

Improving the complexity of Parys' recursive algorithm^{*}

K. Lehtinen, S. Schewe, and D. Wojtczak

University of Liverpool, Liverpool, UK
[k.lehtinen,sven.schewe,d.wojtczak]@liverpool.ac.uk

Abstract. Parys has recently proposed a quasi-polynomial version of Zielonka's recursive algorithm for solving parity games. In this brief note we suggest a variation of his algorithm that improves the complexity to meet the state-of-the-art complexity of broadly $2^{O((\log n)(\log c))}$, while providing polynomial bounds when the number of colours is logarithmic.

Keywords: Parity games, Zielonka's algorithm, quasi-polynomial complexity

1 Introduction

In 2017 Calude et al. published the first quasi-polynomial algorithm for solving parity games [CJK⁺17]. Since then, several alternative algorithms have appeared [Leh18,JL17], the most recent of which is Parys's quasi-polynomial version of the Zielonka's recursive algorithm [Par19].

Parys's algorithm, although enjoying much of the conceptual simplicity of Zielonka's algorithm [Zie98], has a complexity that is a quasi-polynomial factor larger than [CJK⁺17], [JL17], and [FJS⁺17]. More precisely, their complexity is, modulo a small polynomial factor, $\binom{c'+l}{l}$, with c' being c or $c/2$ and $l \in O(\log n)$, for games with n positions and c colours. This also provides fixed-parameter tractability and a polynomial bound for the common case where the number of colours is logarithmic in the number of states. We propose a simplification that brings the complexity of Parys's algorithm down to match this. Note, however, that in a fine grained comparison the recursive algorithm still operates symmetrically, going through every colour, rather than just half of them, and $O(\log n)$ hides a factor of 2. Thus, a very careful analysis still reveals a small gap.

We also briefly comment on the relationship between this recursive algorithm and universal trees.

Notation A parity game $G = (V, V_E, E, \Omega : V \rightarrow [0..c])$ is a two-player game between players Even and Odd, on a finite graph (V, E) , of which positions are partitioned between those belonging to Even, V_E and those belonging of Odd $V_O = V \setminus V_E$, and labelled by π with integer colour from a finite co-domain

^{*} Supported by EPSRC project Solving Parity Games in Theory and Practice.

$[0..c]$ by π . We assume that every position has a successor and that there are no self-loops.

A play π is an infinite path through the game graph. It is winning for Even if the highest colour occurring infinitely often on it is even; else it is winning for Odd. We write $\pi[i]$ for the i^{th} position in π and $\pi[0, j]$ for its prefix of length $j + 1$.

A strategy for a player maps every prefix of a play ending in a position that belongs to this player to one of its successors. A play π agrees with a strategy σ for Even (Odd) if whenever $\pi[i] \in V_E$ (V_O), then $\sigma(\pi[0, i]) = \pi[i + 1]$. A strategy for a player is winning from a position v if all plays beginning at v that it agrees with are winning for that player. Parity games are determined: from every position, one of the two players has a winning strategy [Mar75].

Even's (Odd's) winning region in a parity game is the set of nodes from which Even (Odd) has a winning strategy. We are interested in the problem of computing, given a parity game G , the winning regions of each player.

Given a set $S \subseteq V$, the E-attractor of S in G , written $\text{ATTR}_E(S, G)$, is the set of nodes from which Even has a strategy which only agrees with plays that reach S . O-attractors, written $\text{ATTR}_O(S, G)$ are defined similarly for Odd.

An even dominion is a set of nodes $P \subseteq V$ such that nodes in $P \cap V_E$ have at least one successor in P and nodes in $P \cap V_O$ have all of their successors in P , and Even has a winning strategy within the game induced by P . An odd dominion is defined similarly.

2 The Algorithm

We first recall Parys' quasi-polynomial version of Zielonka's algorithm in Algorithm 1. In brief, the difference between this algorithm and Zielonka's is that this procedure takes a parameter that bounds the size of the dominions the procedure looks for; it first removes one player's dominions (and their attractors) of size up to half the parameter until this does not yield anything anymore, then searches for a single dominion of the size up to the input parameter, then again carries on with searching for small dominions. In each of the recursive calls, the algorithm solves a parity game with one colour less, and either half the input parameter (most of the time) or the full input parameter (once). The correctness hinges on the observation that only one dominion can be larger than half the size of the game, so the costliest call with the full size of the game as parameter needs to be called just once.

Our simplification, in Algorithm 2, replaces each of the two while-loops with a single recursive call that also halves the precision parameter, but, unlike Parys's algorithm, operates on the whole input game arena at once, rather than on a series of subgames of lower maximal colour.

For both algorithms, the dual, SOLVE_O is defined by replacing E with O and vice-versa.

