# A deterministic subexponential algorithm for solving parity games

*Marcin Jurdziński* * *Mike Paterson* † *Uri Zwick* ‡

**Abstract**

The existence of polynomial time algorithms for the solution of parity games is a major open problem. The fastest known algorithms for the problem are *randomized* algorithms that run in subexponential time. These algorithms are all ultimately based on the randomized subexponential simplex algorithms of Kalai and of Matoušek, Sharir and Welzl. Randomness seems to play an essential role in these algorithms. We use a completely different, and elementary, approach to obtain a *deterministic* subexponential algorithm for the solution of parity games. Our deterministic algorithm is almost as fast as the randomized algorithms mentioned above.

## 1 Introduction

A parity game [EJ91, GTW02] is played on a graph $(V, E)$ by two players, *Even* and *Odd*, who move a token from vertex to vertex along the edges of the graph so that an infinite path is formed. A partition $(V_0, V_1)$ is given of the set $V$ of vertices; player Even moves if the token is at a vertex of $V_0$ and player Odd moves if the token is at a vertex of $V_1$. Finally a priority function $p : V \to \{ 1, 2, \ldots, d \}$ is given. The players compete for the parity of the highest priority occurring infinitely often: player Even wins if $\limsup_{i\to\infty} p(v_i)$ is even while player Odd wins if it is odd, where $v_0, v_1, v_2, \ldots$ is the infinite path formed by the players. The algorithmic problem of solving parity games is, given a parity game $G = (V_0, V_1, E, p)$ and an initial vertex $v_0 \in V$, to determine whether player Even has a winning strategy in the game if the token is initially placed on vertex $v_0$. Algorithms for solving parity games [Zie98, Jur00, VJ00, GTW02, BSV03] usually compute the winning sets $W_0$ and $W_1$, i.e., the sets of vertices from which players Even and Odd, respectively, have a winning strategy. By the Determinacy Theorem for parity games [EJ91, GTW02] the winning sets $W_0$ and $W_1$ form a partition of the set of vertices $V$. Unfortunately none of these algorithms is known to run in polynomial time and the existence of a polynomial time algorithm for the solution of parity games is a major open problem [EJS93, GTW02].

The original motivation for the study of parity games comes from the area of formal verification of systems by temporal logic model checking [CGP99, GTW02]. The problem of solving parity games is polynomial time equivalent to the non-emptiness problem of $\omega$-automata on infinite trees with Rabin-chain acceptance conditions [EJS93] and to the the model checking problem of the modal $\mu$-calculus (modal fixpoint logic) which is a specification formalism of choice in formal specification and validation [Eme96, GTW02]. The model checking problem is a fundamental algorithmic problem in automated hardware and software verification [Eme96, CGP99].

Another important motivation to study the problem of solving parity games is its intriguing complexity theoretic status: the problem is known to be in NP ∩ co-NP [EJS93] and even in UP ∩ co-UP [Jur98] but, as mentioned, despite considerable efforts of the community [EJS93, Jur00, VJ00, GTW02, BSV03, Obd03] no polynomial time algorithm has been found so far. Moreover, parity games are polynomial time reducible to mean payoff games [ZP96] and simple stochastic games [Con92] and a stochastic generalization of parity games was also studied [dAM04, CJH04]. The problems of solving all those games are also in UP ∩ co-UP [Jur98, CJH04] and Condon has shown that simple stochastic games are log-space complete in the class of log-space randomized alternating Turing machines [Con92].

Let $n = |V|$ and $m = |E|$ be the number of vertices and edges of a parity game graph and let $d$ be the number of different priorities assigned to vertices by the priority function $p : V \to \{1, 2, \ldots, d\}$. For parity games with a small number of priorities, more specifically if $d = O(n^{1/2})$, the progress-measure lifting algorithm [Jur00] gives currently the best time complexity of $O(dm \cdot (2n/d)^{d/2})$. If $d = \Omega(n^{1/2+\varepsilon})$ then the randomized

*Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom. (mju@dcs.warwick.ac.uk)

†Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom. (msp@dcs.warwick.ac.uk)

‡School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. (zwick@cs.tau.ac.il)

algorithm of Björklund et al. [BSV03] has a better (expected) running time bound of $n^{O(\sqrt{n/\log n})}$.

