RAIRO-Theor. Inf. Appl. 45 (2011) 449–457
DOI: 10.1051/ita/2011124

Available online at:
www.rairo-ita.org

RECURSIVE ALGORITHM FOR PARITY GAMES REQUIRES EXPONENTIAL TIME

Oliver Friedmann[1]

Abstract. This paper presents a new lower bound for the recursive algorithm for solving parity games which is induced by the constructive proof of memoryless determinacy by Zielonka. We outline a family of games of linear size on which the algorithm requires exponential time.

Mathematics Subject Classification. 05C57.

1. Introduction

Parity games are simple two-player games of perfect information played on directed graphs whose nodes are labeled with natural numbers, called priorities. A play in a parity game is an infinite sequence of nodes whose winner is determined by the highest priority occurring infinitely often.

Solving parity games appears in several fields of theoretical computer science, *e.g.* as solution to the problem of complementation of tree automata [1, 5] or as algorithmic back end to the model checking problem of the modal μ-calculus [2,12].

There are many algorithms that solve parity games, such as the divide-and-conquer algorithm due to Zielonka [14] and its recent improvement by Jurdziński *et al.* [8], the small progress measures algorithm due to Jurdziński [7] with its recent improvement by Schewe [9], the model-checking algorithm due to Stevens and Stirling [11] and finally the two strategy improvement algorithms by Vöge and Jurdziński [13] and Schewe [10]. From now on, we will refer to Zielonka's divide-and-conquer algorithm as the "recursive algorithm".

Solving parity games is one of the few problems that belong to the complexity class $\mathtt{NP} \cap \mathtt{coNP}$ and that is not (yet) known to belong to $\mathtt{P}$ [2]. It has also been

Keywords and phrases. Parity games, recursive algorithm, lower bound, μ calculus, model checking.

[1] Institut für Informatik, LMU München 80538 Munich, Germany.
Oliver.Friedmann@googlemail.com

Article published by EDP Sciences

© EDP Sciences 2011

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press

shown that solving parity games belongs to UP∩coUP [6]. The currently best known upper bound on the deterministic solution of parity games is $\mathcal{O}(|E| \cdot |V|^{\frac{1}{3}|\Omega|})$ due to Schewe's big-step algorithm [9].

In this paper we will investigate the worst-case runtime behaviour of the recursive algorithm, an algorithm whose practicality is usually highly underestimated. In fact, it ranks among the best: the strategy iteration algorithms due to Vöge and Jurdziński [13] and Schewe [10], the recursive algorithm due to Zielonka [14] as well as the small progress measures algorithm due to Jurdziński [7] seem to be the best algorithms in practice [4].

Section 2 defines the basic notions of parity games and some notations that are employed throughout the paper. Section 3 recaps the recursive algorithm. In Section 4, we outline a family of games on which the algorithm requires an exponential number of iterations.

2. Parity games

A *parity game* is a tuple $G = (V, V_0, V_1, E, \Omega)$ where (V, E) forms a directed graph whose node set is partitioned into $V = V_0 \cup V_1$ with $V_0 \cap V_1 = \emptyset$, and $\Omega : V \to \mathbb{N}$ is the *priority function* that assigns to each node a natural number called the *priority* of the node. We assume the underlying graph to be total, *i.e.* for every $v \in V$ there is a $w \in W$ s.t. $(v, w) \in E$. In the following we will restrict ourselves to finite parity games. Let $ind(G)$ denote the number of different priorities in the game G.

We also use infix notation vEw instead of $(v, w) \in E$ and define the set of all *successors* of v as $vE := \{w \mid vEw\}$. The size $|G|$ of a parity game $G = (V, V_0, V_1, E, \Omega)$ is defined to be the cardinality of E, *i.e.* $|G| := |E|$; since we assume parity games to be total w.r.t. E, this seems to be a reasonable way to measure the size.

The game is played between two players called 0 and 1: starting in a node $v_0 \in V$, they construct an infinite path through the graph as follows. If the construction so far has yielded a finite sequence $v_0 \dots v_n$ and $v_n \in V_i$ then player i selects a $w \in v_n E$ and the play continues with the sequence $v_0 \dots v_n w$.

Every play has a unique winner given by the *parity* of the greatest priority that occurs infinitely often in a play. The winner of the play $v_0 v_1 v_2 \dots$ is player i iff $\max\{p \mid \forall j . \exists k \geq j : \Omega(v_k) = p\} \bmod 2 = i$.

