Attractors Is All You Need: Parity Games In Polynomial Time

Rick van der Heijden
Eindhoven University of Technology

Abstract

This paper provides a polynomial-time algorithm for solving parity games that runs in $\mathcal{O}\left(n^2 \cdot (n+m)\right)$ time—ending a search that has taken decades. Unlike previous attractor-based algorithms, the presented algorithm only removes regions with a determined winner. The paper introduces a new type of attractor that can guarantee finding the minimal dominion of a parity game. The attractor runs in polynomial time and can peel the graph empty.

1 Introduction

A parity game is an infinite-duration two-player game on a finite graph. The two players, even ($\Diamond$) and odd ($\Box$), are pushing a token over the edges of the graph. Each node of the graph is associated with a natural number *priority*. A *play* is the infinite sequence of nodes visited by the token. The winner of a play is determined by the lowest priority that is encountered infinitely often. If the parity of the priority is even, then even wins; otherwise, odd wins.

The nodes determine the player that can push the token next; this is the *owner* of the node. Moreover, every node must have at least one outgoing edge. A *winning strategy* for a player is a strategy that ensures all plays are won from the starting vertex regardless of the other player's moves. A *memoryless strategy* depends only on the current node, and not the history of the play. An essential result in the field of parity games is of *positional determinacy* [BSV04, EJ91, McN93, Zie98], i.e., the winner always has a memoryless winning strategy. Figure 1 illustrates a parity game.

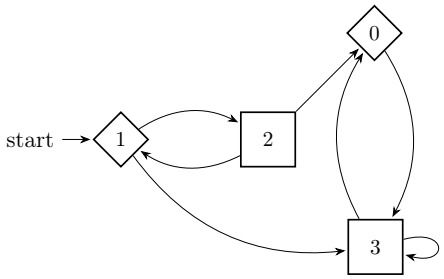


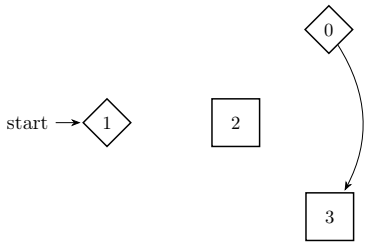
Figure 1: An example of a parity game.

In the example of Figure 1, the nodes owned by even are $\Diamond$-shaped and those owned by odd are $\Box$-shaped. Odd has a memoryless strategy which is to push from 2 to 0 (see Figure 2c) and keep the token at 3 (see Figure 2d). Even can only move it to either 3 from 0 (see Figure 2a) or to either 2 or 3 from 1 (see Figure 2b), in neither case can even avoid odd from winning.

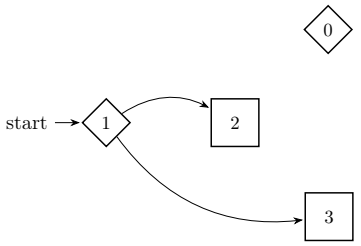
Parity games have various practical applications and are interesting from a complexity theoretic viewpoint. They are related to various fields such as modal μ-calculus, tree automata, Muller games, and synthesis [BL18, DJL19, KV98, PW23, Rab75, CJK$^+$17]. Parity games have been well studied mainly because of their relation to problems in formal verification and synthesis. They are polynomially equivalent to the model checking problem in modal μ-calculus, e.g., used in model checkers, and it is also polynomial equivalent to the emptiness problem for nondeterministic automata on infinite trees with parity acceptance condition [EJS01, KV98, Maz02]. Furthermore, parity games influenced works on ω-word automata translation [BL18, DJL19], linear optimisation [Fri11, FHZ11], and Markov decision process [Fea10].

Whether parity games admit a polynomial time algorithm has been asked by many researchers, including Emerson and Jutla in 1991 [EJ91]. Parity games are shown to be in NP $\cap$ co-NP [EJS93] and also UP $\cap$ co-UP [Jur98]. This made the problem rare complexity-wise, together with other well-known problems such as *mean-payoff games* and *integer factorisation*. The first algorithms for this problem were exponential [McN93, Zie98, Jur00]. The earliest subexponential algorithms were developed at the start of this century [PV01, JPZ06]. Lastly, in 2017, the first quasipolynomial algorithm [CJK$^+$17] was invented which resulted in a wave of quasipolynomial algorithms [DJT20, FJdK$^+$19, JL17, Leh18, LPSW22, Par20, Par19].

