Solving Parity Games in Big Steps*

Sven Schewe

Universität des Saarlandes, 66123 Saarbrücken, Germany

Abstract. This paper proposes a new algorithm that improves the complexity bound for solving parity games. Our approach combines McNaughton's iterated fixed point algorithm with a preprocessing step, which is called prior to every recursive call. The preprocessing uses ranking functions similar to Jurdziński's, but with a restricted codomain, to determine all winning regions smaller than a predefined parameter. The combination of the preprocessing step with the recursive call guarantees that McNaughton's algorithm proceeds in big steps, whose size is bounded from below by the chosen parameter. Higher parameters result in smaller call trees, but to the cost of an expensive preprocessing step. An optimal parameter balances the cost of the recursive call and the preprocessing step, resulting in an improvement of the known upper bound for solving parity games from approximately $O(m\, n^{\frac{1}{2}c})$ to $O(m\, n^{\frac{1}{3}c})$.

1 Introduction

Parity games have many applications in model checking [1–6] and synthesis [5, 1, 7–10]. In particular, modal and alternating-time μ-calculus model checking [5, 4], synthesis [10, 9] and satisfiability checking [5, 1, 7, 8] for reactive systems, module checking [6], and ATL* model checking [3, 4] can be reduced to solving parity games. This relevance of parity games led to a series of different approaches to solving them [11–25].

The complexity of solving parity games is still an open problem. All current deterministic algorithms have complexity bounds which are (at least) exponential in the number of colors [11, 12, 15–17, 19, 25] ($n^{O(c)}$), or in the squareroot of the number of game positions [13, 24, 25] ($n^{O(\sqrt{n})}$). Practical considerations suggest to assume that the number of colors is small compared to the number of positions. Indeed, all listed applications but μ-calculus model checking are guaranteed to result in parity games where the number of states is exponential in the number of colors. In μ-calculus model checking, the size of the game is determined by the product of the transition system under consideration (which is usually large), and the size of the formula (which is usually small). The number of colors is determined by the alternation depth of the specification, which, in turn, is usually small compared to the specification itself. Algorithms that are exponential only in the number of colors are thus considered the most attractive.

* This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS).

The first representatives of algorithms in the complexity class $n^{O(c)}$ follow the iterated fixed point structure induced by the parity condition [11, 12, 17]. The iterated fixed point construction leads to a time complexity of $O(m\,n^{c-1})$ for parity games with m edges, c colors, and n game positions. The upper complexity bound for solving parity games was first reduced by Browne et al. [16] to $O\big(mn^{\lceil 0.5c\rceil+1}\big)$, and slightly further by Jurdziński [19] to $O\big(c\,m\,(\frac{n}{\lfloor 0.5c\rfloor})^{\lfloor 0.5\,c\rfloor}\big)$.

The weakness of recursive algorithms that follow the iterated fixed point structure [11, 12, 17] is the potentially incremental update achieved by each recursive call. Recently, a big-step approach [24] has been proposed to reduce the complexity of McNaughton's algorithm for games with a high number of colors ($c \in \omega(\sqrt{n})$) to the bound $n^{O(\sqrt{n})}$ known from randomized algorithms [13, 25].

We discuss a different big-step approach that improves the complexity for the relevant lower end of the spectrum of colors, resulting in the complexity $O\big(m\,\big(\frac{\kappa\,n}{c}\big)^{\gamma(c)}\big)$ for solving parity games, where κ is a small constant and $\gamma(c) = \frac{c}{3} + \frac{1}{2} - \frac{1}{3c} - \frac{1}{\lceil \frac{c}{2}\rceil \lfloor \frac{c}{2}\rfloor}$ if c is even, and $\gamma(c) = \frac{c}{3} + \frac{1}{2} - \frac{1}{\lceil \frac{c}{2}\rceil \lfloor \frac{c}{2}\rfloor}$ if c is odd.

To guarantee big update steps, we use an algorithm which is inspired by Jurdziński's [19] approach for solving parity games. His approach is adapted by restricting the codomain of the used ranking function. The resulting algorithm is exploited in a preprocessing step for finding winning regions bounded by the size of a parameter. Compared to [24], this results in a significant cut in the cost for finding small winning regions, since the running time for the preprocessing algorithm is polynomial in the parameter, and exponential only in the number of colors ($O\big(\binom{\pi+\lceil 0.5c\rceil}{\pi}\big)$). Using a parameter of approximately $\sqrt[3]{c\,n^2}$ results in the improved $O\big(m\,\big(\frac{\kappa\,n}{c}\big)^{\gamma(c)}\big)$ complexity bound for solving parity games.

2 Preliminaries

2.1 Parity Games

A parity game $\mathcal{P} = (V_{even}, V_{odd}, E, \alpha)$ consists of a finite directed game graph $\mathcal{D} = (V_{even} \uplus V_{odd}, E)$ without sinks, whose vertices are partitioned into two sets V_{even} and V_{odd}, called the game positions of player *even* and *odd*, respectively, and an evaluation function $\alpha : V_{even} \uplus V_{odd} \to \mathbb{N}$ that maps each game position v to an integer value $\alpha(v)$, called the color of v. For technical reasons we additionally require that the minimal color is 0, and use games with highest color d and games with $c = d + 1$ colors as synonyms. We use $V = V_{even} \uplus V_{odd}$ for the game positions, and extend the common intersection and subtraction operations on digraphs to parity games. ($\mathcal{P} \cap F$ and $\mathcal{P} \smallsetminus F$ thus denote the parity games resulting by restricting the game graph $\mathcal{D}$ of $\mathcal{P}$ to $\mathcal{D} \cap F$ and $\mathcal{D} \smallsetminus F$, respectively.)