A *strategy* for player i is a – possibly partial – function $\sigma : V^* V_i \to V$, s.t. for all sequences $v_0 \dots v_n$ with $v_{j+1} \in v_j E$ for all $j = 0, \dots, n-1$, and all $v_n \in V_i$ we have: $\sigma(v_0 \dots v_n) \in v_n E$. A play $v_0 v_1 \dots$ *conforms* to a strategy σ for player i if for all j we have: if $v_j \in V_i$ then $v_{j+1} = \sigma(v_0 \dots v_j)$. Intuitively, conforming to a strategy means to always make those choices that are prescribed by the strategy. A strategy σ for player i is a *winning strategy* in node v if player i wins every play that begins in v and conforms to σ. We say that player i *wins* the game G starting in v iff player i has a winning strategy for G starting in v.

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press

A strategy σ for player i is called *positional* if for all $v_0 \dots v_n \in V^*V_i$ and all $w_0 \dots w_m \in V^*V_i$ we have: if $v_n = w_m$ then $\sigma(v_0 \dots v_n) = \sigma(w_0 \dots w_m)$. That is, the value of the strategy on a finite path only depends on the last node on that path.

With G we associate two sets $W_0, W_1 \subseteq V$ such that W_i is the set of all nodes v s.t. player i wins the game G starting in v. Parity games enjoy determinacy meaning that for every node v in the game either $v \in W_0$ or $v \in W_1$ [1]. Moreover, they enjoy *positional* determinacy, *i.e.* for every $v \in W_i$, there is a positional winning strategy σ_v for player i. Furthermore, it is not difficult to show that, whenever player i has positional winning strategies σ_v for all $v \in U$ for some $U \subseteq V$, then there is also a single positional strategy σ that is winning for player i from every node in U.

The problem of solving a given parity game is to compute W_0 and W_1, and sometimes the corresponding positional winning strategies σ_0 and σ_1 for the players on their respective winning regions as well. We omit the computation of winning strategies in this paper.

Let $U \subseteq V$ and $i \in \{0, 1\}$. The i-attractor of U is the least set W s.t. $U \subseteq W$ and whenever $v \in V_i$ and $vE \cap W \neq \emptyset$, or $v \in V_{1-i}$ and $vE \subseteq W$ then $v \in W$. Hence, the i-attractor of U contains all nodes from which player i can move "towards" U and player $1-i$ must move "towards" U. The i-attractor of U is denoted by $Attr_i(G, U)$.

Let A be an arbitrary attractor set. The game $G \setminus A$ is defined to be the game restricted to the nodes $V \setminus A$, *i.e.* $G \setminus A := (V \setminus A, V_0 \setminus A, V_1 \setminus A, E \setminus (A \times V \cup V \times A), \Omega\,|_{V \setminus A})$. Note that A being an attractor ensures the required totality of $G \setminus A$.

3. The recursive algorithm

The recursive algorithm by Zielonka [14] decomposes the game at hand to smaller ones recursively by simultaneous induction on the number of priorities and the number of nodes in the game. In the base case, if the game is empty, the empty winning sets can be directly obtained. In the other cases winning sets and winning strategies can be assembled out of winning sets and strategies for smaller subgames and an attractor strategy for one of the players reaching the set of nodes with maximal priority in the game.

It is based on the observation that higher priorities in a parity game dominate all lower priorities, no matter how many there are. Let p be the highest priority occurring in the game G, let U be a non-empty set of nodes with priority p and let i be the parity of p. Now remove the i-attractor A of U and consider the so obtained subgame G'.

If player i wins the whole game G', then i also wins the whole game G: whenever player $1-i$ decides to visit A, player i's winning strategy would be to reach U. Then every play that visits A infinitely often has p as the highest priority occurring infinitely often, or otherwise it stays eventually in G' and hence is won by i.

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press

Otherwise, if player i does not win G' completely, *i.e.* player $1-i$ wins a non-empty subset W'_{1-i}, we know player $1-i$ also wins on W'_{1-i} w.r.t. G, because player i cannot force player $1-i$ to leave W'_{1-i}. Hence, we compute the $1-i$-attractor B of W'_{1-i} w.r.t. G, remove it as safe winning region for $1-i$ from the game and recursively solve the subgame $G\backslash$B.

The algorithm therefore can be specified as follows. In the original version of the algorithm, the non-empty subset U of nodes with priority p is the whole set of nodes with priority p. The freedom of choosing U can be seen as a rule for the recursive algorithm. Nevertheless, there is no indication of any benefits for practical solving as well as for the analysis of the lower bound by choosing a proper subset here. See Algorithm 1 for a pseudo-code specification.