The main contribution of this paper is a *deterministic* algorithm for solving parity games which achieves roughly the same complexity as the randomized algorithm of Björklund et al. [BSV03]: the complexity of our algorithm is $n^{O(\sqrt{n/\log n})}$ if the out-degree of all vertices is bounded, and it is $n^{O(\sqrt{n})}$ otherwise. The randomized algorithm of Björklund et al. [BSV03] is based on the randomized algorithm of Ludwig for simple stochastic games [Lud95], which in turn is inspired by the subexponential randomized simplex algorithms for linear programming by Kalai and by Matoušek et al. [Kal92, MSW96]. In contrast, our deterministic algorithm for parity games is obtained by a modification of a more elementary algorithm of McNaughton and Zielonka for parity games [Zie98, GTW02]. The methods we use are thus very different from those of Ludwig and Björklund et al. Our method is applicable, so it seems, only to parity games, while the randomized algorithm of Ludwig can be adapted to optimization of non-trivial classes of pseudo-Boolean functions, including strategy evaluation functions of parity, mean payoff, and simple stochastic games [BSV04, Hal04].

## 2 Preliminaries

A *parity game* $G = (V_0, V_1, E, p)$ is composed of two disjoint sets of vertices $V_0$ and $V_1$, an edge set $E \subseteq V \times V$, where $V = V_0 \cup V_1$, and a *priority* function $p : V_0 \cup V_1 \to \{1, 2, \ldots, d\}$, for some integer $d$, defined on its vertices. Every vertex $u \in V$ has at least one outgoing edge $(u, v) \in E$. The game is played by two players: *Even*, also referred to as Player 0, and *Odd*, also referred to as Player 1. The game starts at some vertex $v_0 \in V$. The players construct an infinite path as follows: Let $u$ be the last vertex added so far to the path. If $u \in V_0$, then Player 0 chooses an edge $(u, v) \in E$. Otherwise, if $u \in V_1$, then Player 1 chooses an edge $(u, v) \in E$. In either case, vertex $v$ is added to the path, and a new edge is then chosen by either Player 0 or Player 1. As each vertex has at least one outgoing edge, the path constructed can always be continued. Let $v_0, v_1, \ldots$ be the infinite path constructed by the two players, and $p(v_0), p(v_1), \ldots$ the sequence of the priorities of the vertices on the path. Player 0 wins the game if the largest priority seen infinitely many times is even, while Player 1 wins otherwise.

Given a parity game $G = (V_0, V_1, E, p)$, we let $V(G) = V_0 \cup V_1$ be the set of vertices of $G$, $E(G) = E$ be the set of edges of $G$, and $d(G) = \max_{v \in V(G)} p(v)$ be the highest priority attached to a vertex of $G$. We also let $A_d(G) = p^{-1}(d)$ be the set of vertices labelled by $d$, the highest priority in the game.

The winning set for Player $i$, denoted by $win_i(G)$, is the set of vertices of the game from which Player $i$ has a way of winning, no matter what her opponent does. By the Determinacy Theorem for parity games [EJ91, GTW02] $win_0(G) \cup win_1(G) = V$.

A set $B \subseteq V$ is said to be *$i$-closed*, where $i \in \{0, 1\}$, if for every $u \in B \cap V_i$ there exists $v \in B$ such that $(u, v) \in E$, and if for every $(u, v) \in E$ with $u \in B \cap V_{\neg i}$ we have $v \in B$. (We use $\neg i$ for the complement of $i$ in $\{0, 1\}$.) In other words, a set $B$ is $i$-closed if Player $i$ can always choose to stay in $B$ while Player $\neg i$ cannot escape from it.