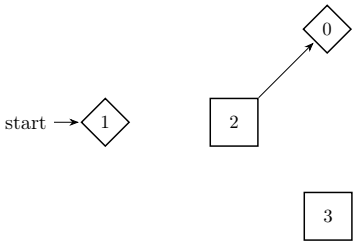
This paper answers the decades old question of whether parity games admit a polynomial time



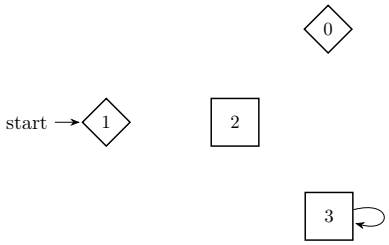
(a) All actions for node 0.



(b) All actions for node 1.



(c) Strategy of □ for node 2.



(d) Strategy for □ for node 3.

Figure 2: An overview of the memoryless strategy of □ for the parity game of Figure 1.

algorithm. It does so by presenting a polynomial time algorithm based on a novel approach that utilises attractors (regions of the graph where one player can force the other), ensuring that removed regions have a determined winner. First, Section 2 introduces the necessary preliminaries. Afterwards, Section 3 introduces the algorithm and proves its correctness and time bounds. Lastly, Section 4 concludes the paper.

2 Preliminaries

A parity game $G = (V, E, \rho, (V_\Diamond, V_\Box))$ is a finite directed graph. The function $\rho : V \to \mathbb{N}$ assigns priorities to the vertices. The sets $(V_\Diamond, V_\Box)$ is a partition of the vertices, i.e., $V = V_\Diamond \cup V_\Box$, and $V_\Diamond$ are the vertices owned by $\Diamond$ and $V_\Box$ by $\Box$. The edges E are a left-total relation, i.e., each vertex has at least one outgoing edge. The successors of a vertex u are denoted by $E(u)$. Lastly, to reason in general, $\bigcirc \in \{\Diamond, \Box\}$ denotes either player and $\overline{\bigcirc} = \{\Diamond, \Box\} \setminus \bigcirc$ denotes the other.

A play $\pi = v_1 v_2 v_3 \dots$ is an infinite sequence of vertices that adheres to E, i.e., $(v_i, v_{i+1}) \in E$ for all i. Let $\inf(\pi)$ be the set of priorities of π. Then, player even wins the play π iff $\min\{\inf(\pi)\}$ is even, otherwise odd wins. A strategy for player $\bigcirc$ is a partial function $\varrho_\bigcirc : V_\bigcirc \rightharpoonup V$ that assigns the successor for each vertex. A strategy $\varrho_\bigcirc$ is consistent with a play $\pi = v1, \dots, v_i, v_{i+1}, \dots$ iff $\varrho_\bigcirc(v_i) = v_{i+1}$. Moreover, $\text{Play}(v, \varrho_\bigcirc)$ is the set of all plays starting in v consistent with $\varrho_\bigcirc$. Furthermore, a strategy $\varrho_\bigcirc$ is winning from v iff every play in $\text{Play}(v, \varrho_\bigcirc)$ is winning for $\bigcirc$. Lastly, a strategy $\varrho_\bigcirc$ is closed on $W \subseteq V$, if for all $v \in W \cap V_\bigcirc$ implies $\varrho_\bigcirc(v) \in W$ and for all $v \in W \cap V_{\overline{\bigcirc}}$ implies $E(v) \subseteq W$.

A set $W \subseteq V$ is $\bigcirc$-closed iff there exists a strategy $\varrho_\bigcirc$ closed on W. This set W is also a $\overline{\bigcirc}$-trap. A set $D \subseteq V$ is a dominion for player $\bigcirc$ if D is a $\bigcirc$-closed set and $\bigcirc$ wins all vertices in D. Moreover, dominions are closed under union. Furthermore, note that there exists a minimal dominion in each non-empty parity game.

