# Solving Random Regular Parity Games in Polynomial Time

M. Touati[1] joint work with R. Combes[2]

[1]Orange [2]CentraleSupélec

13$^{\text{ème}}$ Atelier en Evaluation des Performances
2-4 December 2024, Toulouse

# Table of Contents

# Parity game

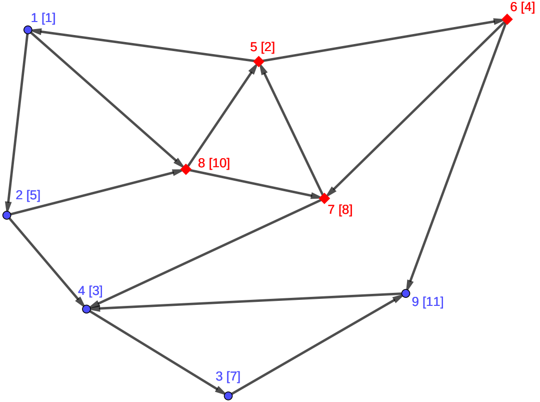


Figure: A parity game. Blue circles belong to player *Odd* and red diamonds to *Even*. Each node is labeled by a pair $x[y]$ where $x$ is the node and $y$ is its priority.

# Rules and play

- Even and Odd play on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by moving the token, starting at a node $v_0$, for an infinite sequence of moves.

# Rules and play

- Even and Odd play on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by moving the token, starting at a node $v_0$, for an infinite sequence of moves.
- If the token is on
  - $v \in \mathcal{V}_E$, then *Even* chooses a successor $v'$ of $v$ and the token is moved to $v'$.
  - $v \in \mathcal{V}_O$, then *Odd* chooses a successor $v'$ of $v$ and the token is moved to $v'$.

# Rules and play

- Even and Odd play on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by moving the token, starting at a node $v_0$, for an infinite sequence of moves.
- If the token is on
  - $v \in \mathcal{V}_E$, then *Even* chooses a successor $v'$ of $v$ and the token is moved to $v'$.
  - $v \in \mathcal{V}_O$, then *Odd* chooses a successor $v'$ of $v$ and the token is moved to $v'$.
- An infinite walk in $\mathcal{G}$ is called a *play* and denoted $p = v_0 v_1 v_2 \ldots$ where $v_0$ is the initial node and $v_{t+1}$ is the successor of $v_t$.

# Rules and play

- Even and Odd play on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by moving the token, starting at a node $v_0$, for an infinite sequence of moves.
- If the token is on
  - $v \in \mathcal{V}_E$, then *Even* chooses a successor $v'$ of $v$ and the token is moved to $v'$.
  - $v \in \mathcal{V}_O$, then *Odd* chooses a successor $v'$ of $v$ and the token is moved to $v'$.
- An infinite walk in $\mathcal{G}$ is called a *play* and denoted $p = v_0 v_1 v_2 \dots$ where $v_0$ is the initial node and $v_{t+1}$ is the successor of $v_t$.
- *Even* wins if the maximal priority occurring infinitely often is even.

# Memoryless strategies and determinacy

**Definition**

A positional (a.k.a. memoryless) strategy for player $i$ is a function $s : \mathcal{V}_i \to \mathcal{V}$ from any node of $i$ to one of its successor.

[1] For each initial position or for the given one.

# Memoryless strategies and determinacy

**Definition**

A positional (a.k.a. memoryless) strategy for player $i$ is a function $s : \mathcal{V}_i \rightarrow \mathcal{V}$ from any node of $i$ to one of its successor.

**Theorem (Positional Determinacy, [Jurdziński et al., 2008])**

*For every parity game, one of the players has a positional winning strategy.*

[1] For each initial position or for the given one.

# Memoryless strategies and determinacy

**Definition**

A positional (a.k.a. memoryless) strategy for player $i$ is a function $s : \mathcal{V}_i \to \mathcal{V}$ from any node of $i$ to one of its successor.

**Theorem (Positional Determinacy, [Jurdziński et al., 2008])**

*For every parity game, one of the players has a positional winning strategy.*

- Parity games can be decided and memoryless strategies are sufficient.
- Solving a parity game = computing[1] which player has a winning strategy.

[1] For each initial position or for the given one.

# Related work

- Connected to logic, model-checking and other games [2] [Grädel et al., 2002].
- Studied in [Emerson et al., 1993] [McNaughton, 1993] and [Zielonka, 1998].
- Computing the value of a parity game is known to be in [Calude et al., 2017] NP ∩ Co-NP, UP ∩ co-UP, QP.
- Many techniques [Jurdzinski and Lazić, 2017]: progress measures, big steps and optimized recursions, strategy improvement and randomization.

