Zielonka's Recursive Algorithm: dull, weak and solitaire games and tighter bounds

Maciej Gazda and Tim A.C. Willemse
Eindhoven University of Technology, The Netherlands

Dull, weak and nested solitaire games are important classes of parity games, capturing, among others, alternation-free μ-calculus and ECTL* model checking problems. These classes can be solved in polynomial time using dedicated algorithms. We investigate the complexity of Zielonka's *Recursive* algorithm for solving these special games, showing that the algorithm runs in $\mathcal{O}(d \cdot (n+m))$ on weak games, and, somewhat surprisingly, that it requires exponential time to solve dull games and (nested) solitaire games. For the latter classes, we provide a family of games $\mathscr{G}$, allowing us to establish a lower bound of $\Omega(2^{n/3})$. We show that an optimisation of Zielonka's algorithm permits solving games from all three classes in polynomial time. Moreover, we show that there is a family of (non-special) games $\mathscr{M}$ that permits us to establish a lower bound of $\Omega(2^{n/3})$, improving on the previous lower bound for the algorithm.

1 Introduction

Parity games [5, 15, 18] are infinite duration, two player games played on a finite directed graph. Each vertex in the graph is owned by one of the two players and vertices are assigned a priority. The game is played by moving a single token along the edges in the graph; the choice where to move next is decided by the player owning the vertex on which the token currently resides. A parity winning condition determines the winner of this infinite play; a vertex in the game is won by the player that can play such that, no matter how the opponent plays, every play from that vertex is won by her, and the winner of each vertex is uniquely determined [15]. From a practical point of view, parity games are interesting as they underpin verification, satisfiability and synthesis problems, see [4, 5, 1].

The simplicity of the gameplay is fiendishly deceptive. Despite continued effort, no polynomial algorithm for solving such games (*i.e.* computing the set of vertices won by each player) has been found. Solving a parity game is known to be in UP $\cap$ coUP [10], a class that neither precludes nor predicts the existence of a polynomial algorithm. In the past, non-trivial classes of parity games have been identified for which polynomial time solving algorithms exist. These classes include *weak* and *dull* games, which arise naturally from alternation-free modal μ-calculus model checking, see [3], and *nested solitaire* games which are obtained from *e.g.* the L_2 fragment of the modal μ-calculus, see [3, 6]. Weak and dull games can be solved in $\mathcal{O}(n+m)$, where n is the number of vertices and m is the number of edges, whereas (nested) solitaire games can be solved in $\mathcal{O}(\log(d) \cdot (n+m))$, where d is the number of different priorities in the game.

One of the most fundamental algorithms for solving parity games is Zielonka's *recursive algorithm* [18]. With a complexity of $\mathcal{O}(n^d)$, the algorithm is theoretically less attractive than *e.g.* Jurdziński's *small progress measures* algorithm [11], Schewe's *bigstep* algorithm [16] or the sub-exponential algorithm due to Jurdziński *et al.* [12]. However, as observed in [8], Zielonka's algorithm is particularly effective in practice, typically beating other algorithms. In view of this, one might therefore ask whether the algorithm is particularly apt at solving natural classes of games, taking advantage of the special struc-

Gabriele Puppis, Tiziano Villa (Eds.): Fourth International
Symposium on Games, Automata, Logics and Formal Verification
EPTCS 119, 2013, pp. 7–20, doi:10.4204/EPTCS.119.4

ture of these games. We explore this question by investigating the complexity of solving weak, dull and nested solitaire classes using Zielonka's algorithm. Our findings are as follows:

- in Section 4.1, we prove that Zielonka's algorithm solves weak games in polynomial time.
- in Section 4.2, we demonstrate that, somewhat surprisingly, Zielonka's algorithm is exponential on dull games and solitaire games.

The exponential lower bounds we obtain utilise a family of dull, solitaire games $\mathscr{G}^k$ with $3k$ vertices on which the algorithm requires 2^k iterations, allowing us to establish a lower bound of $\Omega(2^{n/3})$. This lower bound improves on previously documented lower bounds for this algorithm (*e.g.*, in [7] a lower bound of $\Omega(1.6^{n/5})$ is established).

In addition to the above complexity results we investigate whether the most common improvement of the algorithm permits it to run in polynomial time for all three special classes of games. That is, we prove in Section 5 that integrating Zielonka's algorithm in a *strongly connected component* decomposition algorithm, as suggested in [11, 8], permits solving all three classes in polynomial time. We analyse the complexity of the resulting algorithm for these three classes, showing that the optimised algorithm runs in $\mathscr{O}(n\cdot(n+m))$ for weak, dull and (nested) solitaire games. Note that these worst-case complexities are slightly worse than those for the dedicated algorithms, but that the applicability of the algorithm remains universal; *e.g.*, it is capable of solving arbitrary nestings of dull and solitaire games, and it does not depend on dedicated algorithms for detecting whether the game is special.

The optimised algorithm still requires exponential time on non-special games. For instance, Friedmann's games are resilient to all known optimisations. Drawing inspiration from our family of games $\mathscr{G}^k$ and the games of [7], we define a new family of games $\mathscr{M}^k$ containing $3k$ vertices, that is also resilient to all known optimisations and requires 2^k iterations of the algorithm. This again allows us to establish a lower bound of $\Omega(2^{n/3})$, also improving on the lower bound established by Friedmann in [7]. We experimentally compare the running time of the optimised algorithm on our games to those of Friedmann.

Outline. Before we present our results, we briefly describe parity games in Section 2 and Zielonka's algorithm in Section 3. Our runtime analysis of Zielonka's original algorithm on special games is presented in Section 4. We prove that an optimisation of the algorithm runs in polynomial time on special games in Section 5, and we prove that, in general, the optimisation's complexity is $\Omega(2^{n/3})$ in Section 6. In Section 7, we wrap up with some conclusions.

2 Parity Games

A parity game is an infinite duration game, played by players *odd*, denoted by $\Box$ and *even*, denoted by $\Diamond$, on a directed, finite graph. The game is formally defined as follows.

Definition 1 A pseudo parity game is a tuple $(V,E,\mathscr{P},(V_\Diamond,V_\Box))$, where

- V is a finite set of vertices, partitioned in a set $V_\Diamond$ of vertices owned by player $\Diamond$, and a set of vertices $V_\Box$ owned by player $\Box$,
- $E \subseteq V \times V$ is an edge relation,
- $\mathscr{P}$:$V \to \mathbb{N}$ is a priority function that assigns priorities to vertices, players.