LEMMA 2.1. *For every $i \in \{0, 1\}$, the set $win_i(G)$ is $i$-closed.*

*Proof.* Let $j = \neg i$. Assume for the sake of contradiction that the set $win_i(G)$ is not $i$-closed. Then there is a vertex $v \in win_i(G)$ from which Player $j$ can escape to $V \setminus win_i(G)$. By the Determinacy Theorem we have that $V \setminus win_i(G) = win_j(G)$ and hence Player $j$ has the following winning strategy from vertex $v$: first escape to the set $V \setminus win_i(G)$, then "restart" the play and follow a winning strategy in $win_j(G)$. This contradicts $v \in win_i(G)$. □

Let $A \subseteq V$ be an arbitrary set. The $i$-attraction of the set $A$, denoted $attr_i(A)$, is the set of vertices from which Player $i$ can force the game to enter the set $A$ at least once.

LEMMA 2.2. *For any set $A \subseteq V$ and $i \in \{0, 1\}$, the set $attr_i(A)$ can be computed in $O(m)$ time, where $m = |E|$ is the number of edges in the game.*

*Proof.* The vertices of $A$ are clearly in $attr_i(A)$. We thus initialize $B \leftarrow A$. We then iteratively add to $B$ every vertex of $V_i$ that has at least one edge going into $B$, and every vertex of $V_{\neg i}$ whose all edges go into $B$. We stop when no new vertices can be added to $B$. It is easy to see that this processes can be performed in $O(m)$ time, and that when it ends $B = attr_i(A)$, as required. □

LEMMA 2.3. *For any set $A \subseteq V$ and $i \in \{0, 1\}$, the set $V \setminus attr_i(A)$ is $(\neg i)$-closed.*

*Proof.* Let $v \in V \setminus attr_i(A)$. If $v \in V_{\neg i}$ then there must be an edge $(v, w) \in E$ from vertex $v$ into the set $V \setminus attr_i(A)$, i.e., such that $w \notin attr_i(A)$, since otherwise the iterative procedure from the proof of Lemma 2.2 would add vertex $v$ to $attr_i(A)$. Similarly, if $v \in V_i$ then all edges from vertex $v$ must go into the set $V \setminus attr_i(A)$. Therefore, the set $V \setminus attr_i(A)$ is $(\neg i)$-closed. □

If $B \subseteq V$ is such that every vertex in $V \setminus B$ has a successor in $V \setminus B$, then the game $G \setminus B$ is the game obtained from $G$ by removing the vertices of $B$ and all the edges that touch them. (A vertex $v$ is a successor of $u$ in $G = (V, E)$ if and only if $(u, v) \in E$.)

LEMMA 2.4. *Let $G$ be a parity game and let $i \in \{0, 1\}$ and $j = \neg i$. If $W \subseteq win_i(G)$ and $\overline{W} = attr_i(W)$, then $win_i(G) = win_i(G \setminus \overline{W}) \cup \overline{W}$ and $win_j(G) = win_j(G \setminus \overline{W})$.*

*Proof.* It suffices to exhibit strategies for Players $j$ and $i$ that are winning for them in the game $G$, respectively, from the sets $win_j(G \setminus \overline{W})$ and $win_i(G \setminus \overline{W}) \cup \overline{W}$. We claim that a winning strategy for Player $j$ from the set $win_j(G \setminus \overline{W})$ in the subgame $G \setminus \overline{W}$ is also winning for her from the same set in the original game $G$. It suffices to establish that Player $i$ cannot escape from the set $win_j(G \setminus \overline{W})$ in the game $G$. This follows from Lemma 2.3 (the set $V \setminus \overline{W}$ is $j$-closed in $G$, and hence Player $i$ cannot escape to $\overline{W}$) and from Lemma 2.1 (the set $win_j(G \setminus \overline{W})$ is $j$-closed in $G \setminus \overline{W}$, and hence Player $i$ cannot escape to $win_i(G \setminus \overline{W})$).