Plays. Intuitively, a game is played by placing a pebble on a vertex $v \in V_{even} \uplus V_{odd}$ of $\mathcal{D}$. Whenever the pebble is on a position $v \in V_{even}$, player *even* chooses an edge $e = (v, v') \in E$ originating in v, and moves the pebble to v'. Symmetricly, if the pebble is on a position $v \in V_{odd}$, player *odd* chooses an edge $e = (v, v') \in E$

originating in v, and moves the pebble to v'. In this way, they successively construct an infinite *play* $\pi = v_0 v_1 v_2 v_3 \ldots \in (V_{even} \uplus V_{odd})^\omega$.

A play is evaluated by the highest color that occurs infinitely often. Player *even* (*odd*) wins a play $\pi = v_0 v_1 v_2 v_3 \ldots$ if the highest color occurring infinitely often in the sequence $\alpha(\pi) = \alpha(v_0)\alpha(v_1)\alpha(v_2)\alpha(v_3)\ldots$ is even (odd).

Strategies. Let $\mathcal{D} = (V_{even} \uplus V_{odd}, E)$ be a finite game graph with positions $V = V_{even} \uplus V_{odd}$. A *strategy* for player *even* is a function $f : V^* V_{even} \to V$ which maps each finite history of a play that ends in a position $v \in V_{even}$ to a successor v' of v. (That is, there is an edge $(v, v') \in E$ from v to v'.) A play is *f-conform* if every decision of player *even* in the play is in accordance with f. A strategy is called *memoryless* if it only depends on the current position. A memoryless strategy for *even* can be viewed as a function $f : V_{even} \to V$ such that $(v, f(v)) \in E$ for all $v \in V_{even}$. For a memoryless strategy f of player *even*, we denote with $\mathcal{D}_f = (V_{even} \uplus V_{odd}, E_f)$ the game graph obtained from $\mathcal{D}$ by deleting all transitions from states in V_{even} that are not in accordance with f. (That is, $\mathcal{D}_f$ is a directed graph where all positions owned by player *even* have outdegree 1.) The analogous definitions are made for player *odd*.

A strategy f of player *even* (*odd*) is called *v-winning* if all f-conform plays that start in v are winning for player *even* (*odd*). A position $v \in V$ is v-winning for player *even* (*odd*) if *even* (*odd*) has a v-winning strategy. We call the sets of v-winning positions for player *even* (*odd*) the *winning region* of *even* (*odd*). Parity games are memoryless determined:

Theorem 1. *[11] For every parity game $\mathcal{P}$, the game positions are partitioned into a winning region W_{even} of player* even *and a winning region W_{odd} of player* odd*. Moreover, player* even *and* odd *have memoryless strategies that are v-winning for all positions in their respective winning region.*

Dominions and Attractors. We call a subset $D \subseteq W_\sigma$ of a winning region a *dominion* of player $\sigma \in \{even, odd\}$, if player σ has a memoryless strategy f that is v-winning for all $v \in D$, such that D is not left in any f-conform play ($E_f \cap D \times V \setminus D = \emptyset$). The *$\sigma$-attractor* $A \subseteq V$ of a set $F \subseteq V$ of game positions is the set of those game positions, from which player σ has a memoryless strategy to force the pebble into a position in F. The σ-attractor A of a set F can be defined as the least fixed point of sets that contain F, and that contain a game position v of player σ ($\overline{\sigma}$) if they contain some successor (all successors) of v. (For convenience, we use $\overline{odd}$ and $\overline{even}$ for *even* and *odd*, respectively.)

Constructing this least fixed point is obviously linear in the number of edges of the parity game, and we can fix a memoryless strategy (the attractor strategy) for player σ to reach F in finitely many steps during this construction.

Lemma 1. *For a given parity game $\mathcal{P} = (V_{even}, V_{odd}, E, \alpha)$, and a set F of game positions, we can compute the σ-attractor A of F and a memoryless strategy for σ on $A \setminus F$ to reach F in finitely many steps in time $O(m)$.* □

For a given dominion D for player σ in a parity game $\mathcal{P}$, we can reduce solving $\mathcal{P}$ to computing the σ-attractor A of D, and solving $\mathcal{P} \setminus A$.

Lemma 2. *[24] Let $\mathcal{P}$ be a parity game, D a dominion of player $\sigma \in \{even, odd\}$ for $\mathcal{P}$ with σ-attractor A. Then the winning region (and strategy) of player $\overline{\sigma}$ in $\mathcal{P}$ is her winning region (and strategy) in the subgame $\mathcal{P} \setminus A$. The winning strategy of player σ can be composed by her winning strategy on $\mathcal{P} \setminus A$, her attractor strategy (on $A \setminus D$), and her winning strategy on her dominion (in $\mathcal{P} \cap D$).*

2.2 A Ranking Function Based Approach to Solving Parity Games

So far, Jurdziński's algorithm [19] for solving parity games has been the technique with the best complexity bound. His algorithm draws from the comparably small codomain of the used ranking function (the progress measure).

The method for computing small dominions discussed in Section 3 adopts his techniques by restricting the codomain of the ranking function, sacrificing completeness. Some of the theorems stated in this subsection are thus slightly more general than the theorems in [19], but they are arranged such that the proofs provided in [19] can be applied without changes.

For a parity game $\mathcal{P} = (V_{even}, V_{odd}, E, \alpha)$ with maximal color d, a *σ-progress measure* is, for $\sigma \in \{even, odd\}$, a function $\rho : V_{even} \uplus V_{odd} \to \mathcal{M}^\sigma$ whose codomain $\mathcal{M}^\sigma \subseteq \{f{:}\{0, \ldots, d\} \to \mathbb{N} \mid f(c){=}0$ if c is σ, and $f(c) \leq |\alpha^{-1}(c)|$ otherwise$\} \cup \{\top\}$ contains a maximal element $\top$ and a set of functions from $\{0, \ldots, d\}$ to the integers. The codomain $\mathcal{M}^\sigma$ satisfies the requirements that every σ ($\in \{even, odd\}$) integer $\leq d$ is mapped to 0, while all other integers c are mapped to a value bounded by the number $|\alpha^{-1}(c)|$ of c-colored game positions. (Jurdziński uses the maximal codomain $\mathcal{M}^\sigma_\infty$ defined by replacing containment with equality.) For simplicity, we require downward closedness: if $\mathcal{M}^\sigma$ contains a function $f \in \mathcal{M}^\sigma$, then every function f' which is pointwise smaller than f ($f'(c) \leq f(c) \forall c \leq d$) is also contained in $\mathcal{M}^\sigma$.

