Winning Cores in Parity Games

Steen Vester

Technical University of Denmark, Kgs. Lyngby, Denmark
stve@dtu.dk

Abstract

Whether parity games can be solved by a polynomial-time algorithm is a well-studied problem which has not yet been resolved. In this talk we propose a new direction for approaching this problem based on the novel notion of a winning core.

We give two different, but equivalent, definitions of a winning core and show a number of interesting properties about them. This includes showing that winning cores can be computed in polynomial time if and only if parity games can be solved in polynomial time and that computation of winning cores is in the intersection of NP and co-NP.

We also present a deterministic polynomial-time approximation algorithm for solving parity games based on computing winning cores. It runs in time $O(d \cdot n^2 \cdot m)$ where d is the number of colors, n is the number of states and m is the number of transitions. The algorithm returns under-approximations of the winning regions in parity games. It works remarkably well in practice as it solves all benchmark games from the PGSolver framework in our experiments completely and outperforms existing algorithms in most cases. Correctness of the output of the algorithm can be checked efficiently.

1 Introduction

Solving parity games [1] is an important problem of both theoretical and practical interest. This is known to be in NP $\cap$ co-NP [2] and UP $\cap$ co-UP [7] but in spite of the development of many different algorithms (see e.g. [13, 18, 8, 17, 9, 15]), frameworks for benchmarking such algorithms [6, 10] and families of parity games designed to expose the worst-case behaviour of existing algorithms [8, 4, 5] it has remained an open problem whether a polynomial-time algorithm exists.

Various problems for which polynomial-time algorithms are not known can been reduced in polynomial time to the problem of solving parity games. Among these are model-checking of the propositional μ-calculus [11, 3, 16], the emptiness problem for parity automata on infinite binary trees [14, 2] and solving boolean equation systems [12].

Some of the most notable algorithms from the litterature of solving parity games include the recursive algorithms from [13, 18] using $O(n^d)$ time, the small progress measures algorithm [8] using $O(d \cdot m \cdot (n/d)^{d/2})$ time, the strategy improvement algorithm [17] using $O(n \cdot m \cdot 2^m)$ time, the big step algorithm [15] using $O(m \cdot n^{d/3})$ time and the dominion decomposition algorithm [9] using $O(n^{\sqrt{n}})$ time. Here, n is the number of states in the game, m is the number of transitions and d is the maximal color occuring in the game.

2 Contributions

First, we introduce some notation. In the following we fix a finite parity game $\mathcal{G}$ (for a definition, see e.g. [18]) with colors in $\{1, ..., d\}$. The set of winning states for player j in $\mathcal{G}$ is denoted $W_j(\mathcal{G})$. We say that a (finite or infinite) sequence $\rho = s_0 s_1 ...$ of states with at least one transition is *0-dominating* if the greatest color $e = \max\{c(s_i) \mid i > 0\}$ occuring in a non-initial state of ρ is even and *1-dominating* if it is odd. Examples are shown in Figure 1.

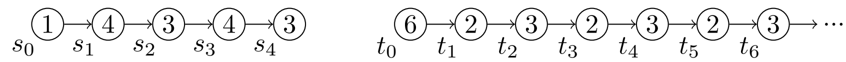


Figure 1: The sequence to the left is 0-dominating and the sequence to the right is 1-dominating.

We say that a play ρ begins with k consecutive j-dominating sequences if there exists indices $i_0 < i_1 < ... < i_k$ with $i_0 = 0$ such that $\rho_{i_\ell}\rho_{i_\ell+1}...\rho_{i_{\ell+1}}$ is j-dominating for all $0 \leq \ell < k$. This definition is straightforwardly extended to an infinite number of consecutive j-dominating sequences. As examples, the sequence on the left in Figure 1 begins with two consecutive 0-dominating sequences $s_0 s_1$ and $s_1 s_2 s_3$ whereas the sequence to the right begins with only one 0-dominating sequence $t_0 t_1$, but not two consecutive 0-dominating sequences.

For a player j and a parity game $\mathcal{G}$ the *winning core* $A_j(\mathcal{G})$ is defined as the set of states in $\mathcal{G}$ from which player j can force that the play begins with an infinite number of consecutive j-dominating sequences. Our main results on winning cores are the following.