Now we exhibit a winning strategy for Player $i$ in the game $G$ from the set $win_i(G \setminus \overline{W}) \cup \overline{W}$. By the assumption that $W \subseteq win_i(G)$, there is a strategy $\sigma$ for Player $i$ in the game $G$ which is winning for her from all vertices in $W$. Let $\tau$ be a winning strategy for Player $i$ from the set $win_i(G \setminus \overline{W})$ in the subgame $G \setminus \overline{W}$. We construct a strategy $\pi$ for Player $i$ in the game $G$ by composing strategies $\tau$ and $\sigma$ in the following way: if the play so far is contained in the set $win_i(G \setminus \overline{W})$ then follow strategy $\tau$, otherwise force the play in a finite number of steps into the set $W$ and "restart" the play following the strategy $\sigma$ henceforth. The strategy $\pi$ is well-defined because by Lemma 2.1 Player $j$ can escape from $win_i(G \setminus \overline{W})$ only into the set $\overline{W}$. It is a winning strategy for Player $i$ because if strategy $\pi$ ever switches from following $\tau$ into following $\sigma$ then an infinite suffix of the play is winning for Player $i$ and in parity games removing an arbitrary finite prefix of a play does not change the winner. □

LEMMA 2.5. *Let $G$ be a parity game. Let $d = d(G)$ be the highest priority and let $A = A_d(G)$ be the set of vertices of highest priority. Let $i = d \bmod 2$ and $j = \neg i$. Let $G' = G \setminus attr_i(A)$. Then, $win_j(G') \subseteq win_j(G)$. Also, if $win_j(G') = \emptyset$, then $win_i(G) = V(G)$, i.e., Player $i$ wins from every vertex of $G$.*

*Proof.* We claim that a winning strategy for Player $j$ from vertices in the set $win_j(G')$ in the subgame $G'$ is also winning for her in the whole game $G$. Indeed, by Lemma 2.1 Player $i$ cannot escape from $win_j(G')$ to $win_i(G')$ and by Lemma 2.3 Player $i$ cannot escape from $win_j(G')$ to $attr_i(A)$.

Suppose now that $win_j(G') = \emptyset$. Let $\tau$ be a winning stategy for Player $i$ from $win_i(G')$ (which, by determinacy, is equal to $V \setminus attr_i(A)$) in the subgame $G'$. We construct a strategy $\pi$ for Player $i$ in the following way: if a play so far is contained in the set $win_i(G')$ then follow strategy $\tau$; otherwise the current vertex is in $attr_i(A)$ so force the play in a finite number of steps into the set $A$; moreover, each time the play reenters the set $win_i(G')$ "restart" the play and follow strategy $\tau$ henceforth, etc. If a play following the strategy $\pi$ eventually never leaves the set $win_i(G')$ then it has an infinite suffix played according to strategy $\tau$ and hence it is winning for Player $i$. Otherwise it leaves the set $win_i(G')$ infinitely often so it visits a vertex of the maximal priority $d$ infinitely often and hence it is winning for Player $i$ because $i = d \bmod 2$. □

## 3 An exponential algorithm

A simple exponential-time algorithm for the solution of parity games is given in Figure 1. This algorithm originates from the work of McNaughton [McN93] and was first presented for parity games by Zielonka [Zie98, GTW02]. Algorithm $\mathbf{win}(G)$ receives a parity game $G$ and returns the pair of winning sets $(win_0(G), win_1(G))$ for the two players.

Algorithm $\mathbf{win}(G)$ is based on Lemmas 2.4 and 2.5. It starts by letting $d$ be the largest priority in $G$, and $A$ be the set of vertices having this highest priority. It also lets $i = d \bmod 2$, the index of the player associated with the parity of this highest priority, and $j = \neg i$, the index of the other player. The algorithm then finds the winning sets $(U_0, U_1)$ of the smaller game $G' = G \setminus attr_i(A)$ using a recursive call. By Lemma 2.5, if $U_j = \emptyset$, then Player $i$ wins from all vertices of $G$ and we are done. Otherwise, again by Lemma 2.5, we know that $U_j \subseteq win_j(G)$. The algorithm then finds the winning sets $(U_0, U_1)$ of the smaller game $G'' = G \setminus attr_j(U_j)$ by a second recursive call. By Lemma 2.4 we then know that $win_i(G) = U_i$ and thus $win_j(G) = V(G) \setminus U_i$.