For each color $c \leq d$, we define a relation $\rhd_c \subseteq \mathcal{M}^\sigma \times \mathcal{M}^\sigma$. $\rhd_c$ is the smallest relation that contains $\{\top\} \times \mathcal{M}^\sigma$ and a pair of functions $(f, f') \in \rhd_c$ if there is a color $c' \geq c$ such that $f(c') > f'(c')$, and $f(c'') = f'(c'')$ holds true for all colors $c'' > c'$, or if c is σ and $f(c') = f'(c')$ holds true for all $c' \geq c$. That is, $\rhd_c$ is defined by using the lexicographic order, ignoring all colors smaller than c. f needs to be greater than f' by this order, and strictly greater if c is $\overline{\sigma}$. $\rhd_0$ defines an order $\succeq$ on $\mathcal{M}^\sigma$ (the lexicographic order). From this order, we infer the preorder $\sqsupseteq$ on progress measures, which requires that $\succeq$ is satisfied pointwise ($\rho \sqsupseteq \rho' \Leftrightarrow \forall v \in V. \rho(v) \succeq \rho'(v)$). We call a σ progress measure ρ *valid* iff every position $v \in V_\sigma$ has some successor $v' \in V$ with $\rho(v) \rhd_{\alpha(v)} \rho(v')$, and if, for every position $v \in V_{\overline{\sigma}}$ and every successors $v' \in V$ of v, $\rho(v) \rhd_{\alpha(v)} \rho(v')$ holds true.

Let, for a σ progress measure ρ, $\|\rho\| = V \setminus \rho^{-1}(\top)$ denote the game positions that are not mapped to the maximal element $\top$ of $\mathcal{M}^\sigma$. A valid σ progress measure ρ serves as a witness for a winning strategy for player σ on $\|\rho\|$: If we fix a memoryless strategy f for player σ that satisfies $\rho(v) \rhd_{\alpha(v)} \rho(f(v))$ for all $v \in V_\sigma$, then every cycle $v_1 v_2 \ldots v_l = v_1$ with maximal color $c_{max} = \alpha(v_1)$ that is reachable in an f-conform play satisfies $\rho(v_1) \rhd_{\alpha(v_1)} \rho(v_2) \rhd_{\alpha(v_2)} \ldots \rhd_{\alpha(v_{l-1})} \rho(v_l)$. If c_{max} is not σ, this can be relaxed to $\rho(v_1) \rhd_{c_{max}} \rho(v_2) \rhd_{c_{max}-1} \rho(v_3) \rhd_{c_{max}-1} \ldots \rhd_{c_{max}-1} \rho(v_l)$, which is only satisfied if $\rho(v_i) = \top$ holds for all $i = 1, \ldots, l$.

Theorem 2. *[19] Let $\mathcal{P} = (V_{even}, V_{odd}, E, \alpha)$ be a parity game with valid σ progress measure ρ. Then player σ wins on $\|\rho\|$ with any memoryless winning strategy that maps a position $v \in \|\rho\| \cap V_\sigma$ to a position v' with $\rho(v) \rhd_{\alpha(v)} \rho(v')$.*

Such a successor must exist, since the progress measure is valid. The $\sqsupseteq$-*least* valid σ progress measure is well defined and can be computed efficiently.

Theorem 3. *[19] The $\sqsupseteq$-least valid σ progress measure ρ_μ exists and can, for a parity game with m edges and c colors, be computed in time $O(c\,m\,|\mathcal{M}^\sigma|)$.*

When using the maximal codomain $\mathcal{M}^\sigma_\infty$, which contains the function ρ that assigns each $\overline{\sigma}$ value c to $\rho(c) = |\alpha^{-1}(c)|$, for the progress measures, the $\sqsupseteq$-least valid σ progress measure ρ_μ determines the complete winning region of player σ.

Theorem 4. *[19] For a parity game $\mathcal{P} = (V_{even}, V_{odd}, E, \alpha)$, and for the codomain $\mathcal{M}^\sigma_\infty$ for the progress measures, $\|\rho_\mu\|$ coincides with the winning region W_σ of player σ for the $\sqsupseteq$-least valid σ progress measure ρ_μ.*

For parity games with c colors, the size $|\mathcal{M}^\sigma_\infty|$ of the maximal codomain can be estimated by $(\frac{n}{\lfloor 0.5c \rfloor})^{\lfloor 0.5c \rfloor} + 1$ if σ is even, and by $(\frac{n}{\lceil 0.5c \rceil})^{\lceil 0.5c \rceil} + 1$ if σ is odd.

Corollary 1. *[19] Parity games with three colors can be solved and a winning strategy for the player who wins on the highest color constructed in time $O(m\,n)$.*

3 Computing Small Dominions

Computing small dominions efficiently is an essential step in the algorithm introduced in Section 4. In this section, we show that we can efficiently compute a dominion of either player, which is guaranteed to contain all dominions with size bounded by a parameter π. To compute such a dominion, we draw from the efficient computation of the $\sqsupseteq$-least valid σ progress measure (Theorem 3).