Algorithm 1 $\text{SOLVE}_E(G, h, p_E, p_O)$

```

1: if  $G = \emptyset \vee p_E \leq 1$  then
2:   return  $\emptyset$ ;
3: end if
4: while  $W_O \neq 0$  do
5:    $N_h := \{v \in G \mid \pi(v) = h\}$ ;
6:    $H := G \setminus \text{ATTR}_E(N_h, G)$ 
7:    $W_O := \text{SOLVE}_O(H, h - 1, \lfloor p_O/2 \rfloor, p_E)$ ;
8:    $G := G \setminus \text{ATTR}_O(W_O, G)$ ;
9: end while
10:  $N_h := \{v \in G \mid \pi(v) = h\}$ ;
11:  $H := G \setminus \text{ATTR}_E(N_h, G)$ 
12:  $W_O := \text{SOLVE}_O(H, h - 1, p_O, p_E)$ ;
13:  $G := G \setminus \text{ATTR}_O(W_O, G)$ ;
14: while  $W_O \neq 0$  do
15:    $N_h := \{v \in G \mid \pi(v) = h\}$ ;
16:    $H := G \setminus \text{ATTR}_E(N_h, G)$ 
17:    $W_O := \text{SOLVE}_O(H, h - 1, \lfloor p_O/2 \rfloor, p_E)$ ;
18:    $G := G \setminus \text{ATTR}_O(W_O, G)$ ;
19: end while
20: return  $G$ 

```

Algorithm 2 $\text{SOLVE}_E(G, h, p_E, p_O)$

```

1: if  $G = \emptyset \vee p_E \leq 1$  then
2:   return  $\emptyset$ ;
3: end if
4: if  $p_O \leq 1$  then
5:   return  $G$ 
6: end if
7:  $W'_O := G \setminus \text{SOLVE}_E(G, h, p_E, \lfloor p_O/2 \rfloor)$ ;
8:  $W_O := \text{ATTR}_O(W'_O, G)$ ;
9:  $G' := G \setminus W_O$ ;
10:  $N_h := \{v \in G' \mid \pi(v) = h\}$ ;
11:  $G'' := G' \setminus \text{ATTR}_E(N_h, G')$ 
12:  $W'_O := \text{SOLVE}_O(G'', h - 1, p_O, p_E)$ ;
13: if  $W'_O \neq \emptyset$  then
14:    $W''_O := \text{ATTR}_O(W'_O, G')$ 
15:    $W_O := W_O + W''_O$ 
16:    $G' := G' \setminus W''_O$ ;
17:    $W'_O := G' \setminus \text{SOLVE}_E(G', h, p_E, \lfloor p_O/2 \rfloor)$ ;
18:    $W_O := W_O + \text{ATTR}_O(W'_O, G')$ ;
19:    $G := G \setminus W_O$ ;
20: end if
21: return  $G$ 

```

3 Correctness

Lemma 1. $\text{SOLVE}_E(G, h, p_E, p_O)$, where h is even and no smaller than the maximal priority in G , returns a set that:

- i) contains all even dominions up to size p_E , and
- ii) does not intersect with an odd dominion with size up to p_O .

Similarly, $\text{SOLVE}_O(G, h, p_O, p_E)$, where h is odd and no smaller than the maximal priority in G , returns a set that:

- i) contains all odd dominions up to size p_O , and
- ii) does not intersect with an even dominion with size up to p_E .

Proof. We show this by induction over the sum $h + p_E + p_O$.

Base case $h + p_E + p_O = 1$ Since we assume that there are no self-loops, non-empty dominions have size at least two; hence any set will do.

Induction step We consider the case of SOLVE_E ; the case of SOLVE_O is similar.

We first show i) that $\text{SOLVE}_E(G, h, p_E, p_O)$ returns all even dominions up to size p_E . Let D be such a dominion. According to the IH, D does not intersect with W'_O in line 7 and therefore it does not intersect with W_O on line 8 either. It is therefore contained in G' on line 9. The intersection D' of D and G'' is an even dominion in G'' , and therefore, from the IH, does not intersect with W'_O on line 12 nor its attractor in G' , W''_O . D' therefore does not intersect with W_O on line 15 and is contained in G' on line 16. D' , by IH does not intersect with W'_O on line 17, nor its attractor in G' , and therefore neither does D . D is therefore contained in the returned G .

We proceed with showing ii) that $\text{SOLVE}_E(G, h, p_E, p_O)$ returns a set that does not intersect with odd dominions of size up to p_O . Let D be such a dominion, let S be the union of odd dominions up to size $\lfloor p_O/2 \rfloor$ contained in D and let A be its O-attractor in G .

S is contained in W'_O on line 7 by IH, and therefore A is contained in W_O on line 8 and does not intersect with G' on line 9. If $A = D$ then D is contained in W_O on line 8 and we are done.