Algorithm 1 Recursive Algorithm

procedure SOLVE(G)
  if $V_G = \emptyset$ **then**
    $(W_0, \sigma_0) \leftarrow (\emptyset, \bot)$
    $(W_1, \sigma_1) \leftarrow (\emptyset, \bot)$
    return (W_0, σ_0), (W_1, σ_1)
  else
    $p \leftarrow \max\{\Omega_G(v) \mid v \in V_G\}$
    $i \leftarrow p \bmod 2$
    $U \leftarrow$ non-empty subset of $\{v \in V_G \mid \Omega_G(v) = p\}$
    $\tau \leftarrow$ arbitrary strategy for player i on U
    $(A, \tau') \leftarrow Attr_i(G, U)$
    $(W'_0, \sigma'_0), (W'_1, \sigma'_1) \leftarrow$ SOLVE($G \setminus A$)
    if $W'_{1-i} = \emptyset$ **then**
      $(W_i, \sigma_i) \leftarrow (V_i, \sigma'_i \cup \tau \cup \tau')$
      $(W_{1-i}, \sigma_{1-i}) \leftarrow (\emptyset, \bot)$
      return (W_0, σ_0), (W_1, σ_1)
    else
      $(B, \varrho) \leftarrow Attr_{1-i}(G, W'_{1-i})$
      $(W''_0, \sigma''_0), (W''_1, \sigma''_1) \leftarrow$ SOLVE($G \setminus B$)
      $(W_i, \sigma_i) \leftarrow (W''_i, \sigma''_i)$
      $(W_{1-i}, \sigma_{1-i}) \leftarrow (W''_{1-i} \cup B, \sigma''_{1-i} \cup \varrho \cup \sigma'_{1-i})$
      return (W_0, σ_0), (W_1, σ_1)
    end if
  end if
end procedure

It is not hard to see that this algorithm is sound. Note that the correctness also implies that there are always positional winning strategies for parity games.

Theorem 3.1 ([14])**.** *Let G be a parity game.* SOLVE*(G) terminates and returns the winning sets with positional winning strategies for both players.*

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press

Proof. Let $G = (V, V_0, V_1, E, \Omega)$. We prove the claim by an induction on the number of nodes $|V|$. If G is empty, the algorithm obviously terminates and returns the correct winning sets and strategies.

For the the induction step, let $|V| > 0$. Let $p = \max\{\Omega(v) \mid v \in V\}$, $i = p \bmod 2$, U be a non-empty subset of $\{v \in V \mid \Omega(v) = p\}$, τ be an arbitrary strategy for player i on U and $(A, \tau') = Attr_i(G, U)$.

Consider the game $G' = G \setminus A$. Obviously, $|V_{G'}| < |V|$. By induction hypothesis, it follows that SOLVE(G') terminates and that it returns winning sets W'_0, W'_1 as well as positional winning strategies σ'_0, σ'_1 for G'.

If now $W'_{1-i} = \emptyset$, let $\sigma_i = \sigma'_i \cup \tau \cup \tau'$. We claim that player i indeed wins on V_i following strategy σ_i. Let π be a σ_i-conforming play in G; we distinguish whether π eventually stays in W'_i. If that is the case, it is obviously won by player i, because σ'_i is a winning strategy for i on W'_i by assumption. Otherwise, if π visits A infinitely often, then player i enforces infinitely many visits to U as well by the attractor strategy τ'. Hence, the highest priority occurring infinitely often is p. Therefore, SOLVE(G) terminates and returns the winning sets with positional winning strategies for both players.

Otherwise, if $W'_{1-i} \neq \emptyset$, let $(B, \varrho) = Attr_{1-i}(G, W'_{1-i})$. Consider the game $G'' = G \setminus B$. Obviously, $|V_{G''}| < |V|$. By induction hypothesis, it follows that SOLVE(G'') terminates and that it returns winning sets W''_0, W''_1 as well as positional winning strategies σ''_0, σ''_1 for G''.

First, we argue that σ''_i is still a winning strategy on W''_i in the game G for player i. Let π be a σ''_i-conforming play in G starting from a node in W''_i. It is impossible for player $1-i$ to escape from W''_i, as escaping directly to W''_{1-i} is a contradiction to W''_i being the i-winning set in G'' and escaping directly to B is impossible with B being an $1-i$-attractor.