An arena restriction restricts the parity game $G = (V, E, \rho, (V_\Diamond, V_\Box))$ to a subgame $G \setminus U = \big(V', E', \rho', \big(V'_\Diamond, V'_\Box\big)\big)$ for a set $U \subseteq V$. The sets are defined as $V' = V \setminus U$, $E' = E \cap (V' \times V')$, $V'_\Diamond = V_\Diamond \setminus U$, and $V'_\Box = V_\Box \setminus U$. Lastly, the function ρ' is $\rho'(v) = \rho(v)$ for all $v \in V'$.

The attractor set of $U \subseteq V$ for a player $\bigcirc$, denoted as $\bigcirc$-$Attr(G, U)$, is the set of vertices that player $\bigcirc$ can force into U regardless of the play. It is defined as $\bigcirc$-$Attr(G, U) = \bigcup_{k\in\mathbb{N}} \bigcirc$-$Attr^k(G, U)$, which is defined inductively as:

$$\begin{aligned}
\bigcirc\text{-}Attr^0(G, U) &= U \\
\bigcirc\text{-}Attr^{k+1}(G, U) &= \bigcirc\text{-}Attr^k(G, U) \\
&\quad \cup \Big\{ v \in V_\bigcirc \mid E(v) \cap \bigcirc\text{-}Attr^k(G, U) \neq \emptyset \Big\} \\
&\quad \cup \Big\{ v \in V_{\overline{\bigcirc}} \mid E(v) \subseteq \bigcirc\text{-}Attr^k(G, U) \Big\}
\end{aligned}$$

Note that the *Attr* function is monotone, i.e., for two sets $U \subseteq U'$ $\bigcirc$-$Attr(G, U) \subseteq \bigcirc$-$Attr(G, U')$. Also, note that the *Attr* function is idempotent, i.e., for set U it holds that $\bigcirc$-$Attr(G, \bigcirc\text{-}Attr(G, U)) = \bigcirc$-$Attr(G, U)$. Lastly, the following lemma shows the relation between an attractor and a trap.

Lemma 1 ([Zie98]). *$V \setminus \bigcirc$-$Attr(G, U)$ is a $\bigcirc$-trap*

3 The Algorithm

This section introduces the new algorithm. First, $A(G, d)$ is introduced with its subfunctions. This is a new type of attractor function that is guaranteed to contain a dominion from the parity game. Afterwards, the algorithm based on $A(G, d)$ is presented together with its proofs. Lastly, Section 3.1 analyses the complexity of the algorithm.

First, the algorithm assumes there are no self-cycles, even though they are allowed in parity games. However, it is possible to remove all self-cycles as a pre-routine to solve general parity games correctly [FL09].

The idea behind the function $A(G, d)$ is to find dominions in the parity game. Specifically, given a priority d and a parity game G, $A(G, d)$ starts with all vertices of priorities equal or less than d and of the same parity as d ($U_d(G)$). It finds those vertices that are attracted to vertices of $U_d(G)$ that have a lower priority ($\bigcirc$-$A^*_d(G)$). Lastly, $A(G, d)$ peels off vertices from U that cannot force a return to U or that can return only through cycles with a lower priority and different parity ($A(G, d)$, $U(G, d)$, $\bigcirc$-$A'_d(G, A)$ computation).

To specify this intuition, let first the function $par(d) = d \bmod 2$. Proceeding further, given a parity game G and a priority d, the function $U_d(G)$ and $\bigcirc$-$A^*_d(G)$ are formally defined as follows:

$$\begin{aligned}
U_d(G) &= \bigcup_{0\leq k\leq d, par(k)=par(d)} \{v \in V \mid \rho(v) = k\} \\
Astar &= \bigcup_{0\leq k\leq d, par(k)\neq par(d)} \{v \in \bigcirc\text{-}Attr(G, U_{k-1}(G)) \mid \rho(v) = k\}
\end{aligned}$$

