Solving Parity Games with Tangles

Tom van Dijk (JKU Linz)

Highlights of Logic, Games and Automata, 19 Sept 2018

Three contributions

My work on parity games

- A new solver Oink (fast), TACAS 2018
  - C++, way faster than state-of-the-art
  - Implements many prominent algorithms
- A new algorithm tangle learning, CAV 2018
  - More powerful than recursive/Zielonka, priority promotion
  - Based on a critical insight of tangle attractors
- A powerful exponential parameterized parity game
  - Defeats several attractor-based algorithms, $\mathcal{O}(2^{\sqrt{n}})$
  - Can be defeated by a variation of tangle learning

Parity Games

Attractor computation

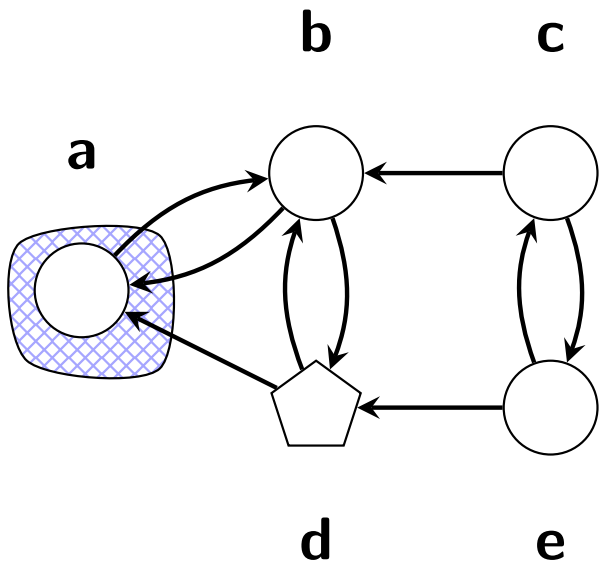
Compute all vertices from which player $\alpha \in \{\bigcirc, \pentagon\}$ can ensure arrival in a given target set

Start with the target set A, then iteratively add vertices to A:

- All vertices of α with an edge to A
- All vertices of $\overline{\alpha}$ with only edges to A