[2] Mean payoff, discounted payoff & simple stochastic games

# Related work

- First QP algorithm in [Calude et al., 2017].
- Other works have refined the analysis and proposed new algorithms [Lehtinen, 2018][Parys, 2019].
- [van Dijk, 2018][Benerecetti et al., 2018] compare algorithms, showing that Zielonka's algorithm and priority promotion perform efficiently in practice.
- About phase transitions and the "average" case complexity see [Mezard and Montanari, 2009].

# Objectives & contributions

- No polynomial algorithm.
- We want to study the "typical" complexity [3] and the influence of the parameters.

[3] Drawing instances at random following some distribution.

# Objectives & contributions

- No polynomial algorithm.
- We want to study the "typical" complexity [3] and the influence of the parameters.

## Objectives

- Propose a polynomial-time algorithm, called SWCP (Self-Winning Cycles Propagation), running in $O\Big(|\mathcal{V}|^2 + |\mathcal{V}||\mathcal{E}|\Big)$ time on any parity game.
- Show that, for "fair" random $d$-out regular parity games, if $d$ satisfies a condition then SWCP solves the game w.h.p. as $|\mathcal{V}| \to +\infty$.

[3] Drawing instances at random following some distribution.

# Table of Contents

# Model

- Focus on parity games on $d$-out regular directed graph.

**Definition ($d$-out regular directed graph)**

A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is d-out regular if every node $v \in \mathcal{V}$ has out-degree $d$.

# Model

- Focus on parity games on $d$-out regular directed graph.

**Definition ($d$-out regular directed graph)**

A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is d-out regular if every node $v \in \mathcal{V}$ has out-degree $d$.

- For any $n, d \in \mathbb{N}$, $\mathcal{G}(n, d)$ denotes the set of $d$-out regular directed graphs with nodes $\mathcal{V} = \{1, \ldots, n\}$.

# Model

- Focus on parity games on $d$-out regular directed graph.

**Definition ($d$-out regular directed graph)**

A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is d-out regular if every node $v \in \mathcal{V}$ has out-degree $d$.

- For any $n, d \in \mathbb{N}$, $\mathcal{G}(n, d)$ denotes the set of $d$-out regular directed graphs with nodes $\mathcal{V} = \{1, \ldots, n\}$.
- Let $X : (\Omega, \mathcal{F}, \mu) \to (\mathcal{G}(n, d) \times \{Even, Odd\}^n \times \mathcal{C}^n, \mathcal{B})$ be a random game graph s.t.,

$$\forall \omega \in \Omega,\ X(\omega) = (X_{graph}(\omega), X_{owner}(\omega), X_{priority}(\omega))$$

# Assumptions

1. $X_{graph}$, $X_{owner}$ and $X_{priority}$ are independent.

# Assumptions

1. $X_{graph}$, $X_{owner}$ and $X_{priority}$ are independent.
2. The graph is drawn uniformly in $\mathcal{G}(n, d)$.
   - For any $(\mathcal{V}, \mathcal{E}) \in \mathcal{G}(n, d)$, $\mathbb{P}(X_{graph} = (\mathcal{V}, \mathcal{E})) = \frac{1}{|\mathcal{G}(n,d)|}$

# Assumptions

1. $X_{graph}$, $X_{owner}$ and $X_{priority}$ are independent.
2. The graph is drawn uniformly in $\mathcal{G}(n, d)$.
   - For any $(\mathcal{V}, \mathcal{E}) \in \mathcal{G}(n, d)$, $\mathbb{P}(X_{graph} = (\mathcal{V}, \mathcal{E})) = \frac{1}{|\mathcal{G}(n,d)|}$
3. Owners are assigned independently and "fairly".
   - $(X_{owner}(v))_{v \in \mathcal{V}}$ are iid and $\mathbb{P}(X_{owner}(v) = Even) = \frac{1}{2}$

# Assumptions

1. $X_{graph}$, $X_{owner}$ and $X_{priority}$ are independent.
2. The graph is drawn uniformly in $\mathcal{G}(n,d)$.
   - For any $(\mathcal{V},\mathcal{E}) \in \mathcal{G}(n,d)$, $\mathbb{P}(X_{graph} = (\mathcal{V},\mathcal{E})) = \frac{1}{|\mathcal{G}(n,d)|}$
3. Owners are assigned independently and "fairly".
   - $(X_{owner}(v))_{v\in\mathcal{V}}$ are iid and $\mathbb{P}(X_{owner}(v) = Even) = \frac{1}{2}$
4. Priorities are assigned independently and "fairly".
   - $(X_{priority}(v))_{v\in\mathcal{V}}$ are iid and $\mathbb{P}(X_{priority}(v) \text{ is even }) = \frac{1}{2}$

# Table of Contents

### Definition (Self-winning cycle)

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

### Definition (Self-winning cycle)

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

- For any $v \in \mathcal{V}$, $v$ is a self-winning node if it is in a self-winning cycle.