Second, we argue that $\sigma_{1-i} = \sigma''_{1-i} \cup \varrho \cup \sigma'_{1-i}$ is a winning strategy on $W''_{1-i} \cup B$ for player $1-i$. Let π be a σ_{1-i}-conforming play in G; we distinguish whether π eventually stays in W''_{1-i}. If that is the case, it is obviously won by player $1-i$, because σ''_{1-i} is a winning strategy for $1-i$ on W''_{1-i} by assumption. Otherwise, if π visits B infinitely often, player $1-i$ enforces infinitely many visits to W'_{1-i} as well by the attractor strategy τ'; π even stays in W'_{1-i}, because W'_{1-i} is the $1-i$-winning set w.r.t. G' and player i cannot enforce a visit to A with A being an i-attractor in G. Hence, W'_{1-i} is won by player $1-i$ by assumption. Therefore, SOLVE(G) terminates and returns the winning sets with positional winning strategies for both players. □

For the analysis of the runtime complexity, let `Rec`(G) denote the total number of SOLVE-calls that are executed in order to solve a given parity game G. We fix the U-selection rule now that chooses the whole set of nodes with priority p.

A non-trivial upper bound can be easily derived: due to the fact that the number of different priorities is strictly reduced in the first recursive call and the number of nodes is strictly reduced in both recursive calls, the following recurrence $f(n, p)$, where n is an upper bound on the number of nodes and p is an upper bound on the

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press

number of different priorities, obviously describes an upper bound on the number of iterations that are required to solve a game with at most p different priorities and at most n nodes.

$$f(0,0) = 1$$
$$f(n+1, p+1) \leq f(n,p) + f(n, p+1).$$

Since $f(n,p) = n^p$ satisfies the recurrence, this yields the upper bound n^p. Similarly, it would be possible to derive that for arbitrary selection policies 2^n would be an upper bound on the number of iterations.

Theorem 3.2. *Let G be a parity game. Then $\texttt{Rec}(G) \in \mathcal{O}(|V_G|^{ind(G)})$ w.r.t. the whole-set rule and $\texttt{Rec}(G) \in \mathcal{O}(2^{|V_G|})$ for arbitrary policies.*

4. Exponential lower bound

We provide a concrete exponential lower bound on the number of iterations required by the recursive algorithm to solve parity games. The construction will yield the lower bound independently of the actual rule, because every "important" node in our family of games will have a different priority.

The games will be denoted by $G_n = (V_n, V_{n,0}, V_{n,1}, E_n, \Omega_n)$ and are of linear size. The sets of nodes are

$$V_n := \{a_1, \ldots, a_n, b_1, \ldots, b_n, c_0, \ldots, c_{n-1}, d_0, \ldots, d_{n-1}, e_0, \ldots, e_{n-1}\}.$$

The players, priorities and edges are described in Table 1. The game G_3 is depicted in Figure 1; nodes owned by player 0 are drawn as circles and nodes owned by player 1 are drawn as rectangles.

Table 1. The recursive lower bound game G_n.

Node	Player	Priority	Successors
a_i	$1 - (i \bmod 2)$	$1 - (i \bmod 2)$	$\{b_i, d_{i-1}\}$
b_i	$i \bmod 2$	$1 - (i \bmod 2)$	$\{a_i\} \cup (\{c_i\} \cap V_n)$
c_i	$1 - (i \bmod 2)$	$3i + 5$	$\{b_{i+1}, d_i\}$
d_i	$i \bmod 2$	$3i + 4$	$\{e_i\} \cup (\{d_{i-1}, d_{i+1}\} \cap V_n)$
e_i	$1 - (i \bmod 2)$	$3i + 3$	$\{b_{i+1}, d_i\}$

Fact 4.1. *The game G_n has $5 \times n$ nodes, $11 \times n - 3$ edges and $3 \times n + 2$ as the highest priority. In particular, $|G_n| = \mathcal{O}(n)$.*

Basically, solving the game G_n, requires G_{n-1} to be solved within the first recursive descent and G_{n-2} to be solved within the second recursive descent. Therefore, the number of recursion steps can be described by the *Fibonacci sequence*.