To peel off vertices, $A(G, d)$ employs $\bigcirc$-$A'_d(G, A)$ to determine which vertices attracted to $U_d(G)$ can avoid vertices of lower priority. That is, this set determines which nodes $\bigcirc$ cannot force

to be won by itself. Formally, given a parity game G, attractor set A, and priority d, the function $\bigcirc\text{-}A'_d(G, A)$ is formally defined as follows:

$$\bigcirc\text{-}A'_d(G, A) = \overline{\bigcirc}\text{-}Attr(G, \{v \in (A \setminus (\bigcirc\text{-}A^*_d(G) \cup U_d(G))) \mid \rho(v) < d\})$$

At last, $A(G, d)$ is defined as $A(G, d) = \bigcap_{k \in \mathbb{N}} A^k(G, d)$, where A^k and underlying functions are defined as:

$$\begin{aligned}
U^0(G, d) &= U_d(G) \\
A^0(G, d) &= \bigcirc\text{-}Attr\left(G, U^0(G, d)\right) \\
U^{k+1}(G, d) &= U^k(G, d) \setminus \overline{\bigcirc}\text{-}Attr\left(G, \left(V \setminus A^k(G, d)\right) \cup \bigcirc\text{-}A'_d\left(G, A^k(G, d)\right)\right) \\
A^{k+1}(G, d) &= \bigcirc\text{-}Attr\left(G, U^{k+1}(G, d)\right)
\end{aligned}$$

First, it is crucial to prove that $A(G, d)$ is a dominion with the following theorem.

Theorem 2. *For a given parity game G and priority d of player $\bigcirc$, $A(G, d)$ is a $\bigcirc$-dominion.*

Proof. Let k be such that $A(G, d) = A^k(G, d)$. It suffices to show that all vertices in $U^k(G, d)$ either stay in $U^k(G, d)$ or can reach some vertex $u \in U^k(G, d)$ via $A(G, d)$ such that $p(u) < p(v)$ for each v we must visit in $A(G, d)$. First, it is proved that u visits either $U^k(G, d)$ or $A(G, d)$.

Let $u \in U^k(G, d)$, then if u is owned by $\bigcirc$ and it has some neighbour in $U^k(G, d)$, then it holds, similarly for when u is owned by $\overline{\bigcirc}$ with all neighbours.

Assume the previous does not hold. If u is owned by $\bigcirc$, then it must have at least one neighbour in $A^k(G, d) \setminus \bigcirc\text{-}A'_d\left(G, A^k(G, d)\right)$; otherwise $u \notin U^k(G, d)$. Similarly, if u is owned by $\overline{\bigcirc}$ for all neighbours not in $U^k(G, d)$. This concludes that u either visits $U^k(G, d)$ or $A(G, d)$.

Let $v \in A^k(G, d) \setminus U^k(G, d)$ be the vertex with the lowest priority that we must visit from u to $U^k(G, d)$. If v is owned by $\bigcirc$ and there is no play ending at some $u' \in U^k(G, d)$ with $\rho(v) > \rho(u')$, then clearly $v \in \bigcirc\text{-}A'_d\left(G, A^k(G, d)\right)$, similarly for if u is owned by $\overline{\bigcirc}$ and for all choices. However, as v must be visited, it follows that $u \notin U^k(G, d)$. Hence, such v does not exist. □

Remark. Note that this proof indirectly uses the fact that there are no self-loops in the game. We assume that the token is always pushed to another vertex.

Furthermore, we demonstrate that $A(G, d)$ can be guaranteed to include the minimal dominion.

Lemma 3. *Given a parity game G and a dominion D such that D is minimal in G and winning for player $\bigcirc$. Let $\tilde{d}$ be the maximum priority of parity $par(\tilde{d}) \equiv \bigcirc$. Then, $D \subseteq A\left(G, \tilde{d}\right)$.*

Proof. Let $\breve{d}$ be the maximum priority of parity $par(\breve{d}) = \overline{\bigcirc}$. Let $D_{\bigcirc} = D \cap V_{\bigcirc}$, $D_{\overline{\bigcirc}} = D \cap V_{\overline{\bigcirc}}$, $D_{\tilde{d}} = D \cap U_{\tilde{d}}$, and $D_{\breve{d}} = D \cap U_{\tilde{d}}$. Additionally, note that $D_{\tilde{d}} \subseteq U_{\tilde{d}} = U^0\left(G, \tilde{d}\right)$.

