \neq i$, and whose owner $\wp \equiv_2 i$ is the adversary of the owner $\overline{\wp}$ of γ_i. Let us consider the $\lfloor k/2 \rfloor$ indexes $j \in [0,k]$ such that $i \not\equiv_2 j$. For each such j, we have two cases, depending on whether γ_j belongs to $\mathcal{D}$ or not. If it does, then so does $\delta_{\{i,j\}}^{\wp}$, since both are owned by the same player $\wp$ and are mutually connected. If it does not, then it must have been removed by some application of Item 2 of Definition 4.1. If the involved attractor *w.r.t.* some player $\wp'$ does not attract $\delta_{\{i,j\}}^{\overline{\wp}}$, then it cannot attract $\delta_{\{i,j\}}^{\wp}$ either, as it has no moves to γ_j. If, on the other hand, $\delta_{\{i,j\}}^{\overline{\wp}}$ gets attracted, it must belong to player $\wp'$. As a consequence, $\delta_{\{i,j\}}^{\wp}$ belongs to the opponent player $\overline{\wp'}$, in other words $\wp = \overline{\wp'}$. In either case, we conclude hat $\delta_{\{i,j\}}^{\wp}$ cannot be removed and, therefore, is still contained in $\mathcal{D}$. In addition, all the $\lfloor k/2 \rfloor$ positions $\delta_{\{i,j\}}^{\wp}$, with $j \in_{0,2k}$ and $j \not\equiv_2 i$, are mutually connected to γ_i and their owner is the opponent of the one of γ_i. It is immediate to see that if i is the greatest index such that $\gamma_i \in \mathcal{D}$, then it is contained in the smallest dominion D in $\mathcal{D}$, together with the $\lfloor k/2 \rfloor$ positions $\delta_{\{i,j\}}^{\wp}$ connected to it. Hence, D contains at least $\lfloor k/2 \rfloor + 1$ positions. □

The above observation allows us to obtain an exponential lower bound for the Recursive algorithm combined with memoization, SCC decomposition, and dominion decomposition techniques. Indeed, the brute-force procedure employed by the Dominion Decomposition algorithm of [36] needs at least time $\Omega\left(2^{\lfloor k/2 \rfloor + 1}\right)$ to find a dominion of size $\lfloor k/2 \rfloor + 1$.

Inspired by the Dominion Decomposition approach, in a game with n positions and p priorities, the BigStep algorithm [39] tries to find a dominion of size bounded by $\pi = \sqrt[3]{pn^2}$ before each call to the internal recursive algorithm. Unlike Dominion Decomposition, however, the search is not performed via a brute-force procedure. Big Step employs, instead, a modified version of the SPM algorithm in which the search space of the possible measures evaluated during the lift computation is suitably limited by the parameter π. Moreover, this procedure, called *approximate*, does not look for an arbitrary dominion, but for one of the opposite player $\wp$ *w.r.t.* the parity of the maximal priority in the current subgame. Since player $\wp$ is also the winning player of a game $\mathcal{D}_S^k$ in the SCC family, the approximate SMP procedure is called *w.r.t.* player $\overline{\wp}$ so that, if a fixpoint different from $\top$ is reached, a $\wp$-dominion of the desired size exists. For $\mathcal{D}_S^k$, such a $\wp$-dominion actually exists, therefore a fixpoint different from $\top$ would be reached, thus making the argument in the proof of Lemma 5.2 not applicable. The following lemma, however, shows that even if a fixpoint is reached, the number of lifts performed is still exponential in the index of the game.

Lemma 7.2. *The solution time of the Big Step approximate subroutine on an element $\mathcal{D}_S^k$ of a worst-case family $\{\mathcal{D}_S^k\}_{k\geq 1}^{\omega}$ is $\Omega\left(6^{\frac{k}{2}}\right)$.*