Proposition 1. *Let ρ be a play. Then ρ begins with an infinite number of consecutive j-dominating sequences if and only if ρ is j-dominating and winning for player j.*

Theorem 1. *Let $\mathcal{G}$ be a parity game and j be a player. Then*

1. $A_j(\mathcal{G}) \subseteq W_j(\mathcal{G})$
2. $A_j(\mathcal{G}) = \emptyset$ *if and only if* $W_j(\mathcal{G}) = \emptyset$

Proposition 2. *There exists a parity game $\mathcal{G}$ where $A_j(\mathcal{G})$ is not a j-dominion [9].*

Theorem 2. *Computing winning cores is in* $\mathrm{NP} \cap \textsc{co}\text{-}\mathrm{NP}$.

Theorem 3. *Computing winning cores can be done in polynomial time if and only if parity games can be solved in polynomial time.*

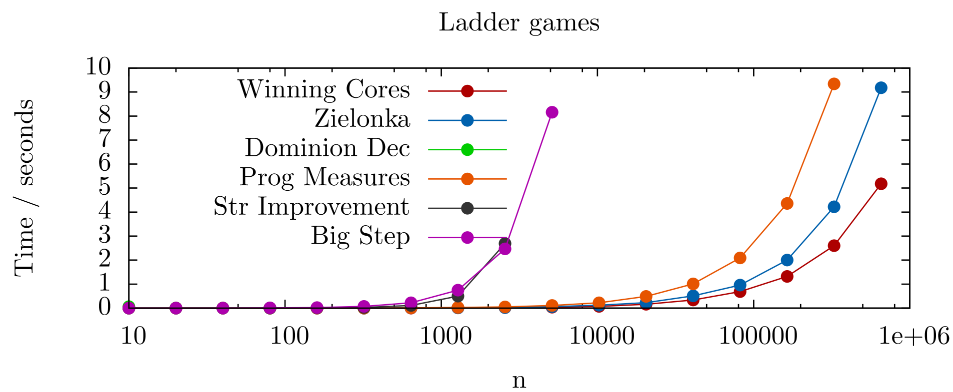
Proposition 1 gives us two equivalent definitions of the same concept which is not immediately obvious. Theorem 1 provides us with valuable information about the winning cores and, further, it is used to design an algorithm for solving parity games based on computing winning cores. Proposition 2 is interesting as many algorithms for solving parity games focus on finding j-dominions whereas the winning core is a subset of winning states that is not necessarily a j-dominion. Finally, Theorem 2 and 3 make the search for a polynomial-time algorithm for computing winning cores a viable direction in the search for a polynomial-time algorithm for solving parity games.

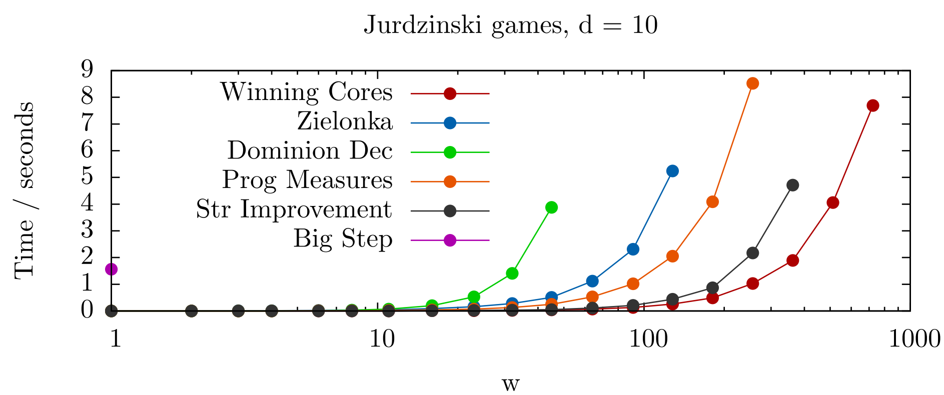
The results are used to develop a new fixpoint algorithm that calculates under-approximations of the winning regions in parity games. This algorithm runs in time $O(d \cdot n^2 \cdot m)$ and is very fast in practice. Further, it can be efficiently checked whether the output of the algorithm is correct. This means that it can be applied with confidence when it outputs the correct result.