Instead of using Jurdziński's codomain $\mathcal{M}^\sigma_\infty$, we use the smaller codomain $\mathcal{M}^\sigma_\pi$ for the progress measures, which contains only those functions f that satisfy $\sum_{c=0}^{d} f(c) \leq \pi$ for some parameter $\pi \in \mathbb{N}$. (d denotes the highest color of the parity game). The size of $\mathcal{M}^\sigma_\pi$ can be estimated by $|\mathcal{M}^\sigma_\pi| \leq \binom{\pi + \lceil 0.5(d+1) \rceil}{\pi} + 1$.

Using $\mathcal{M}^\sigma_\pi$ instead of $\mathcal{M}^\sigma_\infty$, $\|\rho_\mu\|$ contains all dominions of player σ of size $\leq \pi + 1$ (where ρ_μ denotes the $\sqsupseteq$-least valid σ progress measures).

Theorem 5. *Let $\mathcal{P} = (V_{even}, V_{odd}, E, \alpha)$ be a parity game, and let $D \subseteq V$ be a dominion of player $\sigma \in \{even, odd\}$ of size $|D| \leq \pi + 1$. Then there is a valid σ progress measure $\rho : V \to \mathcal{M}^\sigma_\pi$ with $F = \|\rho\|$.*

Proof. Let $\mathcal{P}' = \mathcal{P} \cap D$ be the restriction of $\mathcal{P}$ to D. To solve $\mathcal{P}'$, we can use the maximal codomain ${\mathcal{M}^\sigma_\infty}'$. Since D is a dominion of player σ for $\mathcal{P}$, she has a winning strategy f on the complete subgame $\mathcal{P}'$, and the $\sqsupseteq'$-least progress measure ρ'_μ for this codomain satisfies $\|\rho'_\mu\| = D$ by Theorem 4. Since D has size

$|D| \leq \pi + 1$, it contains at most π postions with $\overline{\sigma}$ color (at least one position needs to have σ color), and thus ρ'_μ is in $\mathcal{M}_\pi^{\sigma\prime}$ (and $\mathcal{M}_\pi^{\sigma\prime} = \mathcal{M}_\infty^{\sigma}{}'$ holds true).

Since D is a dominion of player σ for $\mathcal{P}$, all positions in $V_{\overline{\sigma}} \cap D$ have only successors in D, and we can extend ρ'_μ to a valid σ progress measue ρ for $\mathcal{P}$ by setting $\rho(v) = \rho'_\mu(v)$ for all $v \in D$, and $\rho(v) = \top$ otherwise. ρ is by construction a valid σ progress measure in $\mathcal{M}_\pi^\sigma$ that satisfies $\|\rho\| = D$. □

By Theorem 3, we can compute the $\sqsupseteq$-least valid σ progress measure ρ_μ in time $O(c\,m\,|\mathcal{M}_\pi^\sigma|)$, and by Theorem 2, we can construct a winning strategy for player σ on $\|\rho_\mu\|$ within the same complexity bound.

Corollary 2. *For a given parity game $\mathcal{P}$ with c colors and m edges, we can construct a forced winning region F for player σ that contains all forced winning regions F' of size $|F'| \leq \pi + 1$ in time $O\big(c\,m\,\binom{\pi + \lceil 0.5c \rceil}{\pi}\big)$. A winning strategy for player σ on F can be constructed within the same complexity bound.* □

4 Solving Parity Games in Big Steps

The algorithm proposed in this paper accelerates McNaughton's iterated fixed point approach for solving parity games [11, 12, 17] by using the approximation technique discussed in the previous section to restrict the size of the call tree.

McNaughton's Algorithm. McNaughton's algorithm, as depicted below in Procedure *McNaughton*, takes a parity game $\mathcal{P} = (V_{even}, V_{odd}, E, \alpha)$ as input and returns the ordered pair (W_{even}, W_{odd}) of winning regions for both players.

Procedure *McNaughton*($\mathcal{P}$):

1. set d to the highest color occurring in $\mathcal{P}$
2. if $d = 0$ then return $(V, \emptyset)$
3. set $(\sigma, \overline{\sigma})$ to (*even*,*odd*) if d is even, and to (*odd*,*even*) otherwise
4. set $W_{\overline{\sigma}}$ to $\emptyset$
5. repeat
   (a) set $\mathcal{P}'$ to $\mathcal{P} \setminus \sigma$-*Attractor*$(\alpha^{-1}(d), \mathcal{P})$
   (b) set (W'_{even}, W'_{odd}) to *McNaughton*$(\mathcal{P}')$
   (c) if $W'_{\overline{\sigma}} = \emptyset$ then
      i. set W_σ to $V \setminus W_{\overline{\sigma}}$
      ii. return (W_{even}, W_{odd})
   (d) set $W_{\overline{\sigma}}$ to $W_{\overline{\sigma}} \cup \overline{\sigma}$-*Attractor*$(W'_{\overline{\sigma}}, \mathcal{P})$
   (e) set $\mathcal{P}$ to $\mathcal{P} \setminus \overline{\sigma}$-*Attractor*$(W'_{\overline{\sigma}}, \mathcal{P})$

Evaluating one-color games is trivial, and Procedure *McNaughton* returns the winning regions for this case without further computations (line 2, this case servers as induction basis for the correctness prove).