Example of attractor computation

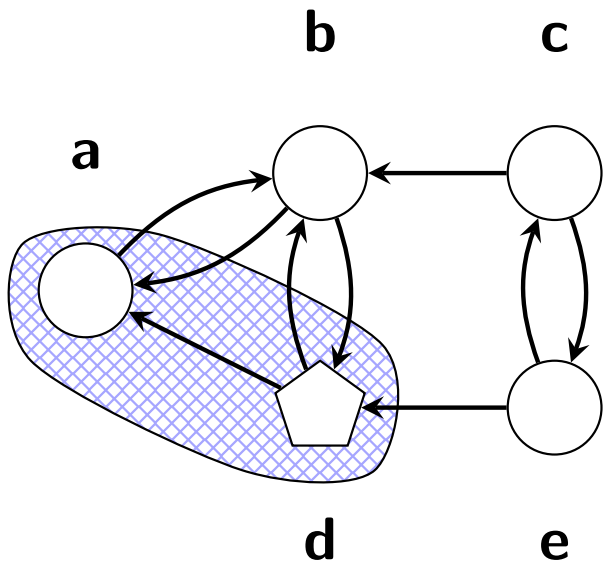
Computing the ⬠-attractor to **a**



Initial set: $\{\mathbf{a}\}$
Can attract: **d** but not **b**

Example of attractor computation

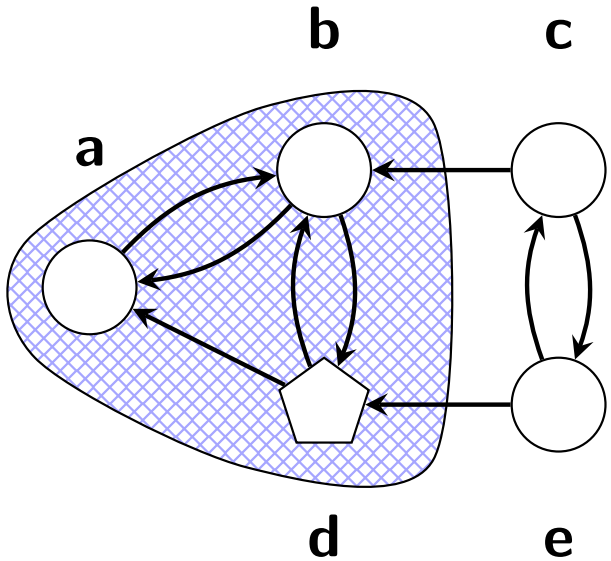
Computing the ⬠-attractor to **a**



Current set: $\{\mathbf{a}, \mathbf{d}\}$
Can attract: **b** but not **e**

Example of attractor computation

Computing the ⬠-attractor to **a**

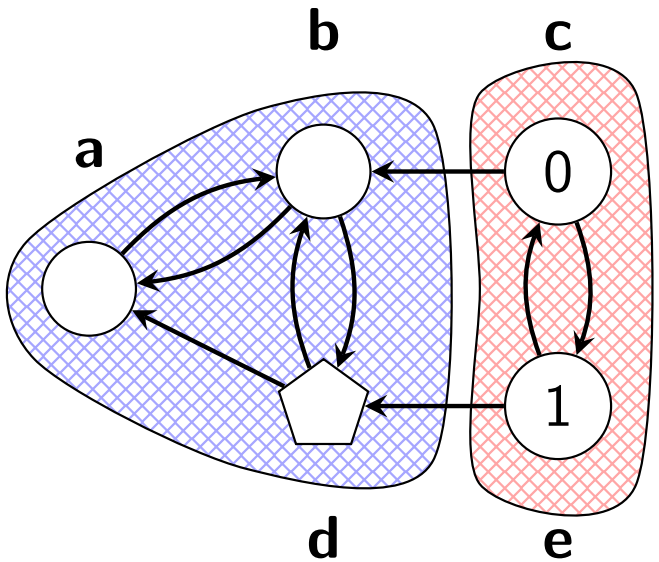


Current set: $\{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$
Can attract: neither **c** nor **e**

Highlights

The notion of a tangle

- A tangle is a strongly connected subgame, such that one player can win all plays that stay in the tangle
- One player has a strategy to win all cycles in the subgame
- Therefore the other player must leave the subgame



Highlights

The notion of a tangle

- A tangle is a strongly connected subgame, such that one player can win all plays that stay in the tangle
- One player has a strategy to win all cycles in the subgame
- Therefore the other player must leave the subgame

The role of tangles in parity game solving algorithms

- Many algorithms implicitly explore tangles
- They often explore the same tangles over and over again
- This leads to an exponential number of steps

Highlights

Main insight

Tangles can be used with attractor computation!
The losing player **must leave** the tangle.
Thus we can attract the entire tangle.

$\Rightarrow$ Tangle attractor!

Highlights

Tangle learning

- Decompose game using the tangle attractor
  - Compute attractor set to highest priority
  - Remove this attractor set from the game
- Compute new tangles from decomposition
  - Each "closed" attractor set is a snare
  - The SCCs of each snare are new tangles
- Repeat until the game is solved