Proof. The exploited SPM variant on the game $\mathcal{D}_S^k$ for player $\overline{\wp} = \mathrm{pr}(\alpha_k) \bmod 2 = 1 - k \bmod 2$ uses measures $m \in M^{\overline{\wp}} \setminus \{\top\}$ such that $\sum_{d \in \mathrm{dom}(m)} m(d) < \pi$, where $\pi = \sqrt[3]{pn^2}$ with $p = 2(k+1)$ and $n = \frac{3k^2+\wp}{4} + 4k + 3$. Similarly to the argument in the proof of Lemma 5.2, every such measure is isomorphic to a $(k+1)$-tuple $\langle b_{\frac{k-\wp}{2}}, \ldots, b_0, t_{\frac{k+\wp}{2}-1}, \ldots, t_0 \rangle$ of numbers, with $b_i \in \{0,1\}$, for all $i \in [0, \frac{k-\wp}{2}]$, $t_i \in \{0,1,2\}$, for all $i \in [1, \frac{k+\wp}{2} - 1]$, and $t_0 \in [0, \frac{3k^2+\wp}{4} + k + 2]$, if $\wp = 1$, and $t_0 \in \{0,1,2\}$, otherwise. Note, indeed, that a game of the SCC family of index k contains precisely $\frac{3k^2+\wp}{4} + k + 2$ nodes with priority 0. Therefore, it is easy to observe that all measures m with $t_0 \in \{0,1,2\}$ satisfy the above restriction, since $\sum_{d \in \mathrm{dom}(m)} m(d) \leq \sum_{i=0}^{\frac{k-\wp}{2}} 1 + \sum_{i=0}^{\frac{k+\wp}{2}-1} 2 = \frac{k-\wp}{2} + 1 + 2\frac{k+\wp}{2} = \frac{3k+\wp}{2} + 1 < \pi$. Now, as noted above, the smallest $\wp$-dominion contained into $\mathcal{D}_S^k$ has size $k+1 < \pi$, hence, the approximate subroutine will find it once a fixed point $\iota_\omega = \mu\iota.\mathsf{lift}^{\wp}(\iota)$ of the $\mathsf{lift}^{\wp}$ function is reached. Due to the structure of the core family, such a fixed point maps all positions $\alpha_i, \beta_i, \gamma_i$ to the following measures:

- for all $i \in [0, \frac{k-\wp}{2}]$ and $d \in [0, 2k+1+\wp]$ with $d \equiv_2 \overline{\wp}$, it holds that $\iota_\omega(\alpha_{k-2i})(d) = \iota_\omega(\alpha_{k-2i-1})(d) = 1$, if $d \geq 2(k-i)+1+\wp$, and $\iota_\omega(\alpha_{k-2i})(d) = \iota_\omega(\alpha_{k-2i-1})(d) = 0$, otherwise; intuitively, the measures associated with the positions α_{k-2i} and α_{k-2i-1} are isomorphic to the $(k+1)$-tuple $\langle 1, \ldots, 1, 0, \ldots, 0\rangle$, where only the first $i+1$ components are set to 1, while the remaining ones to 0;
- for all $d \in [0, 2k+1+\wp]$ with $d \equiv_2 \overline{\wp}$, it holds that $\iota_\omega(\beta_k)(d) = \iota_\omega(\gamma_k)(d) = 0$; intuitively, β_k and γ_k belong to the $\wp$-dominion contained into $\mathcal{D}_S^k$ and no lift is ever performed on their measures;
- for all $i \in [0, \frac{k-\wp}{2}[$, it holds that $\iota_\omega(\beta_{2i+\wp}) = \iota_\omega(\gamma_{2i+\wp}) = \iota_\omega(\alpha_{2i+\wp+1})$; intuitively, the positions $\beta_{2i+\wp}$ and $\gamma_{2i+\wp}$ just propagate the measure associated with the $\alpha_{2i+\wp+1}$ position to which they are connected, since the parity of their priority is not congruent to the player $\overline{\wp}$ *w.r.t.* which the execution of the algorithm is performed;
- for all $i \in [0, \frac{k-\wp}{2}]$ and $d \in [0, 2k+1+\wp]$ with $d \equiv_2 \overline{\wp}$, it holds that $\iota_\omega(\gamma_{2i+\overline{\wp}})(d) = 1$, if $d \geq 2(k-i)+1+\wp$ or $d = 2i+\overline{\wp}$, and $\iota_\omega(\gamma_{2i+\overline{\wp}})(d) = 0$, otherwise; intuitively, the measure associated with $\gamma_{2i+\overline{\wp}}$ needs to be the smallest one greater than the one associated with $\alpha_{2i+\overline{\wp}+1}$ *w.r.t.* the truncation priority $2i+\overline{\wp}$, since the parity of the latter is congruent to the player $\overline{\wp}$;
- finally, for all $i \in [0, \frac{k-\wp}{2}]$ and $d \in [0, 2k+1+\wp]$ with $d \equiv_2 \overline{\wp}$, it holds that $\iota_\omega(\beta_{2i+\overline{\wp}})(d) = 1$, if $d \geq 2(k-i)+1+\wp$, $\iota_\omega(\beta_{2i+\overline{\wp}})(d) = 2$, if $d = 2i+\overline{\wp}$, and $\iota_\omega(\beta_{2i+\overline{\wp}})(d) = 0$, otherwise; intuitively, the measure associated with $\beta_{2i+\overline{\wp}}$ needs to be the smallest one greater than the one associated with $\gamma_{2i+\overline{\wp}}$ *w.r.t.* the truncation priority $2i+\overline{\wp}$, since the parity of the latter is congruent to the player $\overline{\wp}$.