Procedure *McNaughton* computes in every recursive call (line 5*b*) a *dominion* of player $\overline{\sigma}$ for $\mathcal{P}$: Player $\overline{\sigma}$ has (by induction hypothesis) a winning strategy f for $W_{\overline{\sigma}}$ in $\mathcal{P}'$ and no f-conform strategy starting in the statespace V' of $\mathcal{P}'$ can leave V' in $\mathcal{P}$, since V' is the complement of a σ-attractor (line 5*a*). Solving $\mathcal{P}$

Procedure *Winning-Regions*($\mathcal{P}$):

1. set d to the highest color occurring in $\mathcal{P}$
2. if $d = 0$ then return $(V, \emptyset)$ – one color ⇒ use McNaughton's [11, 12, 17] algorithm
3. set $(\sigma, \overline{\sigma})$ to (*even*,*odd*) if d is even, and to (*odd*,*even*) otherwise
4. set n to the size $|V|$ of $\mathcal{P}$
5. if $d = 2$ then – three colors ⇒ use Jurdziński's [19] algorithm
   (a) set W_{even} to $Approximate(\mathcal{P}, n, even)$ – c.f. Corollary 1
   (b) return $(W_{even}, V \setminus W_{even})$
6. set $W_{\overline{\sigma}}$ to $\emptyset$
7. repeat
   (a) if $d > 2$ then – two colors ⇒ use McNaughton's [11, 12, 17] algorithm
      i. set $W'_{\overline{\sigma}}$ to $\overline{\sigma}$-$Attractor(Approximate(\mathcal{P}, \pi(n, d+1), \overline{\sigma}), \mathcal{P})$ – c.f. Corollary 2
      ii. set $W_{\overline{\sigma}}$ to $W_{\overline{\sigma}} \cup W'_{\overline{\sigma}}$
      iii. set $\mathcal{P}$ to $\mathcal{P} \setminus W'_{\overline{\sigma}}$
   (b) set $\mathcal{P}'$ to $\mathcal{P} \setminus \sigma$-$Attractor(\alpha^{-1}(d), \mathcal{P})$
   (c) set (W'_{even}, W'_{odd}) to *Winning-Regions*($\mathcal{P}'$)
   (d) if $W'_{\overline{\sigma}} = \emptyset$ then
      i. set W_σ to $V \setminus W_{\overline{\sigma}}$
      ii. return (W_{even}, W_{odd})
   (e) set $W_{\overline{\sigma}}$ to $W_{\overline{\sigma}} \cup \overline{\sigma}$-$Attractor(W'_{\overline{\sigma}}, \mathcal{P})$
   (f) set $\mathcal{P}$ to $\mathcal{P} \setminus \overline{\sigma}$-$Attractor(W'_{\overline{\sigma}}, \mathcal{P})$

Fig. 1. Procedure *Winning-Regions*($\mathcal{P}$) returns the ordered pair (W_{even}, W_{odd}) of winning regions for player *even* and *odd*, respectively. V and α denote the game positions and the coloring function of the parity game $\mathcal{P}$. $Approximate(\mathcal{P}, \pi, \sigma)$ computes a dominion for player σ, which contains all dominions of player σ of size less than or equal to $\pi + 1$ (c.f. Corollary 2). σ-$Attractor(F, \mathcal{P})$ computes the respective σ-attractor of a set F of game positions in a game parity $\mathcal{P}$ (c.f. Lemma 1).

can thus be reduced to constructing the $\overline{\sigma}$-attractor $A_{\overline{\sigma}}$ of $W_{\overline{\sigma}}$ (line 5d), and solving $P \setminus A_{\overline{\sigma}}$ (line 5e).

If the recursive call (line 5b) provides the result that player σ wins from every position in $\mathcal{P}'$, she wins from every position in $\mathcal{P}$ (following her winning strategy for $\mathcal{P}'$ in V' and an attractor strategy to d-colored positions (line 5a) otherwise), and Procedure *McNaughton* terminates (lines 5c – 5cii).

Proceeding in Big Steps. As observed by Jurdziński, Paterson and Zwick [24], McNaughton's algorithm can be adopted by computing any dominion of player $\overline{\sigma}$ (instead of the particular dominion returned by the recursive call). In [24], this observation is exploited by performing a brute-force search for dominions of size $\sqrt{n}$ (where $n = |\mathcal{P}|$ denotes the number of game positions), and performing a recursive call only if no such dominion exists. The cost for each brute-force search is $n^{\sqrt{n}}$, which coincides with the upper bound on the size of the call tree, improving the complexity bound for the theoretical case of parity games with a high number of colors – $c \in \omega(\sqrt{n})$ – to $O(n^{\sqrt{n}})$.

Brute-force search, however, is too expensive, and does not improve the complexity bound for the common case that the number of colors is small. We therefore propose to use the efficient approximation technique introduced in Section 3 instead. As a further difference, we propose to perform a recursive call after each approximation step, resulting in the guarantee that the progress (that is, the set of evaluated positions) in each iteration step exceeds the size defined by the chosen parameter. The resulting algorithm is depicted in Figure 1.

The set $W'_{\overline{\sigma}}$ computed in line $7ai$ is the $\overline{\sigma}$-attractor of the dominion of player $\overline{\sigma}$ in $\mathcal{P}$ computed by the approximation procedure (c.f. Corollary 2) introduced in Section 3, and thus itself a dominion of player $\overline{\sigma}$. The set $W''_{\overline{\sigma}}$ computed in the recursive call (line $7c$) is a dominion of player $\overline{\sigma}$ in $\mathcal{P} \setminus W'_{\overline{\sigma}}$, and thus $D = W'_{\overline{\sigma}} \cup W''_{\overline{\sigma}}$ is a dominion in $\mathcal{P}$. If the size of D does not exceed the chosen parameter by at least two, D must be contained in the dominion computed in $\mathit{Approximate}(\mathcal{P}, \pi(n, d+1), \overline{\sigma})$, and $W''_{\overline{\sigma}}$ is empty. In this case, the procedure terminates (line $7d$), otherwise, we obtain a progress of at least $\pi(n, d+1) + 2$.

While bigger parameters slow down the approximation procedure (c.f. Corollary 2), they thus restrict the size of the call tree. The best results are obtained if the parameter is chosen such that the cost of calling the approximation procedure (line $7ai$) and the cost of the recursive call (line $7c$) are approximately equivalent. If c is of reasonable size (that is, in $O(\sqrt{n})$), this is the case if we set the parameter approximately to $\sqrt[3]{c\,n^2}$. (The function β defined below for the proof of the complexity quickly converges to $\frac{2}{3}$.)