We write $v \to w$ iff $(v,w) \in E$. A pseudo parity game is a *parity game* if the edge relation is total; *i.e.* for each $v \in V$ there is at least one $w \in V$ such that $(v,w) \in E$.

We depict (pseudo) parity games as graphs in which diamond-shaped vertices represent vertices owned by player $\Diamond$ and box-shaped vertices represent vertices owned by player $\Box$. Priorities, associated with vertices, are written inside vertices.

For a given (pseudo) parity game, we are often interested in the subgame that is obtained by restricting the game to a given set of vertices in some way. Formally, we define such subgames as follows.

Definition 2 Let $G = (V, E, \mathscr{P}, (V_\Diamond, V_\Box))$ be a (pseudo) parity game and let $A \subseteq V$ be an arbitrary non-empty set. The (pseudo) parity game $G \cap A$ is the tuple $(A, E \cap (A \times A), \mathscr{P}|_A, (V_\Diamond \cap A, V_\Box \cap A))$. The (pseudo) parity game $G \setminus A$ is defined as the game $G \cap (V \setminus A)$.

Throughout this section, assume that $G = (V, E, \mathscr{P}, (V_\Diamond, V_\Box))$ is an arbitrary pseudo parity game. Note that in general, whenever G is a *parity game* then it is not necessarily the case that the pseudo parity games $G \setminus A$ and $G \cap A$ are again parity games, as totality may not be preserved. However, in what follows, we only consider constructs in which these operations guarantee that totality *is* preserved.

The game G is said to be *strongly connected*, see [17], if for all pairs of vertices $v, w \in V$, we have $v \to^* w$ and $w \to^* v$, where $\to^*$ denotes the transitive closure of $\to$. A *strongly connected component* of G is a maximal set $C \subseteq V$ for which $G \cap C$ is strongly connected.

Lemma 1 Let $C \subseteq V$ be a strongly connected component. If G is a parity game, then so is $G \cap C$.

Henceforth, we assume that G is a parity game (*i.e.* its edge relation is total), and $\bigcirc$ denotes an arbitrary player. We write $\overline{\bigcirc}$ for $\bigcirc$'s opponent; *i.e.* $\overline{\Diamond} = \Box$ and $\overline{\Box} = \Diamond$. A sequence of vertices $v_1, \dots, v_n$ is a *path* if $v_m \to v_{m+1}$ for all $1 \leq m < n$. Infinite paths are defined in a similar manner. We write p_n to denote the n^{th} vertex in a path p.

A game starts by placing a token on a vertex $v \in V$. Players move the token indefinitely according to a simple rule: if the token is on some vertex $v \in V_\bigcirc$, player $\bigcirc$ gets to move the token to an adjacent vertex. The choice where to move the token next is determined by a partial function $\sigma{:}V^+ \to V$, called a *strategy*. Formally, a strategy σ for player $\bigcirc$ is a function satisfying that whenever it is defined for a finite path $v_1, \dots, v_n$, we have $\sigma(v_1, \dots, v_n) \in \{w \in V \mid v \to w\}$ and $v_n \in V_\bigcirc$. We say that an infinite path $v_1, v_2, \dots$ is *consistent* with a strategy σ for player $\bigcirc$ if for all finite prefixes $v_1, \dots, v_n$ for which σ is defined, we have $\sigma(v_1, \dots, v_n) = v_{n+1}$. An infinite path induced by strategies for both players is called a *play*.

The winner of a play is determined by the *parity* of the *highest* priority that occurs infinitely often on it: player $\Diamond$ wins if, and only if this priority is even. That is, we here consider *max* parity games. Note that, alternatively, one could demand that the *lowest* priority that occurs infinitely often along a play determines the winner; such games would be *min* parity games.

A strategy σ for player $\bigcirc$ is *winning* from a vertex v if and only if $\bigcirc$ is the winner of every play starting in v that is consistent with σ. A vertex is won by $\bigcirc$ if $\bigcirc$ has a winning strategy from that vertex. Note that parity games are *positionally determined*, meaning that a vertex is won by player $\bigcirc$ if $\bigcirc$ has a winning *positional strategy*: a strategy that determines where to move the token next based solely on the vertex on which the token currently resides. Such strategies can be represented by a function $\sigma{:}V_\bigcirc \to V$. A consequence of positional determinacy is that vertices are won by exactly one player [5]. *Solving* a parity game essentially is computing the partition $(W_\Diamond, W_\Box)$ of V of vertices won by player $\Diamond$ and player $\Box$, respectively. We say that a game G is a *paradise* for player $\bigcirc$ if all vertices in G are won by $\bigcirc$.

Special games. Parity games pop up in a variety of practical problems. These include model checking problems for fixed point logics [5], behavioural equivalence checking problems [4] and satisfiability and synthesis problems [1]. In many cases, the parity games underlying such problems are *special games*: parity games with a particular structure. We here consider three such special games: *weak*, *dull* and *nested solitaire* games; these classes have previously been studied in the literature, see *e.g.* [3] and the references therein. The definitions that we present here are taken from [3].

Weak games are game graphs in which the priorities along paths are monotonically descending (this is not to be confused with parity games with *weak parity* conditions). That is, for each pair of vertices v, w in the graph, if $v \to w$, then $\mathscr{P}(v) \geq \mathscr{P}(w)$. Such games correspond naturally to model checking problems for the alternation-free modal μ-calculus.

Definition 3 A parity game is *weak* if the priorities along all paths are descending.

Dedicated solvers for weak games can solve these in $\mathscr{O}(|V| + |E|)$. The algorithm that does so is rather straightforward. Since parity games are total, the set L of vertices with lowest priorities m are immediately won by player $\Diamond$ iff m is even. Any vertex in the game that can be *forced* to L by the player winning L can then be removed from the game; technically, this is achieved by computing the *attractor set* (see the next section) into L. What remains is another weak parity game which can be solved following the same steps until no vertex is left.

Weak games are closely related to dull games: the latter are game graphs in which all *basic cycles* in the graph are disjoint. A basic cycle is a finite path $v_1, \ldots, v_n$ for which $v_n \to v_1$ and no vertex v_i occurs twice on the path. An *even* cycle is a cycle in which the dominating (*i.e.* highest) priority is even; the cycle is an *odd* cycle if the dominating priority occurring on the cycle is odd.

Definition 4 A parity game is *dull* if even cycles and odd cycles are disjoint.