THEOREM 3.1. *Algorithm $\mathbf{win}(G)$ correctly finds the winning sets of the parity game $G$. Its running time is $O(2^n)$, where $n = |V(G)|$ is the number of vertices in $G$.*

*Proof.* The correctness of the algorithm follows from Lemmas 2.4 and 2.5, as argued above. Let $T(n)$ be a maximum running time of algorithm $\mathbf{win}(G)$ on a game with at most $n$ vertices. Algorithm $\mathbf{win}(G)$ makes two recursive calls $\mathbf{win}(G')$ and $\mathbf{win}(G'')$ on games with at most $n - 1$ vertices. Other than that it performs only $O(n^2)$ operations. (The most time consuming operations are the computations of the sets $attr_i(A)$ and

```
algorithm win(G)
if V(G) = ∅ then return (∅, ∅)
d ← d(G) ; A ← A_d(G)
i ← k mod 2 ; j ← ¬i
(U_0, U_1) ← win(G \ attr_i(A))
if U_j = ∅ then
    U_i ← V(G)
else
    (U_0, U_1) ← win(G \ attr_j(U_j))
    U_j ← V(G) \ U_i
endif
return (U_0, U_1)
```

Figure 1: An exponential algorithm for solving parity games.

$attr_j(U_j)$.) Thus $T(n) \leq 2T(n-1) + O(n^2)$. It is easy to see then that $T(n) = O(2^n)$. □

## 4 Finding small dominions

A set $W \subseteq V(G)$ is said to be a *$i$-dominion* if Player $i$ can win from any vertex of $W$ without ever leaving $W$. Note, in particular, that an $i$-dominion must also be $i$-closed. A set $W \subseteq V(G)$ is said to be a *dominion* if it is either a 0-dominion or a 1-dominion. Clearly $win_0(G)$, the winning set of Player 0, is a 0-dominion, and $win_1(G)$, the winning set of Player 1, is a 1-dominion.

LEMMA 4.1. *Let $G$ be a parity game on $n$ vertices and let $\ell \leq n/3$. A dominion of $G$ of size at most $\ell$, if one exists, can be found in $O((2en/\ell)^\ell)$ time.*

*Proof.* If $\ell \leq n/3$ then for all $i \leq \ell$, we have that $\binom{n}{i}/\binom{n}{i-1} \geq 2$. Therefore, the number $\sum_{i=1}^{\ell}\binom{n}{i}$ of subsets $W$ of $V$ of size at most $\ell$ is $O(\binom{n}{\ell})$. For each such subset $W$ we can form the game $G[W]$ which is the game $G$ restricted to $W$. If $G[W]$ is not a well formed game, i.e., if one of the vertices of $G[W]$ has no outgoing edges, then $W$ is clearly not a dominion. Otherwise, we apply the exponential algorithm of the previous section to $G[W]$ and find out, in $O(2^\ell)$ time, whether one of the players can win from all the vertices of $G[W]$. If so, then $W$ is a dominion, otherwise it is not. The total running time is $O(2^\ell\binom{n}{\ell}) = O((2en/\ell)^\ell)$, as required. □

In a game with bounded out-degrees we can find small dominions even faster. For simplicity, the lemma below and the analysis in Section 6 are stated for out-degree at most two. Note, however, that for every constant $b$, and for a game with $n$ vertices and with out-degree at most $b$, one can easily construct an equivalent game with at most $n(b-1)$ vertices and with out-degree at most two.

LEMMA 4.2. *Let $G$ be a parity game with $n$ vertices in which the out-degree of each vertex is at most two. A dominion of $G$ of size at most $\ell$, if one exists, can be found in $O(n2^\ell\ell\log\ell)$ time.*

*Proof.* The algorithm generates at most $O(n2^\ell)$ 0-closed sets of size at most $\ell$ that are candidates for being 0-dominions. Assume that every vertex in the graph has a unique "serial number". If a vertex has two outgoing edges then we say that the edge leading to the vertex with smaller serial number is the 0-th outgoing edge, and the other edge is the 1-st outgoing edge.