**Definition (Self-winning cycle)**

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

- For any $v \in \mathcal{V}$, $v$ is a self-winning node if it is in a self-winning cycle.
- `SWCP`

**Definition (Self-winning cycle)**

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

- For any $v \in \mathcal{V}$, $v$ is a self-winning node if it is in a self-winning cycle.
- `SWCP`
  1. Compute the players' self-winning nodes

### Definition (Self-winning cycle)

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

- For any $v \in \mathcal{V}$, $v$ is a self-winning node if it is in a self-winning cycle.
- `SWCP`
  1. Compute the players' self-winning nodes
  2. Evaluate the corresponding nodes as winning for their owner

### Definition (Self-winning cycle)

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

- For any $v \in \mathcal{V}$, $v$ is a self-winning node if it is in a self-winning cycle.
- `SWCP`
  1. Compute the players' self-winning nodes
  2. Evaluate the corresponding nodes as winning for their owner
  3. Propagate iteratively the values by dynamic programming
     - if $\exists v' \in \mathcal{V}$ s.t. $(v, v') \in \mathcal{E}$ and $v'$ is winning for $Owner(v)$, then $v$ is winning for $Owner(v)$

### Definition (Self-winning cycle)

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

- For any $v \in \mathcal{V}$, $v$ is a self-winning node if it is in a self-winning cycle.
- SWCP
  1. Compute the players' self-winning nodes
  2. Evaluate the corresponding nodes as winning for their owner
  3. Propagate iteratively the values by dynamic programming
     - if $\exists v' \in \mathcal{V}$ s.t. $(v, v') \in \mathcal{E}$ and $v'$ is winning for $Owner(v)$, then $v$ is winning for $Owner(v)$
     - else if $\forall v' \in \mathcal{V}$ s.t. $(v, v') \in \mathcal{E}$, $v'$ is loosing for $Owner(v)$, then $v$ is loosing for $Owner(v)$

### Definition (Self-winning cycle)

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

- For any $v \in \mathcal{V}$, $v$ is a self-winning node if it is in a self-winning cycle.
- `SWCP`
  1. Compute the players' self-winning nodes
  2. Evaluate the corresponding nodes as winning for their owner
  3. Propagate iteratively the values by dynamic programming
     - if $\exists v' \in \mathcal{V}$ s.t. $(v, v') \in \mathcal{E}$ and $v'$ is winning for $Owner(v)$, then $v$ is winning for $Owner(v)$
     - else if $\forall v' \in \mathcal{V}$ s.t. $(v, v') \in \mathcal{E}$, $v'$ is loosing for $Owner(v)$, then $v$ is loosing for $Owner(v)$
     - else the value of $v$ cannot be decided.

## Definition (Self-winning cycle)

A self-winning cycle $\mathcal{C}$ for *Even* (respect. *Odd*) is a cycle in $\mathcal{G}_{Even}$ (respect. $\mathcal{G}_{Odd}$) whose maximal prioriy is even (respect. odd).

- For any $v \in \mathcal{V}$, $v$ is a self-winning node if it is in a self-winning cycle.
- `SWCP`
  1. Compute the players' self-winning nodes
  2. Evaluate the corresponding nodes as winning for their owner
  3. Propagate iteratively the values by dynamic programming
     - if $\exists v' \in \mathcal{V}$ s.t. $(v, v') \in \mathcal{E}$ and $v'$ is winning for $Owner(v)$, then $v$ is winning for $Owner(v)$
     - else if $\forall v' \in \mathcal{V}$ s.t. $(v, v') \in \mathcal{E}$, $v'$ is loosing for $Owner(v)$, then $v$ is loosing for $Owner(v)$
     - else the value of $v$ cannot be decided.
  4. Return the evaluations and strategies.

# Complexity

**Proposition**

SWCP *algorithm has time complexity* $O\left(|\mathcal{V}|^2 + |\mathcal{V}||\mathcal{E}|\right)$.

# Table of Contents

# Intuition

- The objective is to show the conditions implying that `SWCP` solves random d-out regular parity games w.h.p.