Note that every weak game is dull; every dull game, on the other hand, can be converted in linear time to a weak game by changing priorities only. This is achieved by assigning a priority that has the same parity as the highest priority present in a strongly connected component to all vertices in that component. This is harmless as each strongly connected component is either entirely even dominated or entirely odd dominated: if not, even cycles and odd cycles would overlap, contradicting the fact that the game is dull. Working bottom-up, it is straightforward to ensure that the priorities along the paths in the game are descending. As a result, dull games can also be solved in $\mathscr{O}(|V| + |E|)$ using the same algorithm as that for solving weak games. Dull games, too, can be obtained from alternation-free μ-calculus model checking problems, and they correspond naturally to the alternation-free fragment of LFP, see [2].

Solitaire games are games in which only one of the two players gets to make non-trivial choices where to play the token next; nested solitaire games generalise solitaire games to games in which both players may make non-trivial moves, but the interactions between both players is still restricted. Such games arise from model checking problems for the fragment L_2 of the modal μ-calculus, see [6], and they correspond with the solitaire fragment of LFP [3].

Definition 5 A parity game is *solitaire* if all non-trivial moves are made by a single player. The game is *nested solitaire* if each strongly connected component induces a solitaire game.

Nested solitaire games can be solved in $\mathscr{O}(\log(d) \cdot (|V| + |E|))$, see [9], although most implementations use a somewhat less optimal implementation that runs in $\mathscr{O}(d \cdot (|V| + |E|))$, see [3]. The latter algorithm works by computing the strongly connected components of a graph and start searching for an even cycle if all non-trivial moves in the component are made by player $\Diamond$ and an odd cycle otherwise.

Computing whether there is an even cycle (resp. odd cycle) can be done in $\mathcal{O}(\log(d) \cdot (|V| + |E|))$ using the techniques of [14] or, in $\mathcal{O}(d \cdot (|V| + |E|))$ by repeatedly conducting a depth-first search, starting at the lowest even priority in the component. Clearly, in a component where only player $\Diamond$ gets to make non-trivial moves, all vertices are won by player $\Diamond$ iff an even cycle is found. Iteratively solving the final strongly connected component and removing it together with the attractor for the winner of this component solves entire nested solitaire games.

3 Zielonka's Recursive Algorithm

Throughout this section, we assume G is a fixed parity game $(V, E, \mathscr{P}, (V_\Diamond, V_\Box))$, and $\bigcirc$ is an arbitrary player.

Zielonka's algorithm for solving parity games, listed as Algorithm 1, is a divide and conquer algorithm. It constructs winning regions for both players out of the solution of subgames with fewer different priorities and fewer vertices. It removes the vertices with the highest priority from the game, together with all vertices *attracted* to this set of vertices. Attractor sets are formally defined as follows.

Definition 6 The $\bigcirc$-*attractor* into a set $U \subseteq V$, denoted $Attr_\bigcirc(U)$, is defined inductively as follows:

$$\begin{array}{lcl} Attr^0_\bigcirc(U) & = & U \\ Attr^{n+1}_\bigcirc(U) & = & Attr^n_\bigcirc(U) \\ & \cup & \{u \in V_\bigcirc \mid \exists v \in Attr^n_\bigcirc(U):\ u \to v\} \\ & \cup & \{u \in V_{\bar{\bigcirc}} \mid \forall v \in V : u \to v \implies v \in Attr^n_\bigcirc(U)\} \\ Attr_\bigcirc(U) & = & \bigcup\limits_{i \geq 0} Attr^i_\bigcirc(U) \end{array}$$

If needed for clarity, we write $Attr^G_\bigcirc(U)$ to indicate that the $\bigcirc$-attractor is computed in game graph G.

The lemma below states that whenever attractor sets are removed from a parity game, totality is preserved.

Lemma 2 Let $A = Attr_\bigcirc(U) \subseteq V$ be an arbitrary attractor set. If G is a parity game, then so is $G \setminus A$.

The correctness of Zielonka's algorithm hinges on the fact that higher priorities in the game dominate lower priorities, and that any forced revisit of these higher priorities is beneficial to the player aligning with the parity of the priority. For a detailed explanation of the algorithm and proof of its correctness, we refer to [18, 7].

4 Solving Special Games

Zielonka's algorithm is quite competitive on parity games that stem from practical verification problems [8, 13], often beating algorithms with better worst-case running time. While Zielonka's original algorithm is known to run in exponential time on games defined by Friedmann [7], its behaviour on special parity games has never before been studied. It might just be the case that this algorithm is particularly apt to solve such games. We partly confirm this hypothesis in Section 4.1 by proving that the algorithm indeed runs in polynomial time on weak games. Somewhat surprisingly, however, we also establish that Zielonka's algorithm performs poorly when solving dull and (nested) solitaire games, see Section 4.2.

Algorithm 1 Zielonka's Algorithm

1: **function** ZIELONKA(G)
2:     **if** $V = \emptyset$ **then**
3:         $(W_\Diamond, W_\Box) \leftarrow (\emptyset, \emptyset)$
4:     **else**
5:         $m \leftarrow \max\{\mathscr{P}(v) \mid v \in V\}$
6:         **if** $m \bmod 2 = 0$ **then** $p \leftarrow \Diamond$ **else** $p \leftarrow \Box$ **end if**
7:         $U \leftarrow \{v \in V \mid \mathscr{P}(v) = m\}$
8:         $A \leftarrow Attr_p(U)$
9:         $(W'_\Diamond, W'_\Box) \leftarrow$ ZIELONKA$(G \setminus A)$
10:         **if** $W'_{\bar{p}} = \emptyset$ **then**
11:             $(W_p, W_{\bar{p}}) \leftarrow (A \cup W'_p, \emptyset)$
12:         **else**
13:             $B \leftarrow Attr_{\bar{p}}(W'_{\bar{p}})$
14:             $(W'_\Diamond, W'_\Box) \leftarrow$ ZIELONKA$(G \setminus B)$
15:             $(W_p, W_{\bar{p}}) \leftarrow (W'_p, W'_{\bar{p}} \cup B)$
16:         **end if**
17:     **end if**
18:     **return** $(W_\Diamond, W_\Box)$
19: **end function**

4.1 Weak Games

We start with a crucial observation —namely, that for weak games, ZIELONKA solves a paradise in polynomial time— which permits us to prove that solving weak games can be done in polynomial time using ZIELONKA. The proof of the latter, formalised as Proposition 1, depends on three observations, which we first prove in isolation in the following lemma.