For every vertex $v \in V$ and a binary sequence $\langle a_1, \ldots, a_\ell\rangle \in \{0,1\}^\ell$, construct a set $W \subset V$ as follows. Start with $W = \{v\}$ and $r = 1$. Vertices added to $W$ are initially unmarked. As long as there is still an unmarked vertex in $W$, pick the unmarked vertex $u \in W$ with the smallest serial number and mark it. If $u \in V_0$, then add the endpoint of the $a_r$-th outgoing edge of $u$ to $W$, if it is not already there, and increment $r$. If $u \in V_1$, then add the endpoints of all the edges emanating from $u$ to $W$. If at some stage $|W| > \ell$, then discard the set $W$ and restart the construction by considering the next binary sequence.

If process above ends with $|W| \leq \ell$, then a 0-closed set of size at most $\ell$ was found. Furthermore, for every vertex $u \in W \cap V_0$, one of the outgoing edges of $u$ was selected. This corresponds to a suggested strategy for Player 0 in the game $G[W]$. Using an algorithm of King et al. [KKV01] we can check, in $O(\ell\log\ell)$ time, whether this is indeed a winning strategy for Player 0 from all the vertices of $W$. It is easy to see that if there is a 0-dominion of size at most $\ell$ in $G$, then the algorithm will find one.

Finding 1-dominions of size at most $\ell$ can be done in an analogous manner. □

## 5 The new subexponential algorithm

The new algorithm for solving parity games is given in Figure 2. The algorithm **new-win** starts by trying to find a dominion of size at most $\ell$, where $\ell = \lceil\sqrt{n}\rceil$ (and $\ell = \lceil\sqrt{n\log n}\rceil$ for games with bounded out-degree) is a parameter chosen to minimize the running time of the whole algorithm. If such a small dominion is found, then it is easy to remove it, and its attraction set, from the game and recurse on what is left over. If no such

```
algorithm new-win(G)
n ← |V(G)| ; ℓ ← ⌈√n⌉
if V(G) = ∅ then return (∅, ∅)
(W, i) ← dominion(G, ℓ)
if W ≠ ∅ then
    (U_0, U_1) ← new-win(G \ attr_i(W))
    U_i ← V(G) \ U_{¬i}
else
    (U_0, U_1) ← old-win(G)
endif
return(U_0, U_1)
```

```
algorithm old-win(G)
d ← d(G) ; A ← A_d(G)
i ← d mod 2 ; j ← ¬i
(U_0, U_1) ← new-win(G \ attr_i(A))
if U_j = ∅ then
    U_i ← V(G)
else
    (U_0, U_1) ← new-win(G \ attr_j(U_j))
    U_j ← V(G) \ U_i
endif
return (U_0, U_1)
```

Figure 2: The new subexponential algorithm for solving parity games.

small dominion is found, then **new-win**$(G)$ simply calls algorithm **old-win**$(G)$ which is almost identical to the exponential algorithm **win**$(G)$ of Section 3. The only difference between **old-win**$(G)$ and **win**$(G)$ is that the recursive calls are made to **new-win**$(G)$ and not to **win**$(G)$.

THEOREM 5.1. *Algorithm* **new-win**$(G)$ *correctly finds the winning sets of a parity game $G$. Its running time is $n^{O(\sqrt{n})}$.*

*Proof.* The correctness of the algorithm is immediate. We next analyse its running time. Let $T(n)$ be the maximum running time of **new-win**$(G)$ on a game with at most $n$ vertices.

Algorithm **new-win**$(G)$ starts by trying to find winning sets of size at most $\ell$. By Lemma 4.1 this takes $O((2en/\ell)^\ell) = n^{O(\sqrt{n})}$ time. If a non-empty dominion is found, then the algorithm simply proceeds on the remaining game, which has at most $n-1$ vertices, and the remaining running time is therefore at most $T(n-1)$. Otherwise, a call to **old-win**$(G)$ is made. This results in a call to **new-win**$(G \setminus attr_i(A))$, which takes at most $T(n-1)$ time. If the set $U_j$ returned by the call is empty, we are done. Otherwise, as $U_j$ is a $j$-dominion of $G$, we know that $|U_j| > \ell$. Thus, the second recursive call **new-win**$(G \setminus attr_j(U_j))$ takes only $T(n-\ell)$ time. We thus get

$$T(n) \leq n^{O(\sqrt{n})} + T(n-1) + T(n-\ell)\ .$$

This recurrence relation is analysed in the next section where it is shown that $T(n) = n^{O(\sqrt{n})}$. □

A slightly better complexity is achieved on graphs with out-degree bounded by two.