Highlights

Two Counters game

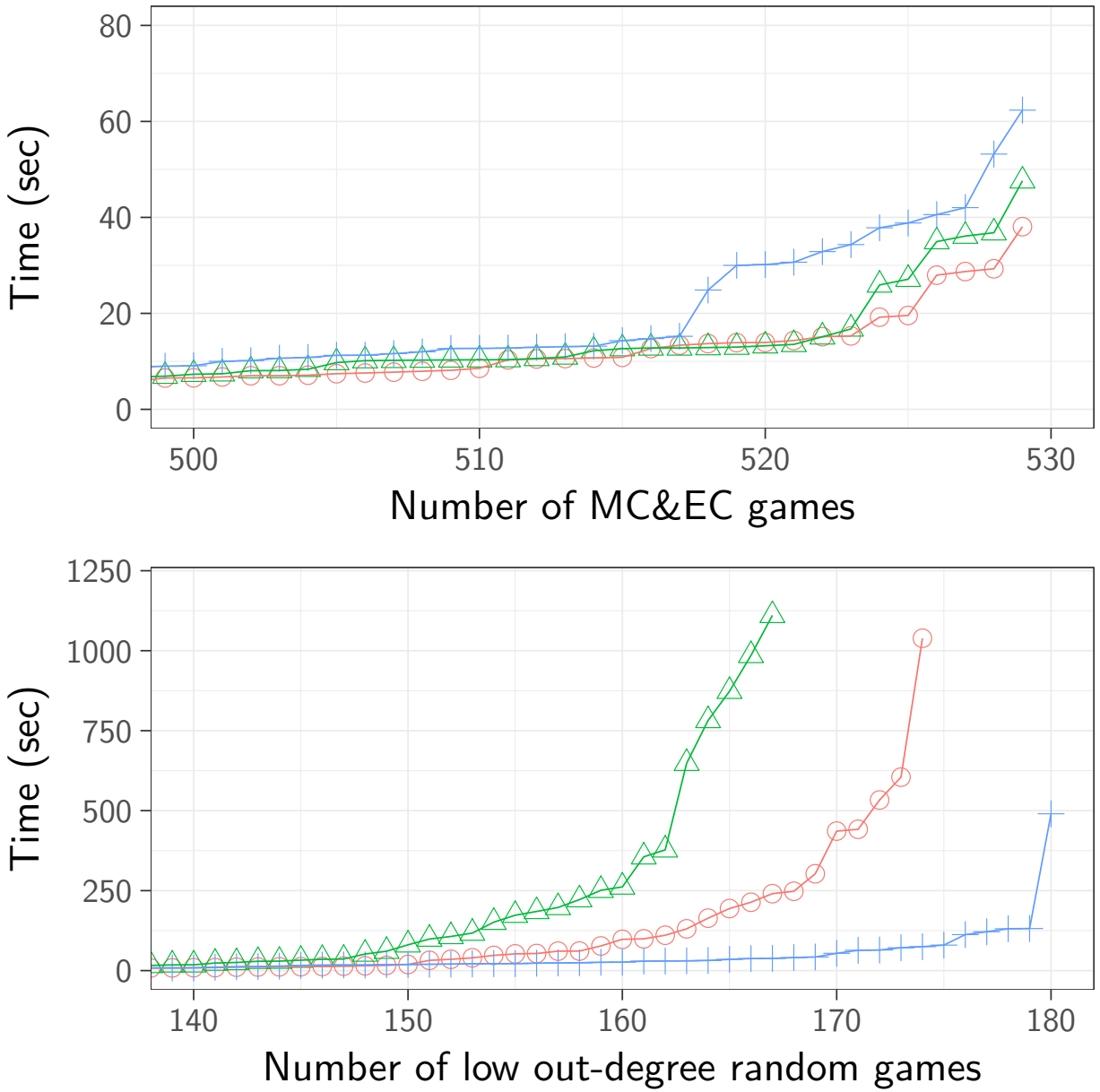
- Game based on N-bit binary counters for both players
- Exponential lower bound for:
  - Recursive algorithm (McNaughton/Zielonka)
  - Priority promotion & variations PP+, DP, RR
  - The `APT` algorithm
  - Tangle learning
- (to be submitted; currently on arXiv)

Empirical evaluation

- Evaluated using Oink (TACAS 2018)
- Benchmarks
  - Model checking and equivalence checking games [Keiren 2015]
  - Random games
  - Random games with max out-degree 2
- Runtimes in seconds, timeout 20 minutes

Solver	**MC&EC**	**Random**	**Random (low degree)**	
	time	time	time	timeouts
priority promotion	503	21	12770	6
recursive	576	21	23119	13
tangle learning	808	21	2281	0

Empirical evaluation



Conclusions

- Fast moving field with great progress in the last few years
- Existing algorithms implicitly explore tangles repeatedly
- Tangles can be used with attractor computation
- Tangle learning explicitly remembers tangles
- See for yourself: `https://www.github.com/trolando/oink`

Solver	**MC&EC**	**Random**	**Random (low degree)**	
	time	time	time	timeouts
priority promotion	503	21	12770	6
recursive	576	21	23119	13
tangle learning	808	21	2281	0