Lemma 3 Let $G = (V, E, \mathscr{P}, (V_\Diamond, V_\Box))$ be a weak parity game. Suppose G is a paradise for player $\bigcirc$; *i.e.*, G is won entirely by $\bigcirc$. Then ZIELONKA, applied to G, has the following properties:

1. in the first recursive call in line 9, the argument $G \setminus A$ is also a paradise for player $\bigcirc$.
2. if the second recursive call (line 14) is reached, then its argument $(G \setminus B)$ is the empty set.
3. edges that are used in the computation of attractor sets (lines 8 and 13) are not considered in subroutines.

Proof: We prove all three statements below.

1. Observe that $A = Attr_p(U) = U$, since, in a weak game, no vertex with lower priority has an edge to a vertex in U. In particular, the subgame $G \setminus A$ is $\bar{\bigcirc}$-closed, and hence must be won entirely by $\bigcirc$, if G is a $\bigcirc$-paradise.
2. The second recursive call can be invoked only if $W'_{\bar{p}} \neq \emptyset$. From the above considerations we know that this implies $\bar{p} = \bigcirc$, and $W'_{\bar{p}} = G \setminus A$ is a paradise for $\bigcirc$. We also have $G = W'_{\bar{p}} \cup A$. Since every game staying in A would be losing for $\bigcirc$, it must be the case that $A \subseteq Attr_{\bar{p}}(W'_{\bar{p}})$. But then $B = Attr_{\bar{p}}(W'_{\bar{p}}) = G$, and hence $G \setminus B = \emptyset$.
3. Edges that are considered in the computation of both $Attr_p(U)$ (line 8) and $Attr_{\bar{p}}(W'_{\bar{p}})$ have sources only in U; since no vertices from U are included in the subgame considered in the first recursive

call, and the second call can only take the empty set as an argument. Therefore, these edges will not be considered in the subroutines.

□

Proposition 1 Let $G = (V, E, \mathscr{P}, (V_\Diamond, V_\Box))$ be a weak parity game. Suppose G is a paradise for player $\bigcirc$; *i.e.*, G is won entirely by $\bigcirc$. Then ZIELONKA runs in $\mathscr{O}(|V| + |E|)$.

Proof: We analyse the running time $T(k)$ of ZIELONKA when it is called on a subgame G_k of G with exactly k priorities. Let v_k denote the number of nodes with the highest priority in G_k, and with e_k the number of edges that are considered in the attractor computations (lines 8 and 13) on G_k.

If we assume that the representation of the game has some built-in functionality that allows us to inspect the nodes in the order of priority, then the time required to execute the specific lines of the procedure can be bounded as follows:

- line 7: $c \cdot v_k$ for some constant c
- lines 8 and 13 in total: $c \cdot e_k$ for some constant c
- the remaining lines: z for some constant $z \in \mathbb{N}$

We obtain:

$$\begin{array}{lcl} T(k) & \leq & c \cdot (v_i + e_i + z) + T(k-1) \\ T(k) & \leq & \sum_{i=1}^{k} c \cdot (v_i + e_i + z) \\ T(k) & \leq & c \cdot (\sum_{i=1}^{k} v_i + \sum_{i=1}^{k} e_i + \sum_{i=1}^{k} z) \end{array}$$

Let d denote the total number of priorities occurring in G. Observe that from Lemma 3, we have:

$$\sum_{i=1}^{d} v_i = |V| \text{ and } \sum_{i=1}^{d} e_i = |E|$$

The total execution time of ZIELONKA on G can be bounded by:

$$T(d,V,E) \leq c \cdot (|V| + |E| + \mathscr{O}(d))$$

Hence we obtain $T(d,V,E) = \mathscr{O}(|V| + |E|)$. □

The above proposition is used in our main theorem below to prove that solving weak games using ZIELONKA can be done in polynomial time: each second recursive call to ZIELONKA will effectively be issued on a paradise or an empty game. By proposition 1, we know that ZIELONKA will solve a paradise in linear time.

Theorem 1 ZIELONKA requires $\mathscr{O}(d \cdot (|V| + |E|))$ to solve weak games with d different priorities, $|V|$ vertices and $|E|$ edges.

Proof: The key observation is that ZIELONKA, upon entering the second recursive call in line 14 is invoked on a game that is a paradise. Consider the set of vertices $V \setminus B$ of the game G at that point. It contains the entire set W'_p, and possibly a subset of U. Now, if player $\overline{p}$ could force a play in a node $v \in W'_p$ to $W'_{\overline{p}}$, it could be done only via set U. But this would violate the weakness property. Player p has a winning strategy on $V \setminus B$, which combines the existing strategy on W'_p and if necessary any strategy on U (because whenever a play visits U infinitely often, it is won by p). Thus, the game $G \setminus B$ that is then considered is a p-paradise.

As a result, by Proposition 1, the game $G \setminus B$ is solved in $\mathcal{O}(|V|+|E|)$. Based on these observations, we obtain the following recurrence for ZIELONKA:

$$\begin{aligned} T(0,V,E) &\leq \mathcal{O}(1) \\ T(d+1,V,E) &\leq T(d,V,E)+\mathcal{O}(|V|)+\mathcal{O}(|E|) \end{aligned}$$

Thus, a non-trivial upper bound on the complexity is $\mathcal{O}(d \cdot (|V|+|E|))$. $\square$

Next, we show this bound is tight. Consider the family of parity games $\mathscr{W}^n = (V^n, E^v, \mathscr{P}^n, (V^n_\Diamond, V^n_\Box))$, where priorities and edges are defined in Table 1 and V^n is defined as $V^n = \{v_1, \dots, v_{2n}, u_0, u_1\}$.

Table 1: The family $\mathscr{W}$ of games; $1 \leq i \leq n$.

Vertex	Player	Priority	Successors
v_i	$\Diamond$	$i+2$	$\{v_{i-1}, v_{n+i}\} \cup \{u_0 \mid i=1\}$
v_{n+i}	$\Box$	$i+2$	$\{v_i, v_{n+i-1}\} \cup \{u_1 \mid i=1\}$
u_0	$\Diamond$	0	$\{u_0\}$
u_1	$\Box$	1	$\{u_1\}$

The game $\mathscr{W}^4$ is depicted in Figure 1. The family $\mathscr{W}$ has the following characteristics.