Let us first prove that $D_{\breve{d}} \subseteq \bigcirc\text{-}A^*_{\breve{d}}(G)$. Assume, for the sake of contradiction, that there exists a $v \in D_{\breve{d}}$ such that $v \notin \bigcirc\text{-}A^*_{\breve{d}}(G)$. Subsequently, $v \notin \bigcirc\text{-}Attr\left(G, U_{p(v)-1}(G)\right)$. This means that there exists a strategy of $\overline{\bigcirc}$ for v such that either it avoids $U_{p(v)-1}(G)$, or for every $u \in U_{p(v)-1}(G)$ the token visits after v satisfies $p(v) < p(u)$. Therefore, v is not winning for $\bigcirc$. Moreover, if the winning strategy $\varrho_{\bigcirc}$ on D does not visit v infinitely often, then there exists a $D' \subsetneq D$ such that

D' is a dominion. Thus, in both cases, it contradicts that D is a minimal dominion. Hence, such v does not exist.

By definition of D, we have that $\forall v \in D_{\overline{\bigcirc}} \cap D_{\tilde{d}} : E(v) \subseteq D$ and $\forall v \in D_{\bigcirc} \cap D_{\tilde{d}} : E(v) \cap D \neq \emptyset$. Therefore, if there exists a k and $u \in D_{\tilde{d}}$ such that $u \notin U^{k+1}\left(G, \tilde{d}\right)$ and $D_{\tilde{d}} \subseteq U^{k}\left(G, \tilde{d}\right)$, then there must exist some

$$v \in D_{\check{d}} \cap \overline{\bigcirc}\text{-}Attr\left(G, \left(V \setminus A^{k}\left(G, \tilde{d}\right)\right) \cup \bigcirc\text{-}A'_{\tilde{d}}\left(G, A^{k}\left(G, \tilde{d}\right)\right)\right).$$

Without loss of generality, assume that v is the first vertex of D to exist as defined. That is, for some $j \leq k$, it holds that

$$v \in D_{\check{d}} \cap \overline{\bigcirc}\text{-}Attr\left(G, \left(V \setminus A^{j}\left(G, \tilde{d}\right)\right) \cup \bigcirc\text{-}A'_{\tilde{d}}\left(G, A^{j}\left(G, \tilde{d}\right)\right)\right)$$

and

$$D \cap \overline{\bigcirc}\text{-}Attr\left(G, \left(V \setminus A^{j-1}\left(G, \tilde{d}\right)\right) \cup \bigcirc\text{-}A'_{\tilde{d}}\left(G, A^{j-1}\left(G, \tilde{d}\right)\right)\right) = \emptyset.$$

By the definition of D, v cannot be attracted to $\overline{\bigcirc}\text{-}Attr\left(G, \left(V \setminus A^{j}\left(G, \tilde{d}\right)\right) \cup \bigcirc\text{-}A'_{\tilde{d}}\left(G, A^{j}\left(G, \tilde{d}\right)\right)\right)$; therefore, it must be that $v \in \left(V \setminus A^{j}\left(G, \tilde{d}\right)\right) \cup \bigcirc\text{-}A'_{\tilde{d}}\left(G, A^{j}\left(G, \tilde{d}\right)\right)$. However, because $D_{\tilde{d}} \subseteq U^{j}\left(G, \tilde{d}\right)$ implies $D_{\check{d}} \subseteq \bigcirc\text{-}A^{*}_{\tilde{d}}(G)$, it cannot be that $v \in \bigcirc\text{-}A'_{\tilde{d}}\left(G, A^{j}\left(G, \tilde{d}\right)\right)$. Moreover, by the definition of D and $D_{\tilde{d}} \subseteq U^{j}\left(G, \tilde{d}\right)$, it cannot be that $v \in \left(V \setminus A^{j}\left(G, \tilde{d}\right)\right)$. Hence, v cannot exist.