Starting point for the complexity estimation is the case of three colors, where we use Jurdziński's algorithm [19] (Corollary 1). (Skipping lines $5 - 5b$ moves the induction basis further down, resulting in the complexity of $O(m\,n^{1.5})$ for the case of three colors. The optimization obtained by using [19] for three-color games accounts for the $-\frac{1}{\lceil 0.5c \rceil \lfloor 0.5c \rfloor}$ part of the function γ introduced below.)

For fixed numbers of colors, the resulting complexities evolve as follows:

number of colors	3	4	5	6	7	8	$\cdots$
approximation complexity	-	$O(m\,n)$	$O(m\,n^{1\frac{1}{2}})$	$O(m\,n^2)$	$O(m\,n^{2\frac{1}{3}})$	$O(m\,n^{2\frac{3}{4}})$	$\cdots$
chosen parameter $\pi_c(n)$	-	$\sqrt{n}$	$\sqrt{n}$	$\sqrt[3]{n^2}$	$\sqrt[12]{n^7}$	$\sqrt[16]{n^{11}}$	$\cdots$
number of iterations $\frac{n}{\pi_c(n)}$	-	$\sqrt{n}$	$\sqrt{n}$	$\sqrt[3]{n}$	$\sqrt[12]{n^5}$	$\sqrt[16]{n^5}$	$\cdots$
solving complexity	$O(m\,n)$	$O(m\,n^{1\frac{1}{2}})$	$O(m\,n^2)$	$O(m\,n^{2\frac{1}{3}})$	$O(m\,n^{2\frac{3}{4}})$	$O(m\,n^{3\frac{1}{16}})$	$\cdots$

The approximation complexity for $c+1$ colors is chosen to coincide with the complexity of solving a game with c colors. (Its complexity thus coincides with the complexity of each iteration of the repeat loop). The parameter $\pi_c(n)$ is chosen to result in this complexity, and the number of iterations is $i_c(n) = \frac{n}{\pi_c(n)}$, results from this choice. Finally, the resulting complexity for solving games with $c+1$ colors is $i_c(n)$ times the complexity for solving parity games with c colors.

Correctness. In this paragraph, we demonstrate that Procedure *Winning-Regions* computes the winning regions correctly.

Theorem 6. *For a given parity game $\mathcal{P}$, Procedure* Winning-Regions *computes the complete winning regions of both players.*

Proof. We prove the claim by induction. Let d denote the highest color of $\mathcal{P}$.
Induction Basis ($d = 0$, $d = 2$): For $d = 0$, the highest color on every path is obviously 0, and every strategy for player *even* is winning. For $d = 2$, the algorithm follows Jurdziński's [19] algorithm (c.f. Theorem 4 and Corollary 1).
Induction Step ($d \mapsto d + 1$): Let $\mathcal{P}$ be a parity game with highest color $d + 1$.

The call of the Procedure *Approximate* in line 7*ai* provides a (possibly empty) dominion D for player $\overline{\sigma}$ (Theorem 5). The $\overline{\sigma}$-attractor of this set is then added to the winning region of $\overline{\sigma}$ (line 7*aii*), and subtracted from $\mathcal{P}$, which is safe by Lemma 2.

In line 7*b*, the σ-attractor A of the set of states with color $d+1$ is subtracted from $\mathcal{P}$, and the resulting parity game $\mathcal{P}' = \mathcal{P} \setminus A$ is solved by recursively calling the Procedure *Winning-Regions* (line 7*c*). Since the highest color of $\mathcal{P}'$ is $\leq d$, the resulting winning regions are correct by induction hypothesis. $W''_{\overline{\sigma}}$ is a dominion of player $\overline{\sigma}$ in $\mathcal{P}'$, and, due to the σ-attractor construction, also in $\mathcal{P}$. If $W''_{\overline{\sigma}}$ is non-empty, then the $\overline{\sigma}$-attractor of this set is added to the winning region of $\overline{\sigma}$ (line 7*e*), and subtracted from $\mathcal{P}$ (line 7*f*), which is safe by Lemma 2.

Since the size of $\mathcal{P}$ is strictly reduced in every iteration of the loop, the set $W''_{\overline{\sigma}}$ returned after the recursive call in line 7*c* is eventually empty, and the procedure terminates. When $W''_{\overline{\sigma}}$ is empty, player σ wins from all positions in (the remaining) parity game $\mathcal{P}$ by following a memoryless strategy that agrees on every position in $\mathcal{P}'$ with a memoryless winning strategy f on $\mathcal{P}'$, makes an arbitrary (but fixed) choice for positions with color $d+1$, and follows an attractor strategy (from the σ-attractor construction of line 7*b*) on the remaining positions. An f-conform play either eventually stays in $\mathcal{P}'$, in which case it is winning for player σ by induction hypothesis, or always eventually visits a position with color $d + 1$, in which case $d + 1$ is the highest color that occurs infinitely many times. Since $d + 1$ is σ, player σ wins in this case, too. □

Complexity. While the correctness of the algorithm is independent of the chosen parameter, its complexity crucially depends on this choice. We will choose the parameter such that the complexity for the recursive call (line 7*c*) coincides with the complexity of computing the approximation (line 7*ai*).

First, we show that the Procedure *Winning-Regions* proceeds in big steps.