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press

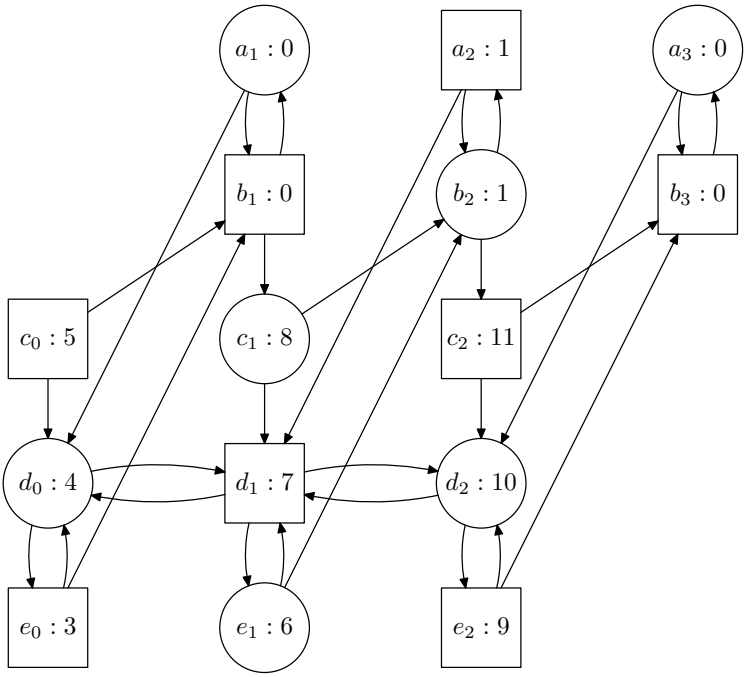


FIGURE 1. The recursive lower bound game G_3.

The Fibonacci sequence is a function $F : \mathbb{N} \to \mathbb{N}$ which is recursively defined as follows:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_{n+2} &= F_{n+1} + F_n \quad \text{for all } n. \end{aligned}$$

It is a well-known fact that $F \in \Omega\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$, and since $\frac{1+\sqrt{5}}{2} > 1$, this particularly implies that the Fibonacci sequence has exponential asymptotic behavior.

Lemma 4.2. *The game G_n is completely won by player $1 - (n \bmod 2)$.*

Proof. By induction on n. It is easy to see that G_1 is won by player 0 and G_2 is won by player 1. Let now $n > 2$ and $i = 1 - (n \bmod 2)$. We know by induction hypothesis that G_{n-2} is won by i.

Now attach the following strategy to the winning strategy for i on G_{n-2}: $\sigma(a_n) = b_n$, $\sigma(b_{n-1}) = c_{n-1}$, $\sigma(d_{n-1}) = e_{n-1}$ and $\sigma(c_{n-2}) = \sigma(e_{n-2}) = b_{n-1}$.

It is easy to see that a_n and b_n are won by player i. Hence, b_{n-1}, c_{n-1}, d_{n-1}, e_{n-1}, c_{n-2} and e_{n-2} are won by player i.

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press

Therefore $G_{n-2} \subseteq G_n$ is still won by player i, as moving to c_{n-2} from nodes in G_{n-2} results in a win of player i. Hence, also a_{n-1} and d_{n-2} are won by player i. □

We will now show that solving G_n requires at least F_n many iterations, which directly implies that the recursive algorithm requires exponentially many iterations on the family $(G_i)_{i>0}$.

Theorem 4.3. *For all $n > 0$, it holds that* $\texttt{Rec}(G_n) \geq F_n$.

Proof. By induction on n. For $n = 1, 2$ this is certainly true. For $n > 2$, we have to show that the solving computation w.r.t. G_n finally requires G_{n-1} and G_{n-2} to be solved in independent subcomputations:

The highest priority in G_n is $p = 3n + 2$, solely due to $U = \{c_{n-1}\}$, and its parity is $i := n \bmod 2$. The i-attractor of U is $A = U$, because the only node leading into c_{n-1} – namely b_{n-1} – is owned by $1-i$ and has more than one edge.

Let $G'_n = G_n \setminus A$. We will now show that sub-solving G'_n requires G_{n-1} to be solved.

- The highest priority in G'_n is $p' = 3n + 1$, solely due to $U' = \{d_{n-1}\}$, and its parity is $i' = 1 - (n \bmod 2)$. The i'-attractor of U' is $A' = \{a_n, b_n, d_{n-1}, e_{n-1}\}$, because the only node leading into A' – namely d_{n-2} – is owned by $1 - i'$ and has an edge not leading into A'.
  Now note that $G'_n \setminus A' = G_{n-1}$ which is to be computed next within this subcomputation.