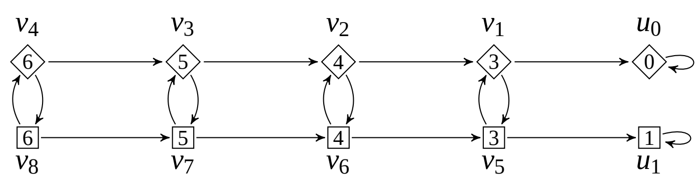

Figure 1: The game $\mathscr{W}^4$.

Proposition 2 The game $\mathscr{W}^n$ is of size linear in n; *i.e.*, $|\mathscr{W}^n| = \mathcal{O}(n)$, it contains $2n+2$ vertices, $4n+2$ edges and $n+2$ different priorities. Moreover, the game $\mathscr{W}^n$ is a weak game.

Lemma 4 In the game $\mathscr{W}^n$, vertices $\{u_0, v_1, \dots, v_n\}$ are won by player $\Diamond$, whereas vertices $\{u_1, v_{n+1}, \dots, v_{2n}\}$ are won by player $\Box$.

Proof: Follows from the fact that, for $0 \leq j < n-1$, the strategy $v_{n-j} \to v_{n-j-1}$, $v_1 \to u_0$ and $u_0 \to u_0$ is winning for player $\Diamond$ for the set of vertices $\{u_0, v_1 1, \dots, v_n\}$ and the strategy $v_{2n-j} \to v_{2n-j-1}$, $v_{n+1} \to u_1$ and $u_1 \to u_1$ is winning for player $\Box$ from the set of vertices $\{u_1, v_{n+1}, \dots, v_{2n}\}$. $\square$

We next analyse the runtime of Zielonka's algorithm on the family $\mathscr{W}$. Let a_n be defined through the following recurrence relation:

$$\begin{aligned} a_0 &= 1 \\ a_{n+1} &= a_n + n + 1 \end{aligned}$$

Observe that the function $\frac{1}{2}n^2$ approximates a_n from below. The proposition below states that solving the family $\mathscr{W}$ of weak parity games requires a quadratic number of recursions.

Proposition 3 Solving $\mathscr{W}^n$, for $n > 0$, requires at least a_n calls to ZIELONKA.

Proof: Follows from the observation that each game $\mathscr{W}^{n+1}$ involves:

1. a first recursive call to ZIELONKA for solving the game $\mathscr{W}^n$.

2. a second recursive call to ZIELONKA for solving either $\mathscr{W}^n \setminus \{v_{2n}, \ldots, v_{n+1}, u_1\}$ or $\mathscr{W}^n \setminus \{v_n, \ldots, v_1, u_0\}$; both require $n+1$ recursive calls to ZIELONKA.

$\square$

Theorem 2 Solving weak games using ZIELONKA requires $\Theta(d \cdot (|V| + |E|))$.

Note that this complexity is a factor d worse than that of the dedicated algorithm. For practical problems such as when solving parity games that come from model checking problems d is relatively small; we expect that for such cases, the difference between the dedicated algorithm and Zielonka's algorithm to be small.

4.2 Dull and Nested Solitaire Games

We next prove that dull games and (nested) solitaire require exponential time to solve using ZIELONKA. Given that dull games can be converted to weak games in linear time, and that Zielonka solves weak games in polynomial time, this may be unexpected.

Our focus is on solitaire games first. We construct a family of parity games $\mathscr{G}^n = (V^n, E^n, \mathscr{P}^n, (V^n_\Diamond, V^n_\Box))$ with vertices $V^n = \{v_0, \ldots, v_{2n-1}, u_1, \ldots u_n\}$. All vertices belong to player $\Diamond$; that is, $V^n_\Diamond = V^n$ and $V^n_\Box = \emptyset$. The priorities and the edges are described by Table 2.

Table 2: The family $\mathscr{G}$ of games; $1 \le i < 2n, 1 \le j \le n$.

Vertex	Priority	Successors
v_i	$i+2$	$\{v_{i-1}\}$
v_0	2	$\{v_0\}$
u_j	1	$\{u_j, v_{2j-1}\}$

Proposition 4 The game $\mathscr{G}^n$ is of size linear in n; *i.e.* $|\mathscr{G}^n| = \mathscr{O}(n)$, it has $3n$ vertices, $4n$ edges and $2n+1$ different priorities. Moreover, the game $\mathscr{G}^n$ is a (nested) solitaire game.

The game $\mathscr{G}^3$ is depicted in Figure 2. Observe that in this game, vertex v_5 has the maximal priority and that this priority is odd. This means that Zielonka's algorithm will compute the odd-attractor to v_5 in line 8 of the algorithm, *i.e.* $Attr_\Box(\{v_5\}) = \{v_5\}$. We can generalise this observation for arbitrary game $\mathscr{G}^n$: in such a game, $Attr_\Box(\{v_{2n-1}\}) = \{v_{2n-1}\}$. Henceforth, we denote the subgame $\mathscr{G}^n \setminus \{v_{2n-1}\}$ by $\mathscr{G}^{n,-}$.

Lemma 5 The game $\mathscr{G}^n$ is won by player $\Diamond$. In the game $\mathscr{G}^{n,-}$, all vertices except for vertex u_n, are won by player $\Diamond$.

Proof: The fact that $\mathscr{G}^n$ is won by player $\Diamond$ follows immediately from the strategy $\sigma: V^n \to V^n$, defined as $\sigma(v_i) = v_{i-1}$ for all $1 \le i < 2n$, $\sigma(u_i) = v_{2i-1}$ for all $i \le n$ and $\sigma(v_0) = v_0$, which is winning for player $\Diamond$. For the game $\mathscr{G}^{n,-}$, a strategy σ' can be used that is as strategy σ for all vertices $v \neq u_n$; for vertex u_n, we are forced to choose $\sigma'(u_n) = u_n$, since u_n is the sole successor of u_n in $\mathscr{G}^{n,-}$. Since the priority of u_n is odd, the vertex is won by player $\Box$. $\square$

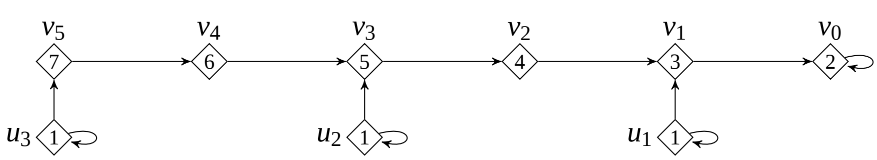

Figure 2: The game $\mathscr{G}^3$.

We now proceed to the runtime of Zielonka's algorithm on the family $\mathscr{G}$.