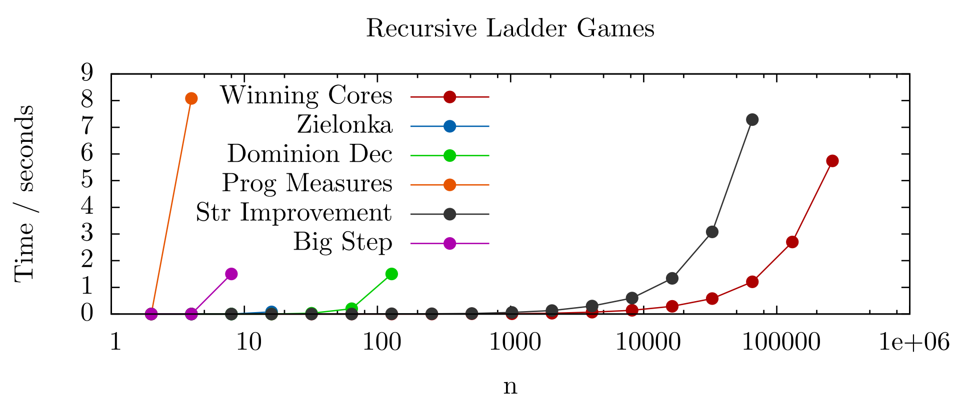
In our experiments the algorithm performs remarkably well returning the complete winning region in most cases. In Figure 2 are experimental results on correctness in randomly generated games. Further, the algorithm returns the correct results on all other benchmark games from the PGSolver framework on which it has been tested. A comparison of running times for some of the benchmarks can be seen below. These benchmarks are games designed to be difficult for some of the existing algorithms to solve. Our algorithm has been implemented in OCaml as a part of the PGSolver framework using the same basic data structures as the other algorithms.

	$d = 4$			$d = \lceil \sqrt{n} \rceil$			$d = n$		
$n \backslash b$	2	5	10	2	5	10	2	5	10
100	0.002%	0.000%	0.000%	0.114%	0.001%	0.000%	0.559%	0.011%	0.000%
1000	0.000%	0.000%	0.000%	0.571%	0.000%	0.000%	2.113%	0.000%	0.000%

Figure 2: Ratio of games where the algorithm did not return the entire winning region. n is the number of states, d is the number of colors and b is the out-degree. For every fixed n, d and b the experiments were done on 100.000 games generated randomly by PGSolver [6].







References

[1] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377, 1991.

[2] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theor. Comput. Sci.*, 258(1-2):491–522, 2001.

[3] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In Albert Meyer, editor, *Proceedings of the First Annual IEEE Symp. on Logic in Computer Science, LICS 1986*, pages 267–278. IEEE Computer Society Press, June 1986.

[4] Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 145–156, 2009.

[5] Oliver Friedmann. Recursive algorithm for parity games requires exponential time. *RAIRO - Theor. Inf. and Applic.*, 45(4):449–457, 2011.

[6] Oliver Friedmann and Martin Lange. Solving parity games in practice. In *Automated Technology for Verification and Analysis, 7th International Symposium, ATVA 2009, Macao, China, October 14-16, 2009. Proceedings*, pages 182–196, 2009.

[7] Marcin Jurdzinski. Deciding the winner in parity games is in UP \cap co-up. *Inf. Process. Lett.*, 68(3):119–124, 1998.

[8] Marcin Jurdzinski. Small progress measures for solving parity games. In *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings*, pages 290–301, 2000.

[9] Marcin Jurdzinski, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 117–123, 2006.

[10] Jeroen J. A. Keiren. Benchmarks for parity games. *CoRR*, abs/1407.3121, 2014.

[11] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[12] Angelika Mader. *Verification of Modal Properties Using Boolean Equation Systems*. Edition versal 8. Bertz Verlag, Berlin, Germany, 1997. Imported from DIES.

[13] Robert McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993.

[14] Andrzej Wlodzimierz Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory - Fifth Symposium, Zaborów, Poland, December 3-8, 1984, Proceedings*, pages 157–168, 1984.

[15] Sven Schewe. Solving parity games in big steps. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, New Delhi, India, December 12-14, 2007, Proceedings*, pages 449–460, 2007.

[16] Colin Stirling. Local model checking games. In *CONCUR '95: Concurrency Theory, 6th International Conference, Philadelphia, PA, USA, August 21-24, 1995, Proceedings*, pages 1–11, 1995.

[17] Jens Vöge and Marcin Jurdzinski. A discrete strategy improvement algorithm for solving parity games. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, pages 202–215, 2000.

[18] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.