Therefore, there does not exist a

$$u \in D \cap \overline{\bigcirc}\text{-}Attr\left(G, \left(V \setminus A^{k}\left(G, \tilde{d}\right)\right) \cup \bigcirc\text{-}A'_{\tilde{d}}\left(G, A^{k}\left(G, \tilde{d}\right)\right)\right)$$

for any k. Hence, $D \subseteq A\left(G, \tilde{d}\right)$. □

At this point, the algorithm can be introduced. Algorithm 1 contains the algorithm presented by this paper. It utilises the new attractor-type function $A(G, d)$ and *Theorem* 3. The algorithm applies $A(G, d)$ to the maximum priority of both players, guaranteeing that it finds at least the minimal dominion. It continues doing so until the graph is empty. Theorem 4 proves the termination of the algorithm, and Theorem 5 uses it to prove the correctness of the algorithm. Note that the CONTINUE on line 12 is to prevent either G from being empty or all vertices with priority $\hat{d}$ from being removed from G.

Lemma 4. *Either $A(G, \hat{d})$ or $A(G, \overline{d})$ is non-empty or G is empty.*

Proof. Given that we compute $A(G, \hat{d})$ and $A(G, \overline{d})$ and every parity game has a minimal dominion, it follows from Theorem 3 that at least one is non-empty. Otherwise, the graph is empty. □

Theorem 5. *Algorithm 1 solves parity games.*

Proof. By Theorem 2 any vertex added to either $W_{\Diamond}$ or $W_{\Box}$ is indeed won by that player. Moreover, either $A(G, \overline{d})$ or $A(G, \hat{d})$ is non-empty or G is empty which follows from Theorem 4. Therefore, each iteration of the while loop in line 3 either removes at least one vertex or terminates, and the loop terminates after at most $|V|$ iterations. Lastly, all vertices removed from G are added to either $W_{\Diamond}$ or $W_{\Box}$, which follows from lines 8-10 and 14-16. □

Algorithm 1 Parity Game Algorithm ($G = (V, E, p, (V_\Diamond, V_\Box))$)

Let $n = |V|$ and $m = |E|$.
Let D be the ordered list of all priorities of G.
Let $W_\Diamond, W_\Box = \emptyset, \emptyset$.
while $V \neq \emptyset$ **do**
    $\hat{d} \leftarrow$ the max priority of G.
    $\bigcirc \leftarrow \Diamond$ if d is even; otherwise, $\Box$.
    $\overline{d} \leftarrow$ the max priority of $\overline{\bigcirc}$.

    Compute $A\big(G, \overline{d}\big)$.
    $W_{\overline{\bigcirc}} = W_{\overline{\bigcirc}} \cup A\big(G, \overline{d}\big)$.
    $G = G \setminus A\big(G, \overline{d}\big)$.
    if $A\big(G, \overline{d}\big) \neq \emptyset$ **then**
        continue
    end if

    Compute $A\Big(G, \hat{d}\Big)$.
    $W_\bigcirc = W_\bigcirc \cup A\Big(G, \hat{d}\Big)$.
    $G = G \setminus A\Big(G, \hat{d}\Big)$.
end while
return $(W_\Diamond, W_\Box)$

3.1 Complexity Analysis

The previous section shows that Algorithm 1 correctly solves parity games. This section will prove that it does so in polynomial time. Before we demonstrate that, we first prove that the complexity of $A(G, d)$ is polynomial with Theorem 6. For simplicity, let $n = |V|$ and $m = |E|$. Note that it is well established that an attractor can be computed in $\mathcal{O}(n + m)$ time.