Proposition 5 Solving $\mathscr{G}^n$, for $n > 0$, requires at least 2^n calls to ZIELONKA.

Proof: Solving the game $\mathscr{G}^1$ requires at least one call to ZIELONKA.

Consider the game $\mathscr{G}^n$, for $n > 1$. Observe that ZIELONKA is invoked recursively on the game $\mathscr{G}^{n,-}$ in the first recursion on line 9. We focus on solving the latter game.

The vertex with the highest priority in $\mathscr{G}^{n,-}$ is $v_{2(n-1)}$. Observe that $A = Attr_{\Box}(\{v_{2(n-1)}\}) = \{v_{2(n-1)}\}$. Note that $\mathscr{G}^{n,-} \setminus A$ contains $\mathscr{G}^{n-1}$ as a separate subgame. The first recursive call in solving $\mathscr{G}^{n,-}$ will therefore also solve the subgame $\mathscr{G}^{n-1}$.

Next, observe that u_n (and u_n alone) is won by player $\Box$, see Lemma 5. We therefore need to compute $B = Attr_{\Diamond}(\{u_n\}) = \{u_n\}$. Now, note that $\mathscr{G}^{n,-} \setminus B$ subsumes the subgame $\mathscr{G}^{n-1}$, which is a separate game in $\mathscr{G}^{n,-} \setminus B$. Therefore, also the second recursive call to ZIELONKA involves solving the subgame $\mathscr{G}^{n-1}$.
$\Box$

The lower bound on the number of iterations for ZIELONKA is thus exponential in the number of vertices.

Theorem 3 Solving (nested) solitaire games using ZIELONKA requires $\Omega(2^{|V|/3})$.

We note that this improves on the bounds of $\Omega(1.6^{|V|/5})$ established by Friedmann. Being structurally more complex, however, his games are robust to typical (currently known) improvements to Zielonka's algorithm such as the one presented in the next section (although this is not mentioned or proved in [7]). Still, we feel that the simplicity of our family $\mathscr{G}$ fosters a better understanding of the algorithm.

Observe that the family $\mathscr{G}$ is also a family of dull games. As a result, we immediately have the following theorem.

Theorem 4 Solving dull games using ZIELONKA requires $\Omega(2^{|V|/3})$.

5 Recursively Solving Special Games in Polynomial Time

The $\mathscr{G}$ family of games of the previous section are easily solved when preprocessing the games using priority propagation and self-loop elimination. However, it is straightforward to make the family robust to such heuristics by duplicating the vertices that have odd priority, effectively creating odd loops that are not detected by such preprocessing steps. In a similar vein, the commonly suggested optimisation to use a strongly connected component decomposition as a preprocessing step can be shown to be insufficient to solve (nested) solitaire games. The family $\mathscr{G}$ can easily be made robust to this preprocessing step: by adding edges from v_0 to all u_i, each game in $\mathscr{G}$ becomes a single SCC.

In this section, we investigate the complexity of a tight integration of a strongly connected component decomposition and Zielonka's algorithm, as suggested by *e.g.* [11, 8]. By decomposing the game each time Zielonka is invoked, large SCCs are broken down in smaller SCCs, potentially increasing the effectiveness of the optimisation. The resulting algorithm is listed as Algorithm 2.

We will need the following lemma:

Lemma 6 If algorithm 2 is invoked on a game that is either dull or (nested) solitaire, then in the entire recursion tree all second recursive calls (line 16) are trivial (with empty set as an argument).

Proof: In case of dull games, since game H is a connected component, each of its subgames is won by player corresponding to $m \bmod 2$, namely p. Hence after the line 11 is executed, we obtain $W'_p = H \setminus A$ and $W'_{\bar{p}} = \emptyset$. The second recursive call will therefore never be invoked.

Algorithm 2 Optimised Zielonka's Algorithm

function ZIELONKA_SCC(G)
    $(W^G_\Diamond, W^G_\Box) \leftarrow (\emptyset, \emptyset)$
    if $V \neq \emptyset$ **then**
        $\mathscr{S} :=$ SCC_GRAPH_DECOMPOSITION(G)
        for each final SCC $C \in \mathscr{S}$ **do**
            $H \leftarrow G \cap C$
            $m \leftarrow \max\{\mathscr{P}(v) \mid v \in C\}$
            if $m \bmod 2 = 0$ **then** $p \leftarrow \Diamond$ **else** $p \leftarrow \Box$ **end if**
            $U \leftarrow \{v \in C \mid \mathscr{P}(v) = m\}$
            $A \leftarrow Attr^H_p(U)$
            $(W'_\Diamond, W'_\Box) \leftarrow$ ZIELONKA_SCC$(H \setminus A)$
            if $W'_{\bar{p}} = \emptyset$ **then**
                $(W_p, W_{\bar{p}}) \leftarrow (A \cup W'_p, \emptyset)$
            else
                $B \leftarrow Attr^H_{\bar{p}}(W'_{\bar{p}})$
                $(W'_\Diamond, W'_\Box) \leftarrow$ ZIELONKA_SCC$(H \setminus B)$
                $(W_p, W_{\bar{p}}) \leftarrow (W'_p, W'_{\bar{p}} \cup B)$
            end if
            $(W^G_\Diamond, W^G_\Box) \leftarrow (W^G_\Diamond \cup Attr^G_\Diamond(W_\Diamond), W^G_\Box \cup Attr^G_\Box(W_\Box))$
            $\mathscr{S} :=$ SCC_GRAPH_DECOMPOSITION$(G \setminus (W^G_\Diamond \cup W^G_\Box))$
        end for
    end if
    return $(W^G_\Diamond, W^G_\Box)$
end function

Now assume that the game is solitaire and owned by player q. If $p = q$, then $Attr^H_p(U) = H$ (the game is p-owned, and strongly connected), and the second call is not invoked at all. Otherwise, the second call is invoked only if $W'_{\bar{p}} \neq \emptyset$. But then $B = Attr^H_{\bar{p}}(W'_{\bar{p}}) = Attr^H_q(W'_{\bar{p}}) = H$ (the game is owned by $\bar{p}$, and strongly connected), and $H \setminus B = \emptyset$. $\Box$

We will now prove that the optimisation suffices to solve special parity games in polynomial time.

Theorem 5 Algorithm 2 solves dull and (nested) solitaire games in $\mathscr{O}(|V| \cdot (|V| + |E|))$ time.