We consider the case of $A \neq D$. Since W_O is an odd attractor, the intersection D' of D and G' is also an odd dominion in G' .

If D' is empty, then $D \subseteq W_O$ and therefore, as D does not intersect with the returned G , we are done. If D' is non-empty, it contains an odd dominion C that does not have a position of colour h . $C \subseteq G''$ on line 11 and by IH is contained in W'_O on line 12. C is therefore contained in W_O on line 15 and so is the dominion $A \cup C$, which is larger than $\lfloor p_O/2 \rfloor$ since $A \neq D$. Then, $D \setminus W_O$ is a dominion of G' on line 16 that is smaller than $\lfloor p_O/2 \rfloor$ and therefore contained in W'_O on line 17. Hence D is included in W_O on line 18 and does not intersect with the returned G .

4 Analysis

Let $f(h, l)$ be the number of calls to SOLVE_E and SOLVE_O of $\text{SOLVE}_E(G, h, p_E, p_O)$ (or of $\text{SOLVE}_O(G, h, p_E, p_O)$, if it is greater) where $l = \lfloor \log p_E \rfloor + \lfloor \log p_O \rfloor$.

An induction on l shows that $f(h, l) \leq 2^l \binom{h+l}{l}$. If $l = 0$ then $p_E \leq 1$ and $p_O \leq 1$ so $\text{SOLVE}_E(G, h, p_E, p_O)$ and $\text{SOLVE}_O(G, h, p_E, p_O)$ return immediately. For $l \geq 1$, we have:

$$\begin{aligned} f(h, l) &\leq 2f(h, l-1) + f(h-1, l) \\ &\leq 2^{l-1} \binom{h+l-1}{l-1} + 2^l \binom{h+l-1}{l} \\ &\leq 2^l \binom{h+l}{l} \end{aligned} \tag{1}$$

Then, as $l = 2\lceil \log n \rceil$, this brings the complexity of the simplified algorithm down by a quasi-polynomial factor from Parys' version.

Remark 1. A (n, d) -universal tree is a tree into which all trees of height d with n leaves can be embedded while preserving the ordering of children. These combinatorial objects have emerged as a unifying thread among quasi-polynomial solutions to parity games and have therefore been the object of a recent spree of attention [CDF⁺19, FGO18, CF]. In particular, the size of a universal tree is at least quasi-polynomial, making this a potentially promising direction for lower bounds. We observe that the call tree where the node $\text{SOLVE}_E(G, h, p_E, p_O)$ has for children its calls to SOLVE_E and SOLVE_O with parameter $h-1$ takes the shape of a universal (n, d) -tree where n is the size of the parity game and d its maximal colour. The recursive approach therefore does not seem to be free from universal trees either.

5 Conclusion

This improvement brings the complexity of solving parity games recursively down to almost match the complexity to the algorithms based on Calude et al.'s method [CJK⁺17, FJS⁺17] and Jurdziński and Lazić's algorithm [JL17]. In particular it is fixed-parameter tractable, and polynomial when the number of colours is logarithmic. However, since the recursion solves the game symmetrically—that is, it goes through every colour, rather than just every other colour—and since the size of only the guarantees for the even or odd dominions are halved, in the $\binom{a}{b}$ notation both a (c vs. $c/2$) and b ($2\log n$ vs. $\log n$) double compared to Jurdziński and Lazić's algorithm [JL17].

Whether this simplification to the recursion scheme makes this algorithm usable in practice remains to be seen.

Acknowledgements

We thank Paweł Parys for his comments.

References

- CDF⁺19. Wojciech Czerwiński, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdziński, Ranko Lazić, and Paweł Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2333–2349. SIAM, 2019.
- CF. Thomas Colcombet and Nathanaël Fijalkow. Universal graphs and good for small games automata: New tools for infinite duration games. to appear in proceedings of FOSSACS 2019.
- CJK⁺17. Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 252–263. ACM, 2017.
- FGO18. Nathanaël Fijalkow, Paweł Gawrychowski, and Pierre Ohlmann. The complexity of mean payoff games using universal graphs. *arXiv preprint arXiv:1812.07072*, 2018.
- FJS⁺17. John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. In *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, pages 112–121. ACM, 2017.
- JL17. Marcin Jurdziński and Ranko Lazic. Succinct progress measures for solving parity games. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, volume 00, pages 1–9, June 2017.
- Leh18. Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 639–648. ACM, 2018.
- Mar75. Donald A Martin. Borel determinacy. *Annals of Mathematics*, pages 363–371, 1975.
- Par19. Paweł Parys. Parity games: Zielonka’s algorithm in quasi-polynomial time. Available online: <https://arxiv.org/abs/1904.12446>, 2019.
- Zie98. Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1):135 – 183, 1998.