Lemma 6. *$A(G, d)$ runs in $\mathcal{O}(n \cdot (n + m))$ time.*

Proof. $A(G, d)$ uses various sub-functions to compute itself, the first are $U_d(G)$ and $\bigcirc$-$A^*_d(G)$. The former runs in $\mathcal{O}(d \cdot n)$ and the latter in $\mathcal{O}(d \cdot (n + m))$. Moreover, note that both these functions only depend on G and can be precomputed for the calculation of A. $\bigcirc$-$A'_d(G, A)$ runs in $\mathcal{O}(n + m)$. A single iteration of U^k also takes $\mathcal{O}(n + m)$ time, as $\bigcirc$-$A'_d(G, A)$ can be computed before the attractor operation, similarly for a single iteration of A^k. Therefore, $A(G, d)$ runs in $\mathcal{O}(k \cdot (n + m))$ and $A(G, d)$ cannot iterate more than n times. Hence, $A(G, d)$ runs in $\mathcal{O}(n \cdot (n + m))$. □

Using this lemma, we can prove that Algorithm 1 runs in polynomial time. In particular, we prove the following theorem.

Theorem 7. *Algorithm 1 runs in $\mathcal{O}\left(n^2 \cdot (n + m)\right)$ time.*

Proof. It is clear that the loop of line 3 dominates the runtime and that $A(G, d)$ dominates the loop. Furthermore, from the proof Theorem 5, we know that the loop is executed at most n times. Hence, Algorithm 1 runs in $\mathcal{O}\left(n^2 \cdot (n + m)\right)$. □

Theorem 8. *There exists an algorithm that solves parity games in $\mathcal{O}\left(n^2 \cdot (n + m)\right)$.*

Proof. Follows from Theorems 5 and 7. □

4 Conclusion

This paper introduced the first full-solver in polynomial-time algorithm for general parity games, solving the decade-old question of whether parity games are in **P**, first posed in 1991 by Emerson and Jutla [EJ91]. The algorithm is based on a new type of attractor-like function that guarantees finding the minimal dominion of the parity game. The algorithm repeatedly utilises these functions and peels off at least the minimal dominion in each iteration. This result also immediately gives polynomial time algorithms for the model checking problem in modal μ-calculus, as well as for the emptiness problem for non-deterministic automata on finite trees with parity acceptance conditions.

The practicality of this algorithm needs further investigation, e.g., for use within model checkers. The running time is $\mathcal{O}\left(n^2 \cdot (n + m)\right)$, but the number of attractor calls still grows quite rapidly. Further investigation can lead to either more efficient polynomial-time algorithms that utilise other well-known techniques in solving parity games. Or, an investigation can determine the types of parity games in which this algorithm is practically efficient.

This paper did not focus on the consequences of this algorithm. However, parity games relate to various fields and have influenced algorithms. The effects on Muller games and their algorithms should be researched. Also, further research into synthesis and ω-word automata translation can yield improvements in these fields.

References

[BL18] Udi Boker and Karoliina Lehtinen. On the Way to Alternating Weak Automata. In Sumit Ganguly and Paritosh Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018)*, volume 122 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:22, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969. doi:10.4230/LIPIcs.FSTTCS.2018.21.

[BSV04] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. Memoryless determinacy of parity and mean payoff games: a simple proof. *Theoretical Computer Science*, 310(1):365–378, January 2004. doi:10.1016/S0304-3975(03)00427-4.

[CJK+17] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 252–263, New York, NY, USA, June 2017. Association for Computing Machinery. doi:10.1145/3055399.3055409.

[DJL19] Laure Daviaud, Marcin Jurdziński, and Karoliina Lehtinen. Alternating Weak Automata from Universal Trees. In Wan Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory (CONCUR 2019)*, volume 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969. doi:10.4230/LIPIcs.CONCUR.2019.18.

[DJT20] Laure Daviaud, Marcin Jurdziński, and K. S. Thejaswini. The Strahler Number of a Parity Game. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 123:1–123:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969. doi:10.4230/LIPIcs.ICALP.2020.123.

[EJ91] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, 1991. IEEE Comput. Soc. Press. doi:10.1109/SFCS.1991.185392.

[EJS93] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In Costas Courcoubetis, editor, *Computer Aided Verification*, pages 385–396, Berlin, Heidelberg, 1993. Springer. doi:10.1007/3-540-56922-7_32.