Proof: Let $\#\mathsf{for}(V)$ denote the total number of iterations of the for loop in the entire recursion tree. Observe that the total execution time of ZIELONKA_SCC can be bounded from above as follows:

$$T(V,E) = \mathscr{O}(\#\mathsf{for}(V) \cdot (|V| + |E|))$$

Indeed, every iteration of the loop (not counting the iterations in subroutines) contributes a maximal factor of $\mathscr{O}(|V| + |E|)$ running time, which results from the attractor computation and SCC decomposition.

We will use subscripts for the values of the algorithm variables in iteration $i \in \{1, \ldots, k\}$, e.g. the value of variable C in iteration i is C_i. Furthermore, by V_i we will denote the set of vertices in the subgame considered in the first recursive call, i.e. $V_i = C_i \setminus A_i$.

We will show that $\#\mathsf{for}(V) \leq |V|$. We have:

$$\begin{array}{lcll} \#\mathsf{for}(V) & \leq & 1 & \text{for } |V| \leq 1 \\ \#\mathsf{for}(V) & \leq & \#\mathsf{for}(V_1) + \cdots + \#\mathsf{for}(V_k) + k & \text{for } |V| > 1 \end{array} \qquad (*)$$

In the second inequality, k is the total number of bottom SCCs considered in line 5. Each of these SCCs may give rise to a recursive call (at most one, see Lemma 6). This recursive call contributes in turn $\#\mathsf{for}(V_i)$ iterations.

We proceed to show $\#\mathsf{for}(V) \leq |V|$ by induction on $|V|$. The base holds immediately from the first inequality. Now assume that $\#\mathsf{for}(V) \leq |V|$ for $|V| < m$.

Obviously $|C_1| + \cdots + |C_k| \leq |V|$. Observe that in every iteration i the set A_i is nonempty, therefore $V_i < C_i$. Therefore $|V_1| + \cdots + |V_k| \leq |V| - k$, or equivalently $|V_1| + \cdots + |V_k| + k \leq |V|$.

Applying the induction hypothesis in the right-hand side of (*) yields $\#\mathsf{for}(V) \leq |V_1| + \cdots + |V_k| + k$, and due to the above observation we finally obtain $\#\mathsf{for}(V) \leq |V|$. □

The above upper bound is slower by a factor V compared to the dedicated algorithms for solving weak and dull games. For nested solitaire games, the optimised recursive algorithm has an above upper bound comparable to that of standard dedicated algorithms for nested solitaire games when the number of different priorities is of $\mathscr{O}(V)$, and it is a factor $V/\log(d)$ slower compared to the most efficient algorithm for solving nested solitaire games.

6 A Tighter Exponential Bound for Zielonka's Optimised Algorithm

In view of the findings of the previous section, it seems beneficial to always integrate Zielonka's recursive algorithm with SCC decomposition. Observe that the family of games we used to establish the lower bound of $\Omega(2^{|V|/3})$ in Section 4 does not permit us to prove the same lower bound for the optimised algorithm. As a result, the current best known lower bound for the algorithm is still $\Omega(1.6^{|V|/5})$. In this section, we show that the complexity of the optimised algorithm is actually also $\Omega(2^{|V|/3})$. The family of games we construct is, like Friedmann's family, resilient to all optimisations we are aware of.

Let $\mathscr{M}^n = (V^n, E^n, \mathscr{P}^n, (V^n_\Diamond, V^n_\Box))$, for $n \geq 1$ be a family of parity games with set of vertices $V^n = \{v_i, u_i, w_i \mid 1 \leq i \leq n\}$. The sets $V^n_\Diamond$ and $V^n_\Box$, the priority function $\mathscr{P}^n$ and the set of edges are described by Table 3. We depict the game $\mathscr{M}^4$ in Figure 3.

Table 3: The family $\mathscr{M}$ of games; $1 \leq i \leq n$.

Vertex	Player	Priority	Successors
v_i	$\Box$ iff $i \bmod 2 = 0$	$i+1$	$\{u_i\} \cup \{v_{i+1} \mid i < n\}$
u_i	$\Box$ iff $i \bmod 2 = 0$	$i \bmod 2$	$\{w_i\} \cup \{v_{i+1} \mid i < n\}$
w_i	$\Diamond$ iff $i \bmod 2 = 0$	$i \bmod 2$	$\{u_i\} \cup \{w_{i-1} \mid 1 < i\}$

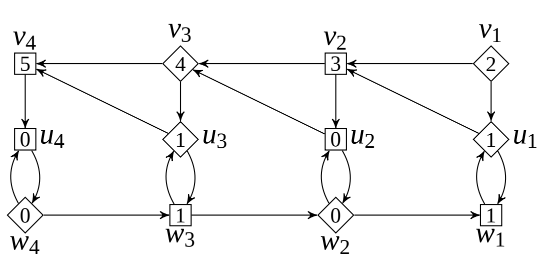

Figure 3: The game $\mathscr{M}^4$.

Proposition 6 The game $\mathscr{M}^n$ is won entirely by player $\Diamond$ for even n and entirely by player $\Box$ for odd n.

Theorem 6 Solving $\mathscr{M}^n$ using either ZIELONKA or ZIELONKA_SCC requires $\Omega(2^{|V|/3})$ time.

Proof: The proof is similar to Prop. 5; we can show that the game $\mathscr{M}^n$ requires 2^n calls to either ZIELONKA or ZIELONKA_SCC. The only significant difference in case of ZIELONKA_SCC is that the game may be potentially simplified in line 4 of Alg.2. However, each game $\mathscr{M}^n$ constitutes a strongly connected subgame, and therefore will not be decomposed. $\square$

We compared the performance of the PGSolver tool, a publicly available tool that contains an implementation of the optimised recursive algorithm, on the family $\mathscr{M}$ to that of Friedmann's family of games (denoted with $\mathscr{F}$), see Figure 4. The figure plots the number of vertices (horizontal axis) and the time required to solve the games (vertical log scale axis), clearly illustrating that $\mathscr{M}$ games are harder.