THEOREM 5.2. *Consider the algorithm* **new-win**$(G)$ *in which the variable $\ell$ is set to $\lceil \sqrt{n \log n} \rceil$. If every vertex in the game $G$ has out-degree at most two then the running time of the algorithm is $n^{O(\sqrt{n/\log n})}$.*

*Proof.* Note that if $\ell = \lceil n \log n \rceil$ then $O(n 2^\ell \ell \log \ell) = n^{O(\sqrt{n/\log n})}$. Therefore, by Lemma 4.2 and by the analysis in the proof of the previous Theorem, the time complexity $T(n)$ satisfies the following recurrence:

$$T(n) \leq n^{O(\sqrt{n/\log n})} + T(n-1) + T(n-\ell).$$

This recurrence is analysed in the next section where it is shown that $T(n) = n^{O(\sqrt{n/\log n})}$. □

## 6 Analysis of the new algorithm

In this section we analyze the recurrences derived in the previous section that characterize the running time of the new algorithm. First we consider the recurrence for game graphs with arbitrary out-degrees.

THEOREM 6.1. *If $T(n)$ is a positive function, such that we have*

$$T(n) \ \leq \ n^{O(\sqrt{n})} + T(n-1) + T(n-\ell),$$

*where $\ell = \lceil \sqrt{n} \rceil$, then $T(n) = n^{O(\sqrt{n})}$.*

*Proof.* Define the function $t(n)$ as follows:

$$t(n) = \begin{cases} 1 + t(n-1) + t(n - \lceil \sqrt{n} \rceil) & \text{if } n \geq 2, \\ 1 & \text{otherwise.} \end{cases}$$

It is easy to prove by induction that

$$T(n) \leq n^{O(\sqrt{n})} \cdot t(n).$$

Therefore, it suffices to establish the following lemma.

LEMMA 6.1. $t(n) = n^{O(\sqrt{n})}$.

*Proof.* Observe that $t(n)$ is the number of nodes in the binary tree whose root is labelled by $n$, and in which every node labelled by a number $k \geq 2$ has two children: the left child labelled by $k-1$ and the right child labelled by $k - \lceil\sqrt{k}\rceil$. Note that every path from the root to a leaf has length at most $n$.

We argue that on every path down the tree from the root there are at most $O(\sqrt{n})$ right children. As long as the labels on the path are no smaller than $n/2$, moving to a right child decreases the label by at least $\sqrt{n/2}$. Therefore, there can be at most $(n/2)/\sqrt{n/2} = \sqrt{n/2}$ right children on the path with labels exceeding $\sqrt{n/2}$. A similar argument gives $\sqrt{n/2^i}$ upper bound on the number of right children on a path whose labels are between $n/2^i$ and $n/2^{i-1}$. Hence the number of right children on a path down the tree from the root is bounded by $\sqrt{n} \cdot \sum_{i=1}^{\infty}(1/\sqrt{2})^i$, i.e., it is $O(\sqrt{n})$.

Note that each such a path is uniquely determined by the positions (out of at most $n$) at which the path visits a right child. It follows that there are at most $\binom{n}{O(\sqrt{n})} = n^{O(\sqrt{n})}$ nodes in the tree, and hence $t(n) = n^{O(\sqrt{n})}$. □ □

The running time of the algorithm for graphs with out-degree at most two satisfies a tighter recurrence relation, which is analysed similarly in the next theorem.