[EJS01] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theoretical Computer Science*, 258(1):491–522, May 2001. doi:10.1016/S0304-3975(00)00034-7.

[Fea10] John Fearnley. Exponential Lower Bounds for Policy Iteration. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis,

editors, *Automata, Languages and Programming*, pages 551–562, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-14162-1_46.

[FHZ11] Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, STOC '11, pages 283–292, New York, NY, USA, June 2011. Association for Computing Machinery. doi:10.1145/1993636.1993675.

[FJdK+19] John Fearnley, Sanjay Jain, Bart de Keijzer, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi-polynomial time and quasi-linear space. *International Journal on Software Tools for Technology Transfer*, 21(3):325–349, June 2019. doi:10.1007/s10009-019-00509-3.

[FL09] Oliver Friedmann and Martin Lange. Solving Parity Games in Practice. In Zhiming Liu and Anders P. Ravn, editors, *Automated Technology for Verification and Analysis*, pages 182–196, Berlin, Heidelberg, 2009. Springer. doi:10.1007/978-3-642-04761-9_15.

[Fri11] Oliver Friedmann. A Subexponential Lower Bound for Zadeh's Pivoting Rule for Solving Linear Programs and Games. In Oktay Günlük and Gerhard J. Woeginger, editors, *Integer Programming and Combinatoral Optimization*, pages 192–206, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-20807-2_16.

[JL17] Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '17, pages 1–9, Reykjavík, Iceland, June 2017. IEEE Press.

[JPZ06] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 117–123, USA, January 2006. Society for Industrial and Applied Mathematics.

[Jur98] Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters*, 68(3):119–124, November 1998. doi:10.1016/S0020-0190(98)00150-1.

[Jur00] Marcin Jurdziński. Small Progress Measures for Solving Parity Games. In Horst Reichel and Sophie Tison, editors, *STACS 2000*, pages 290–301, Berlin, Heidelberg, 2000. Springer. doi:10.1007/3-540-46541-3_24.

[KV98] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 224–233, New York, NY, USA, May 1998. Association for Computing Machinery. doi:10.1145/276698.276748.

[Leh18] Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 639–648, New York, NY, USA, July 2018. Association for Computing Machinery. doi:10.1145/3209108.3209115.

[LPSW22] Karoliina Lehtinen, Paweł Parys, Sven Schewe, and Dominik Wojtczak. A Recursive Approach to Solving Parity Games in Quasipolynomial Time. *Logical Methods in Computer Science*, Volume 18, Issue 1, January 2022. Publisher: Episciences.org. doi:10.46298/lmcs-18(1:8)2022.

[Maz02] René Mazala. Infinite Games. In Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors, *Automata Logics, and Infinite Games: A Guide to Current Research*, pages 23–38. Springer, Berlin, Heidelberg, 2002. doi:10.1007/3-540-36387-4_2.

[McN93] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, December 1993. doi:10.1016/0168-0072(93)90036-D.

[Par19] Paweł Parys. Parity Games: Zielonka's Algorithm in Quasi-Polynomial Time, April 2019. arXiv:1904.12446 [cs]. doi:10.48550/arXiv.1904.12446.

[Par20] Paweł Parys. Parity Games: Another View on Lehtinen's Algorithm. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969. doi:10.4230/LIPIcs.CSL.2020.32.

[PV01] Viktor Petersson and Sergei Vorobyov. A randomized subexponential algorithm for parity games. *Nordic J. of Computing*, 8(3):324–345, September 2001.

[PW23] Paweł Parys and Aleksander Wiącek. Improved Complexity Analysis of Quasi-Polynomial Algorithms Solving Parity Games. In Gianluca Della Vedova, Besik Dundua, Steffen Lempp, and Florin Manea, editors, *Unity of Logic and Computation*, pages 275–286, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-36978-0_22.

[Rab75] Michael O. Rabin. Automata on infinite objects and Church's problem. *The Journal of Symbolic Logic*, 40(4):623–623, December 1975. doi:10.2307/2271835.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1):135–183, June 1998. doi:10.1016/S0304-3975(98)00009-7.