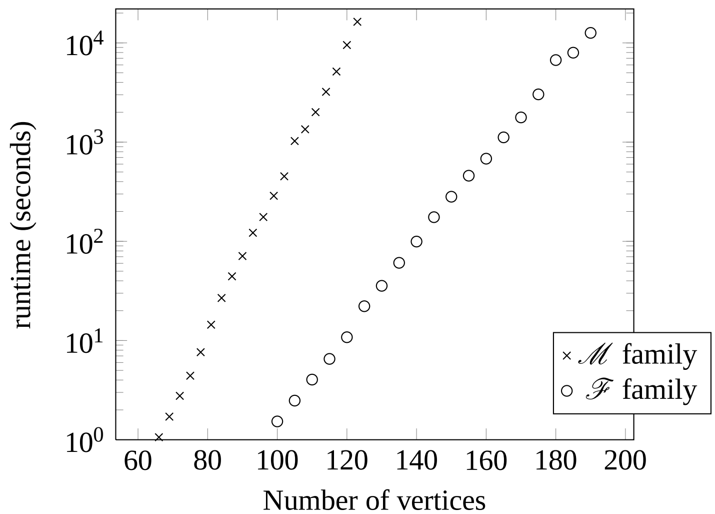

Figure 4: Runtime of the optimised recursive algorithm (vertical log scale axis) in seconds versus number of vertices of the games (horizontal axis).

7 Conclusions

We explored the complexity of solving special parity games using Zielonka's recursive algorithm, proving that weak games are solved in polynomial time and dull and nested solitaire games require exponential time. The family of games $\mathscr{G}$ we used to prove the exponential lower bounds in addition tighten the lower bound to $\Omega(2^{|V|/3})$ for the original algorithm by Zielonka.

We show that a standard optimisation of the algorithm permits solving all three classes of games in polynomial time. The technique used in the optimisation (a tight integration of a strongly connected component decomposition and Zielonka's algorithm) has been previously implemented in [8] and was observed to work well in practice. Our results provide theoretical explanations for these observations.

We furthermore studied the lower bounds of Zielonka's algorithm *with* optimisation. In the last section, we improve on Friedmann's lower bound and arrive at a lower bound of $\Omega(2^{|V|/3})$ for the optimised algorithm. For this, we used a family of games $\mathscr{M}$ for which we drew inspiration from the family $\mathscr{G}$ and the games defined in [7]. We believe that an additional advantage of the families of games $\mathscr{G}$ and $\mathscr{M}$ we defined in this paper over Friedmann's games lies in their (structural) simplicity.

Our complexity analysis for the special games offers additional insight into the complexity of Zielonka's algorithm and its optimisation and may inspire future optimisations of the algorithm. In a similar vein, the same type of analysis can be performed on other parity game solving algorithms from the literature, *e.g.* strategy improvement algorithms.

References

[1] A. Arnold, A. Vincent & I. Walukiewicz (2003): *Games for synthesis of controllers with partial observation*. *TCS* 303(1), pp. 7–34, doi:10.1016/S0304-3975(02)00442-5.

[2] D. Berwanger & E. Grädel (2001): *Games and Model Checking for Guarded Logics*. In: *LPAR*, *LNCS* 2250, Springer, pp. 70–84, doi:10.1007/3-540-45653-8_5.

[3] D. Berwanger & E. Grädel (2004): *Fixed-Point Logics and Solitaire Games*. *Theory Comput. Syst.* 37(6), pp. 675–694. Available at `http://dx.doi.org/10.1007/s00224-004-1147-5`.

[4] T. Chen, B. Ploeger, J. van de Pol & T.A.C. Willemse (2007): *Equivalence Checking for Infinite Systems Using Parameterized Boolean Equation Systems*. In: *CONCUR'07*, pp. 120–135, doi:10.1007/978-3-540-74407-8_9.

[5] E.A. Emerson & C.S. Jutla (1991): *Tree automata, Mu-Calculus and determinacy*. In: *FOCS'91*, IEEE Computer Society, Washington, DC, USA, pp. 368–377, doi:10.1109/SFCS.1991.185392.

[6] E.A. Emerson, C.S. Jutla & A.P. Sistla (2001): *On model checking for the μ-calculus and its fragments*. *Theor. Comput. Sci.* 258(1-2), pp. 491–522, doi:10.1016/S0304-3975(00)00034-7.

[7] O. Friedmann (2011): *Recursive algorithm for parity games requires exponential time*. *RAIRO – Theor. Inf. and Applic.* 45(4), pp. 449–457, doi:10.1051/ita/2011124.

[8] O. Friedmann & M. Lange (2009): *Solving Parity Games in Practice*. In: *ATVA'09*, *LNCS* 5799, Springer, pp. 182–196, doi:10.1007/978-3-642-04761-9_15.

[9] J.F. Groote & M. Keinänen (2005): *A Sub-quadratic Algorithm for Conjunctive and Disjunctive Boolean Equation Systems*. In: *ICTAC*, *LNCS* 3722, Springer, pp. 532–545, doi:10.1007/11560647_35.

[10] M. Jurdziński (1998): *Deciding the Winner in Parity Games is in UP $\cap$ co-UP*. *IPL* 68(3), pp. 119–124, doi:10.1016/S0020-0190(98)00150-1.

[11] M. Jurdziński (2000): *Small Progress Measures for Solving Parity Games*. In: *STACS'00*, *LNCS* 1770, Springer, pp. 290–301, doi:10.1007/3-540-46541-3_24.

[12] M. Jurdziński, M. Paterson & U. Zwick (2006): *A Deterministic Subexponential Algorithm for Solving Parity Games*. In: *SODA'06*, ACM/SIAM, pp. 117–123, doi:10.1145/1109557.1109571.

[13] J.J.A. Keiren (2009): *An experimental study of algorithms and optimisations for parity games, with an application to Boolean Equation Systems*. Master's thesis, Eindhoven University of Technology.

[14] V. King, O. Kupferman & M.Y. Vardi (2001): *On the Complexity of Parity Word Automata*. In: *FoSSaCS*, *LNCS* 2030, Springer, pp. 276–286, doi:10.1007/3-540-45315-6_18.

[15] R. McNaughton (1993): *Infinite games played on finite graphs*. *APAL* 65(2), pp. 149–184, doi:10.1016/0168-0072(93)90036-D.

[16] S. Schewe (2007): *Solving Parity Games in Big Steps*. In: *FSTTCS'07*, *LNCS* 4855, Springer, pp. 449–460, doi:10.1007/978-3-540-77050-3.

[17] R.E. Tarjan (1972): *Depth-First Search and Linear Graph Algorithms*. *SIAM J. Comput.* 1(2), pp. 146–160, doi:10.1137/0201010.

[18] W. Zielonka (1998): *Infinite games on finitely coloured graphs with applications to automata on infinite trees*. *TCS* 200(1-2), pp. 135 – 183, doi:10.1016/S0304-3975(98)00009-7.