Lemma 3. *For every parameter $\pi(n, c)$, the main loop of the algorithm is iterated at most $\left\lfloor \frac{n}{\pi(n,c)+2} \right\rfloor + 1$ times.*

Proof. The $\overline{\sigma}$-attractor $W'_{\overline{\sigma}}$ of the computed approximation D (line 7*ai*) and the winning region $W''_{\overline{\sigma}}$ of $\overline{\sigma}$ are dominions for $\overline{\sigma}$ on $\mathcal{P}$ and $\mathcal{P} \setminus W'_{\overline{\sigma}}$, respectively. Thus, their union $U = W'_{\overline{\sigma}} \cup W''_{\overline{\sigma}}$ is a dominion on $\mathcal{P}$. If the size of U does not exceed $\pi + 1$, than U is contained in D by Corollary 2. In this case, $W''_{\overline{\sigma}}$ is empty, and the loop terminates. Otherwise, a superset of U is subtracted from P during the iteration (line 7*aiii* and 7*f*), which can happen at most $\left\lfloor \frac{n}{\pi(n,c)+2} \right\rfloor$ times. □

Building on this lemma, it is simple to define the parameter π such that the requirement of equal complexities is satisfied: We fix the function γ such that

$\gamma(c)=\frac{c}{3}+\frac{1}{2}-\frac{1}{\lceil 0.5c\rceil\lfloor 0.5c\rfloor}$ if c is odd, and $\gamma(c)=\frac{c}{3}+\frac{1}{2}-\frac{1}{3c}-\frac{1}{\lceil 0.5c\rceil\lfloor 0.5c\rfloor}$ if c is even, . and $\beta(c)=\frac{\gamma(c-1)}{\lceil 0.5c\rceil}$. Finally, we choose $\pi(n,c)$ to be the smallest natural number that satisfies $\frac{n}{\pi(n,c)+2} < \frac{n^{1-\beta(c)}}{2\sqrt[3]{c}} - 1$ ($\pi(n,c) \approx 2\sqrt[3]{c}n^{\beta(c)}$).

Theorem 7. *Solving a parity game $\mathcal{P}$ with $c > 2$ colors, m edges, and n game positions can be performed in time $O\big(m\left(\frac{\kappa n}{c}\right)^{\gamma(c)}\big)$. ($\kappa$ is a small constant.)*

Proof. First we estimate the running time of the procedure without the recursive calls. To estimate the running time of the approximation algorithm $(\pi(n,c)+\lceil 0.5c\rceil)^{\lceil 0.5c\rceil}$ can be estimated by $\kappa_1(\kappa_2\pi(n,c))^{\lceil 0.5c\rceil}$, and the running time of each iteration step (plus the part before the loop (lines $1-6$) and minus the recursive call) can be estimated by $\frac{\kappa_3 m}{\sqrt[3]{(c-1)!}}(\kappa_4 n)^{\gamma(c-1)}$. ($\kappa_1$, κ_2, κ_3 and κ_4 are suitable constants.) We show by induction that the overall running time of the procedure can be estimated by $\frac{\kappa_3 m}{\sqrt[3]{c!}}(\kappa_4 n)^{\gamma(c)}$.
Induction Basis ($c \le 3$): For parity games with one or two and with three colors, we use the algorithms of McNaughton and Jurdziński, respectively, resulting in the complexities $O(n)$, $O(m\,n)$ and $O(m\,n) = O(m\,n^{\gamma(3)})$, respectively.
Induction Step ($c \mapsto c+1$): By induction hypothesis, the cost of every recursive call can (as well as the remaining cost of each iteration step) be estimated by $\frac{\kappa_3 m}{\sqrt[3]{(c-1)!}}(\kappa_4 n)^{\gamma(c-1)}$. Since Lemma 3 implies that the loop is iterated at most $\lfloor \frac{n^{1-\beta(c)}}{2\sqrt[3]{c}} \rfloor$ times, the claim follows immediately ($\gamma(c) = \gamma(c-1) + 1 - \beta(c)$). □

If we impose the restriction that c is not linear in $\sqrt{n}$, that is, if we assume that $c \in o(\sqrt{n})$, this coarse estimation already suffices to show that we can choose any value higher than 1, $2\sqrt{2e}$, and $(2e)^{1.5}$ for κ_2, κ_4, and κ, respectively.

Strategies. If we want to construct the winning strategies of one or both players, the complexity is left unchanged in most cases. The only exception is the construction of winning strategies for player *odd* in three-color games.

Theorem 8. *The algorithm can be extended to compute the winning strategies for both players. The winning strategy for player* odd *on her complete winning region in s parity game with three colors can be constructed in time $O(m\,n^{1.5})$. In all other cases, constructing the winning strategies does not increase the complexity of the algorithm.*

Proof. Extending the procedure to return winning strategies for both players on their respective winning regions only comprises fixing an arbitrary strategy for player *odd* in the trivial case of single-color games (line 2), computing winning strategies for both players for three-color games (line $5a$), computing winning strategies for player $\overline{\sigma}$ in the approximation procedure in line $7ai$, computing the attractor strategies in lines $7ai$, $7b$, and $7e$, and fixing arbitrary strategies for d-colored positions prior to returning the winning regions in line $7aiii$. By the Corollaries 1 and 2, and by Lemma 1, all these extension with the exception

of constructing the winning strategy of player *odd* for games with three colors (line 5a) can be made without changing the complexity.

Computing the winning strategy of player *odd* immediately would increase the complexity of the algorithm. For these three-color games, we therefore *postpone* computing the strategies of player *odd* till after solving the complete game by pushing the respective three-color game (or rather its intersection with the winning region of player *odd*) on a solve-me-later stack. While postponing the construction of the strategies for player *odd* in these subgames, we compute a partial strategy for player *odd* that can be completed to a winning strategy on her complete winning region by filling in winning strategies for these subgames.

Completing the strategies *after* solving the complete game is cheaper, because solving most of the three-color games becomes obsolete: If the recursive call (line 7c) returns a non-empty set $W''_{\overline{\sigma}}$, then the set W''_{σ} is discarded, and it is safe to delete all those games from the top of the solve-me-later stack that refer to W''_{σ}.