Due to Lemma 4.2, G_{n-1} is completely won by player i, and A' is obviously won by player $1-i$; due to the fact that the only edge connecting G_{n-1} and A' in the game G_n is the edge from d_{n-2} to d_{n-1}, it is safe to conclude that solving G'_n indeed returns a partition into winning sets $W'_i = G_{n-1}$ and $W'_{1-i} = A'$.

Since W'_{1-i} is not empty, one has to compute the $1-i$-attractor of W'_{1-i} w.r.t. G_n, which is $B = A' \cup \{b_{n-1}, c_{n-1}, c_{n-2}, e_{n-2}\}$, because all nodes leading into B – namely a_{n-1}, b_{n-2} and d_{n-2} – are owned by player i and have edges not leading into B.

Let $G''_n = G_n \setminus B$. We will finally show that sub-solving G''_n requires G_{n-2} to be solved.

- The highest priority in G''_n is $p'' = 3n - 2$, solely due to $U'' = \{d_{n-2}\}$, and its parity is $i'' = n \bmod 2$. The i''-attractor of U'' is $A'' = \{a_{n-1}, d_{n-2}\}$, because the only other node leading into A'' – namely d_{n-3} – is owned by $1 - i''$ and has an edge not leading into A''.
  Now note that $G''_n \setminus A'' = G_{n-2}$ which is to be computed next within this subcomputation. □

5. Conclusion

We have shown that the recursive algorithm has exponential worst-case runtime complexity. Similar results hold true for almost all other parity game solving

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press

algorithms ([3, 7]) and therefore the recursive algorithm should not be considered to be of less use to the practical solving of parity games. In fact, the recursive algorithm is one of the best global parity game solving algorithms in practice [4].

References

[1] E.A. Emerson and C.S. Jutla, Tree automata, μ-calculus and determinacy, in *Proc. 32nd Symp. on Foundations of Computer Science*. San Juan, Puerto Rico, IEEE (1991) 368–377.

[2] E.A. Emerson, C.S. Jutla and A.P. Sistla, On model-checking for fragments of μ-calculus, in *Proc. 5th Conf. on Computer Aided Verification, CAV'93. Lect. Notes Comput. Sci.* **697** (1993) 385–396.

[3] O. Friedmann, An exponential lower bound for the parity game strategy improvement algorithm as we know it, in *Proc. of LICS* (2009) 145–156.

[4] O. Friedmann and M. Lange, Solving parity games in practice, in *Proc. of ATVA* (2009) 182–196.

[5] E. Grädel, W. Thomas and Th. Wilke Eds., *Automata, Logics, and Infinite Games. Lect. Notes Comput. Sci.* **2500** (2002).

[6] M. Jurdzinski, Deciding the winner in parity games is in $up \cap co - up$. *Inf. Process. Lett.* **68** (1998) 119–124.

[7] M. Jurdziński, Small progress measures for solving parity games, in *Proc. 17th Ann. Symp. on Theoretical Aspects of Computer Science, STACS'00*, edited by H. Reichel and S. Tison. *Lect. Notes Comput. Sci.* **1770** (2000) 290–301.

[8] M. Jurdziński, M. Paterson and U. Zwick, A deterministic subexponential algorithm for solving parity games, in *Proc. 17th Ann. ACM-SIAM Symp. on Discrete Algorithm, SODA'06*. ACM (2006) 117–123.

[9] S. Schewe, Solving parity games in big steps, in *Proc. FST TCS*. Springer-Verlag (2007).

[10] S. Schewe, An optimal strategy improvement algorithm for solving parity and payoff games, in *17th Annual Conference on Computer Science Logic (CSL)* (2008).

[11] P. Stevens and C. Stirling, Practical model-checking using games, in *Proc. 4th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'98*, edited by B. Steffen. *Lect. Notes Comput. Sci.* **1384** (1998) 85–101.

[12] C. Stirling, Local model checking games, in *Proc. 6th Conf. on Concurrency Theory, CONCUR'95. Lect. Notes Comput. Sci.* **962** (1995) 1–11.

[13] J. Vöge and M. Jurdziński, A discrete strategy improvement algorithm for solving parity games, in *Proc. 12th Int. Conf. on Computer Aided Verification, CAV'00. Lect. Notes Comput. Sci.* **1855** (2000) 202–215.

[14] W. Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoret. Comput. Sci.* **200** (1998) 135–183.

Communicated by Ch. Choffrut.
Received January 23, 2011. Accepted September 22, 2011.

https://doi.org/10.1051/ita/2011124 Published online by Cambridge University Press