By applying Lemma 5.1, we can observe that, during the computation of the lifts and before reaching the final measure $\iota_\omega(\gamma_{\overline{\wp}})$, position $\gamma_{\overline{\wp}}$ is assigned all measures in $\mathrm{M}^\wp \setminus \{\top\}$ smaller than or equal to $\iota_\omega(\gamma_{\overline{\wp}})$ that satisfy the previously described restriction. Now, it is easy to see that there are at least $2^\wp \cdot 6^{\frac{k+\wp}{2}} - 3^{\frac{k+\wp}{2}} + 1$ measures $m \in \mathrm{M}^\wp \setminus \{\top\}$ with $m(\overline{\wp}) \in \{0,1,2\}$ smaller than $\iota_\omega(\gamma_{\overline{\wp}})$. Indeed, *(i)* every such measure can be interpreted as the following natural number $3^{\frac{k+\wp}{2}}\left(\sum_{i=0}^{\frac{k-\wp}{2}} 2^i \cdot m(k+1+\wp+2i)\right) + \left(\sum_{i=0}^{\frac{k+\wp}{2}-1} 3^i \cdot m(2i+\overline{\wp})\right)$, *(ii)* every lift of $\gamma_{\overline{\wp}}$ increments the value associated with its measure just by one, and *(iii)* the value of $\iota_\omega(\gamma_{\overline{\wp}})$ is precisely $3^{\frac{k+\wp}{2}}\left(\sum_{i=0}^{\frac{k-\wp}{2}} 2^i\right) + 1 = 3^{\frac{k+\wp}{2}}\left(2^{\frac{k-\wp}{2}+1} - 1\right) + 1 = 2^{\overline{\wp}} \cdot 6^{\frac{k+\wp}{2}} - 3^{\frac{k+\wp}{2}} + 1 = \Omega\left(6^{\frac{k}{2}}\right)$. □

As a consequence, none of the dominion decomposition approaches, combined with memoization and SCC decomposition, can efficiently solve the SCC family.

Corollary 7.1 *(Exponential dominion-decomposition worst case). The solution time of the Recursive algorithm with memoization, SCC decomposition, and dominion decomposition on a game with n positions is* $\Omega\left(2^{\Theta(\sqrt{n})}\right)$ *in the worst case.*

8. Discussion

Existence of a polynomial-time solution algorithm for Parity Game is a long standing open problem ever since [14,4]. The revival of interest on the subject stems from the recent result of Calude et al. [8], which shows that the problem can be solved in quasi-polynomial time. That result led to two quasi-polynomial-time algorithms, proposed by Jurdziński and Lazic [43] and Fearnley et al. [44]. Both these algorithms essentially look for a compact form of progress measure, whose existence provides a winning strategy for one of the players. Their quasi-polynomial complexity derives from the compactness of the representation of the corresponding measure space employed. However, the practical effectiveness of the resulting procedures, as assessed in [52,57], does not seem to rival with existing exponential algorithms, such as the Recursive algorithm or Priority Promotion. More importantly, recent results (see, *e.g.*, [58]) also suggest that these compactness results may not be further improved, thereby raising the question whether the road to a polynomial solution may lie somewhere else. We believe that a better understanding of the weaknesses of the existing algorithmic solutions may provide deeper insights into the problem and potentially lead to new solutions.