As a result, we only need to solve the subgames remaining on the stack after the parity game $\mathcal{P}$ has been solved to complete the winning strategies. Since the sum of the sizes of these games is bounded by the size of the complete game $\mathcal{P}$, this step can be performed in time $O(m\,n^{1.5})$ (using the just established complexity bound for solving games with four colors) if $\mathcal{P}$ has n game positions and m edges, independent of the number of colors of $\mathcal{P}$. □

5 Conclusions

We proposed a novel approach to solving parity games, which reduces the complexity bound for solving parity games from $O\big(c\,m\,(\frac{n}{\lceil 0.5c\rceil})^{\lfloor 0.5\,c\rfloor}\big)$ [19] to $O\big(m\,\big(\frac{\kappa\,n}{c}\big)^{\gamma(c)}\big)$ for $\gamma(c) = \frac{c}{3}+\frac{1}{2}-\frac{1}{3c}-\frac{1}{\lceil\frac{c}{2}\rceil\lfloor\frac{c}{2}\rfloor}$ if c is even, and $\gamma(c) = \frac{c}{3}+\frac{1}{2}-\frac{1}{\lceil\frac{c}{2}\rceil\lfloor\frac{c}{2}\rfloor}$ if c is odd. (κ is a small constant that can be fixed to approximately $(2e)^{1.5}$).

This reduces the exponential factor from $\lfloor\frac{c}{2}\rfloor$ to less than $\frac{c}{3}+\frac{1}{2}$. It is, after the reduction from $c-1$ [11, 12, 17] to $\lceil\frac{c}{2}\rceil+1$ by Browne et al. [16], the second improvement that reduces the exponential growth with the number of colors.

Besides solving parity games, we are often interested in winning strategies for the players, since they serve as witnesses and counter examples in model checking, and as models in synthesis. When constructing these strategies, the improvement in the complexity of the discussed approach is even higher. Constructing winning strategies for both players increase the complexity of the proposed algorithm only for parity games with three colors, where the complexity increases slightly from $O(m\,n)$ to $O(m\,n^{1.5})$. The best previously known bound for constructing winning strategies [19] has been $O\big(c\,m\,(\frac{n}{\lceil 0.5c\rceil})^{\lceil 0.5\,c\rceil}\big)$.

The suggested approach thus provides a significantly improved complexity bound for solving parity games with more than 2, and up to $o(\sqrt{n})$ colors.

References

1. Kozen, D.: Results on the propositional μ-calculus. Theor. Comput. Sci. **27** (1983) 333–354

2. Emerson, E.A., Jutla, C.S., Sistla, A.P.: On model-checking for fragments of μ-calculus. In: CAV. (1993) 385–396
3. de Alfaro, L., Henzinger, T.A., Majumdar, R.: From verification to control: Dynamic programs for omega-regular objectives. In: Proc. LICS, IEEE Computer Society Press (2001) 279–290
4. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM **49**(5) (2002) 672–713
5. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bull. Soc. Math. Belg. **8**(2) (2001)
6. Kupferman, O., Vardi, M.: Module checking revisited. In: Proc. CAV. Volume 1254 of Lecture Notes in Computer Science., Springer-Verlag (1997) 36–47
7. Vardi, M.Y.: Reasoning about the past with two-way automata. In: Proc. ICALP, Springer-Verlag (1998) 628–641
8. Schewe, S., Finkbeiner, B.: The alternating-time μ-calculus and automata over concurrent game structures. In: Proc. CSL, Springer-Verlag (2006) 591–605
9. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. In: Proc. LICS, IEEE Computer Society (2006) 255–264
10. Schewe, S., Finkbeiner, B.: Synthesis of asynchronous systems. In: Proc. LOPSTR, Springer-Verlag (2006) 127–142
11. McNaughton, R.: Infinite games played on finite graphs. Ann. Pure Appl. Logic **65**(2) (1993) 149–184
12. Emerson, E.A., Lei, C.: Efcient model checking in fragments of the propositional μ-calculus. In: Proc. LICS, IEEE Computer Society Press (1986) 267–278
13. Ludwig, W.: A subexponential randomized algorithm for the simple stochastic game problem. Inf. Comput. **117**(1) (1995) 151–155
14. Puri, A.: Theory of hybrid systems and discrete event systems. PhD thesis, Computer Science Department, University of California, Berkeley (1995)
15. Zwick, U., Paterson, M.S.: The complexity of mean payoff games on graphs. Theoretical Computer Science **158**(1–2) (1996) 343–359
16. Browne, A., Clarke, E.M., Jha, S., Long, D.E., Marrero, W.: An improved algorithm for the evaluation of fixpoint expressions. Theoretical Computer Science **178**(1–2) (1997) 237–255
17. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci. **200**(1-2) (1998) 135–183
18. Jurdziński, M.: Deciding the winner in parity games is in UP $\cap$ co-UP. Information Processing Letters **68**(3) (1998) 119–124
19. Jurdziński, M.: Small progress measures for solving parity games. In: Proc. STACS, Springer-Verlag (2000) 290–301
20. Vöge, J., Jurdziński, M.: A discrete strategy improvement algorithm for solving parity games. In: Proc. CAV, Springer-Verlag (2000) 202–215
21. Obdržálek, J.: Fast mu-calculus model checking when tree-width is bounded. In: Proc. CAV. (2003) 80–92
22. Lange, M.: Solving parity games by a reduction to SAT. In Majumdar, R., Jurdziński, M., eds.: Proc. Int. Workshop on Games in Design and Verification. (2005)
23. Berwanger, D., Dawar, A., Hunter, P., Kreutzer, S.: Dag-width and parity games. In: Proc. STACS, Springer-Verlag (2006) 524–436
24. Jurdziński, M., Paterson, M., Zwick, U.: A deterministic subexponential algorithm for solving parity games. In: Proc. SODA, ACM/SIAM (2006) 117–123
25. Björklund, H., Vorobyov, S.: A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. Discrete Appl. Math. **155**(2) (2007) 210–229