THEOREM 6.2. *If $T(n)$ is a positive function, such that we have*

$$T(n) \leq n^{O(\sqrt{n/\log n})} + T(n-1) + T(n-\ell),$$

*where $\ell = \lceil\sqrt{n \log n}\rceil$, then $T(n) = n^{O(\sqrt{n/\log n})}$.*

*Proof.* We proceed in a similar way to the proof of Theorem 6.1. If we define the function $t(n)$ by:

$$t(n) = \begin{cases} 1 + t(n-1) + t(n - \lceil\sqrt{n \log n}\rceil) & \text{if } n \geq 2, \\ 1 & \text{otherwise,} \end{cases}$$

then it is again easy to show that

$$T(n) \leq n^{O(\sqrt{n/\log n})} \cdot t(n).$$

We only need to prove that $t(n) = n^{O(\sqrt{n/\log n})}$. Consider the tree for the function $t(n)$ similar to the one in the proof of Lemma 6.1. When a right child is visited on a path down the tree starting from the root, then the decrease of the label is at least the decrease incurred by visiting $\sqrt{\log n}$ right children in the tree from the proof of Lemma 6.1, because $\ell = \lceil\sqrt{n \log n}\rceil$ and we had $\ell = \lceil\sqrt{n}\rceil$ in Lemma 6.1. Therefore, the number of right children on every path down the tree is $O(\sqrt{n/\log n})$. Hence there are at most $\binom{n}{O(\sqrt{n/\log n})} = n^{O(\sqrt{n/\log n})}$ nodes in the tree. □

## 7 Concluding remarks

We have obtained the first deterministic subexponential algorithm for solving parity games. Our algorithm does not seem to extend in an obvious way to the solution of the more general mean payoff games and simple stochastic games.

## References


[BSV03] H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In *Symposium on Theoretical Aspects of Computer Science, STACS 2003*, volume 2607 of *LNCS*, pages 663–674. Springer, 2003.

[BSV04] H. Bjorklund, S. Sandberg, and S. Vorobyov. Randomized subexponential algorithms for infinite games. Technical Report 2004-09, DIMACS, 2004.

[CGP99] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[CJH04] K. Chatterjee, M. Jurdziński, and T. A. Henzinger. Quantitative stochastic parity games. In *Symposium on Discrete Algorithms, SODA 2004*, pages 114–123. ACM/SIAM, 2004.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[dAM04] L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68:374–397, 2004.

[EJ91] E. A. Emerson and C. Jutla. Tree automata, $\mu$-calculus and determinacy. In *Symposium on Foundations of Computer Science, FOCS'91*, pages 368–377. IEEE Computer Society Press, 1991.

[EJS93] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of $\mu$-calculus. In *International Conference on Computer-Aided Verification, CAV'93*, volume 697 of *LNCS*, pages 385–396. Springer, 1993.

[Eme96] E. A. Emerson. Model checking and mu-calculus. In N. Immerman and Ph. G. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 185–214. American Mathematical Society, 1996.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games. A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

[Hal04] N. Halman. *Discrete and Lexicographic Helly Theorems and Their Relations to LP-type Problems*. PhD thesis, Tel-Aviv University, 2004.

[Jur98] M. Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters*, 68(3):119–124, November 1998.

[Jur00] M. Jurdziński. Small progress measures for solving parity games. In *Symposium on Theoretical Aspects of Computer Science, STACS 2000*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000.

[Kal92] G. Kalai. A subexponential randomized simplex algorithm (Extended abstract). In *Symposium on Theory of Computing, STOC'92*, pages 475–482. ACM Press, 1992.

[KKV01] V. King, O. Kupferman, and M. Y. Vardi. On the complexity of parity word automata. In *Foundations of Software Science and Computation Structures, FoSSaCS 2001*, volume 2030 of *LNCS*, pages 276–286. Springer, 2001.

[Lud95] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.

[McN93] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[MSW96] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4/5):498–516, 1996.

[Obd03] J. Obdržálek. Fast mu-calculus model checking when tree-width is bounded. In *International Conference on Computer-Aided Verification, CAV 2003*, volume 2725 of *LNCS*, pages 80–92. Springer, 2003.

[VJ00] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games (Extended abstract). In *International Conference on Computer-Aided Verification, CAV 2000*, volume 1855 of *LNCS*, pages 202–215. Springer, 2000.

[Zie98] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.

[ZP96] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.