The contribution of this paper towards this end is a family of parity games, which, despite their simplicity, proves to be quite challenging for different solution approaches and provide a lower bound on the execution time for both exponential and quasi-polynomial algorithms. The family is also resilient to effective heuristics, such as memoization and game decomposition techniques, that, in some cases, can be used to break existing lower bounds. In particular, we have shown that solving games in this family requires exponential time for the Recursive algorithm, regardless of whether it is endowed with memoization, SCC-decomposition or dominion decomposition techniques. The family also provides a lower bound to progress measure based approaches. We have shown that solving these games requires exponential time for Small Progress Measure [29] and quasi-polynomial time for its succinct version [43], for which no lower bound was available. We conjecture that the same family also requires quasi-polynomial time for QPT, the quasi-polynomial algorithm proposed in [44]. The game family studied in this paper can, however, be solved in polynomial time by all the Priority Promotion-based algorithms [52,41,42] and by Strategy Improvement [59]. While distinct lower bounds exist for these two approaches as well,

a simple and uniform worst case family that encompasses all the solution algorithms, possibly including the more recent quasi-polynomial algorithms proposed in [48,51], is left as future work.

Declaration of competing interest

No known competing financial interests.

References

[1] M. Benerecetti, D. Dell'Erba, F. Mogavero, Robust exponential worst cases for divide-et-impera algorithms for parity games, in: Games, Automata, Logics, and Formal Verification'17, in: EPTCS, vol. 256, 2017, pp. 121–135.
[2] A. Mostowski, Games with Forbidden Positions, Tech. rep., University of Gdańsk, Gdańsk, Poland, 1991.
[3] E. Emerson, C. Jutla, A. Sistla, On model checking for the muCalculus and its fragments, Theor. Comput. Sci. 258 (1–2) (2001) 491–522.
[4] E. Emerson, C. Jutla, Tree automata, muCalculus, and determinacy, in: Foundation of Computer Science'91, IEEE Computer Society, 1991, pp. 368–377.
[5] A. Martin, Borel determinacy, Ann. Math. 102 (2) (1975) 363–371.
[6] A. Martin, A purely inductive proof of Borel determinacy, in: Symposia in Pure Mathematics'82, Recursion Theory, American Mathematical Society and Association for Symbolic Logic, 1985, pp. 303–308.
[7] M. Jurdziński, Deciding the winner in parity games is in UP ∩ co-UP, Inf. Process. Lett. 68 (3) (1998) 119–124.
[8] C. Calude, S. Jain, B. Khoussainov, W. Li, F. Stephan, Deciding parity games in quasipolynomial time, in: Symposium on Theory of Computing'17, Association for Computing Machinery, 2017, pp. 252–263.
[9] A. Ehrenfeucht, J. Mycielski, Positional strategies for mean payoff games, Int. J. Game Theory 8 (2) (1979) 109–113.
[10] V. Gurvich, A. Karzanov, L. Khachivan, Cyclic games and an algorithm to find minimax cycle means in directed graphs, USSR Comput. Math. Math. Phys. 28 (5) (1990) 85–91.
[11] U. Zwick, M. Paterson, The complexity of mean payoff games on graphs, Theor. Comput. Sci. 158 (1–2) (1996) 343–359.
[12] A. Condon, The complexity of stochastic games, Inf. Comput. 96 (2) (1992) 203–224.
[13] E. Emerson, C.-L. Lei, Temporal reasoning under generalized fairness constraints, in: Symposium on Theoretical Aspects of Computer Science'86, in: LNCS, vol. 210, Springer, 1986, pp. 267–278.
[14] A. Mostowski, Regular expressions for infinite trees and a standard form of automata, in: Symposium on Computation Theory'84, in: LNCS, vol. 208, Springer, 1984, pp. 157–168.
[15] O. Kupferman, M. Vardi, Weak alternating automata and tree automata emptiness, in: Symposium on Theory of Computing'98, Association for Computing Machinery, 1998, pp. 224–233.
[16] E. Grädel, W. Thomas, T. Wilke, Automata, Logics, and Infinite Games: A Guide to Current Research, LNCS, vol. 2500, Springer, 2002.
[17] T. Wilke, Alternating tree automata, parity games, and modal muCalculus, Bull. Belg. Math. Soc. 8 (2) (2001) 359–391.
[18] S. Schewe, B. Finkbeiner, Satisfiability and finite model property for the alternating-time muCalculus, in: Computer Science Logic'06, in: LNCS, vol. 6247, Springer, 2006, pp. 591–605.
[19] R. Alur, T. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (5) (2002) 672–713.
[20] S. Schewe, ATL* satisfiability is 2ExpTime-complete, in: International Colloquium on Automata, Languages, and Programming'08, in: LNCS, vol. 5126, Springer, 2008, pp. 373–385.
[21] K. Chatterjee, T. Henzinger, N. Piterman, Strategy logic, Inf. Comput. 208 (6) (2010) 677–693.
[22] F. Mogavero, A. Murano, M. Vardi, Reasoning about strategies, in: Foundations of Software Technology and Theoretical Computer Science'10, in: LIPIcs, vol. 8, Leibniz-Zentrum fuer Informatik, 2010, pp. 133–144.
[23] F. Mogavero, A. Murano, G. Perelli, M. Vardi, What makes ATL* decidable? A decidable fragment of strategy logic, in: Concurrency Theory'12, in: LNCS, vol. 7454, Springer, 2012, pp. 193–208.
[24] F. Mogavero, A. Murano, G. Perelli, M. Vardi, Reasoning about strategies: on the model-checking problem, Trans. Comput. Log. 15 (4) (2014) 34.
[25] F. Mogavero, A. Murano, G. Perelli, M. Vardi, Reasoning about strategies: on the satisfiability problem, Log. Methods Comput. Sci. 13 (1:9) (2017) 1–37.
[26] M. Benerecetti, F. Mogavero, A. Murano, Substructure temporal logic, in: Logic in Computer Science'13, IEEE Computer Society, 2013, pp. 368–377.
[27] M. Benerecetti, F. Mogavero, A. Murano, Reasoning about substructures and games, Trans. Comput. Log. 16 (3) (2015) 25.
[28] D. Berwanger, E. Grädel, Fixed-point logics and solitaire games, Theor. Comput. Sci. 37 (6) (2004) 675–694.
[29] M. Jurdziński, Small progress measures for solving parity games, in: Symposium on Theoretical Aspects of Computer Science'00, in: LNCS, vol. 1770, Springer, 2000, pp. 290–301.
[30] N. Klarlund, D. Kozen, Rabin measures and their applications to fairness and automata theory, in: Logic in Computer Science'91, IEEE Computer Society, 1991, pp. 256–265.
[31] J. Vöge, M. Jurdziński, A discrete strategy improvement algorithm for solving parity games, in: Computer Aided Verification'00, in: LNCS, vol. 1855, Springer, 2000, pp. 202–215.
[32] J. Fearnley, Non-oblivious strategy improvement, in: Logic for Programming Artificial Intelligence and Reasoning'10, in: LNCS, vol. 6355, Springer, 2010, pp. 212–230.
[33] O. Friedmann, A superpolynomial lower bound for strategy iteration based on snare memorization, Discrete Appl. Math. 161 (10–11) (2013) 1317–1337.
[34] W. Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees, Theor. Comput. Sci. 200 (1–2) (1998) 135–183.
[35] R. McNaughton, Infinite games played on finite graphs, Ann. Pure Appl. Log. 65 (1993) 149–184.
[36] M. Jurdziński, M. Paterson, U. Zwick, A deterministic subexponential algorithm for solving parity games, in: Symposium on Discrete Algorithms'06, SIAM, 2006, pp. 117–123.
[37] M. Jurdziński, M. Paterson, U. Zwick, A deterministic subexponential algorithm for solving parity games, SIAM J. Comput. 38 (4) (2008) 1519–1532.
[38] S. Schewe, Solving parity games in big steps, in: Foundations of Software Technology and Theoretical Computer Science'07, in: LNCS, vol. 4855, Springer, 2007, pp. 449–460.
[39] S. Schewe, Solving parity games in big steps, J. Comput. Syst. Sci. 84 (2017) 243–262.
[40] M. Benerecetti, D. Dell'Erba, F. Mogavero, Solving parity games via priority promotion, in: Computer Aided Verification'16, in: LNCS, vol. 9780 (Part II), Springer, 2016, pp. 270–290.
[41] M. Benerecetti, D. Dell'Erba, F. Mogavero, A delayed promotion policy for parity games, in: Games, Automata, Logics, and Formal Verification'16, in: EPTCS, vol. 226, 2016, pp. 30–45.
[42] M. Benerecetti, D. Dell'Erba, F. Mogavero, Improving priority promotion for parity games, in: Haifa Verification Conference 16, in: LNCS, vol. 10028, Springer, 2016, pp. 1–17.
[43] M. Jurdziński, R. Lazic, Succinct progress measures for solving parity games, in: Logic in Computer Science'17, Association for Computing Machinery, 2017, pp. 1–9.

[44] J. Fearnley, S. Jain, S. Schewe, F. Stephan, D. Wojtczak, An ordered approach to solving parity games in quasi polynomial time and quasi linear space, in: SPIN Symposium on Model Checking of Software'17, Association for Computing Machinery, 2017, pp. 112–121.
[45] T. van Dijk, Attracting tangles to solve parity games, in: Computer Aided Verification'18, in: LNCS, vol. 10982, Springer, 2018, pp. 198–215.
[46] O. Friedmann, M. Lange, Solving parity games in practice, in: Automated Technology for Verification and Analysis'09, in: LNCS, vol. 5799, Springer, 2009, pp. 182–196.
[47] O. Friedmann, Recursive algorithm for parity games requires exponential time, RAIRO Theor. Inform. Appl. 45 (4) (2011) 449–457.
[48] K. Lehtinen, A modal mu perspective on solving parity games in quasi-polynomial time, in: Logic in Computer Science'18, Association for Computing Machinery & IEEE Computer Society, 2018, pp. 639–648.
[49] W. Czerwinski, L. Daviaud, N. Fijalkow, M. Jurdzinski, R. Lazic, P. Parys, Universal trees grow inside separating automata: quasi-polynomial lower bounds for parity games, in: Symposium on Discrete Algorithms'18, SIAM, 2018.
[50] M. Bojańczyk, W. Czerwiński, An Automata Toolbox, 2018, unpublished.
[51] P. Parys, Parity Games: Zielonka's Algorithm in Quasi-Polynomial Time, conference under submission.
[52] M. Benerecetti, D. Dell'Erba, F. Mogavero, Solving parity games via priority promotion, Form. Methods Syst. Des. 52 (2) (2018) 193–226.
[53] D. Neuen, P. Schweitzer, An Exponential Lower Bound for Individualization-Refinement Algorithms for Graph Isomorphism, Tech. rep., arXiv, 2017.
[54] D. Neuen, P. Schweitzer, Benchmark Graphs for Practical Graph Isomorphism, Tech. Rep., 2017.
[55] L. Babai, Graph isomorphism in quasipolynomial time [extended abstract], in: Symposium on Theory of Computing'16, Association for Computing Machinery, 2016, pp. 684–697.
[56] K. Apt, E. Grädel, Lectures in Game Theory for Computer Scientists, Cambridge University Press, 2011.
[57] T. van Dijk, Oink: an implementation and evaluation of modern parity game solvers, in: Tools and Algorithms for the Construction and Analysis of Systems'18, in: LNCS, vol. 10805, Springer, 2018, pp. 291–308.
[58] N. Fijalkow, An optimal value iteration algorithm for parity games, arXiv:1801.09618 [abs].
[59] S. Schewe, An optimal strategy improvement algorithm for solving parity and payoff games, in: Computer Science Logic'08, in: LNCS, vol. 5213, Springer, 2008, pp. 369–384.