# Intuition

- The objective is to show the conditions implying that `SWCP` solves random d-out regular parity games w.h.p.
- For any $v \in \mathcal{V}$, if
  - "around $v$" the graph looks like a tree
  - each path to a leaf of this tree contains nodes that can be "easily decided".

then $v$ can be decided by dynamic programming.

# Intuition

- The objective is to show the conditions implying that `SWCP` solves random d-out regular parity games w.h.p.
- For any $v \in \mathcal{V}$, if
  - "around $v$" the graph looks like a tree
  - each path to a leaf of this tree contains nodes that can be "easily decided".

  then $v$ can be decided by dynamic programming.
- These "easily decided nodes" emerge with self-winning cycles in the players subgraphs.

# Intuition

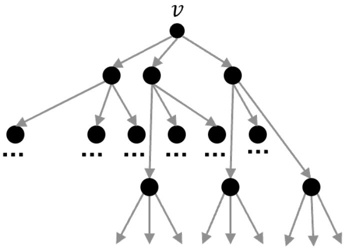


Figure: Rationale of SWCP.

# Intuition

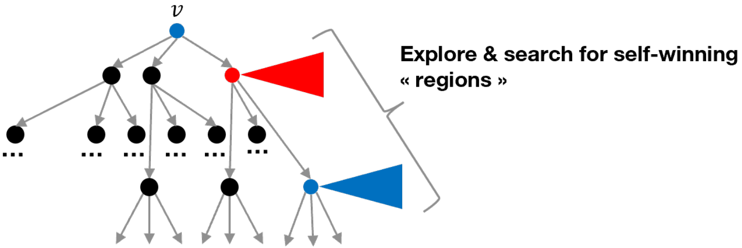


Figure: Rationale of SWCP.

# Intuition

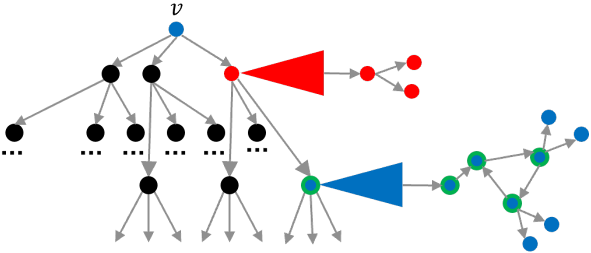


Figure: Rationale of SWCP.

# Sufficient conditions

- For any $v \in \mathcal{V}$, let $\ell : \mathcal{V} \to \{-1, 0, +1\}$ s.t.

$$\ell(v) = \begin{cases} u^4(Owner(v)) \text{ if } \exists \text{ path in } \mathcal{G}_{Owner(v)} \text{ from } v \text{ to a SWN}, \\ 0 \text{ otherwise.} \end{cases}$$

[4] $u(Even) = +1$ and $u(Odd) = -1$

# Sufficient conditions

- For any $v \in \mathcal{V}$, let $\ell : \mathcal{V} \to \{-1, 0, +1\}$ s.t.

$$\ell(v) = \begin{cases} u^4(Owner(v)) \text{ if } \exists \text{ path in } \mathcal{G}_{Owner(v)} \text{ from } v \text{ to a SWN,} \\ 0 \text{ otherwise.} \end{cases}$$

- $A_{h,v} = \{\, T_h(v) \text{ is a tree}\}$
- $B_{h,v} = \{\forall v \to v_1 \to ... \to v_h, \exists j \in \{1, ..., h\} \text{ s.t. } \ell(v_j) \neq 0\}$

[4] $u(Even) = +1$ and $u(Odd) = -1$

# Sufficient conditions

- For any $v \in \mathcal{V}$, let $\ell : \mathcal{V} \to \{-1, 0, +1\}$ s.t.

$$\ell(v) = \begin{cases} u^4(Owner(v)) \text{ if } \exists \text{ path in } \mathcal{G}_{Owner(v)} \text{ from } v \text{ to a SWN,} \\ 0 \text{ otherwise.} \end{cases}$$

- $A_{h,v} = \{\, T_h(v) \text{ is a tree}\}$
- $B_{h,v} = \{\forall v \to v_1 \to \ldots \to v_h, \exists j \in \{1, \ldots, h\} \text{ s.t. } \ell(v_j) \neq 0\}$

**Proposition**

$A_{h,v} \cap B_{h,v} \subseteq \{v \textit{ is decidable at height } h\}$

- $\mathbb{P}(\{v \text{ is not decidable at height } h\}) \leq \mathbb{P}(A^c_{h,v}) + \mathbb{P}(B^c_{h,v})$

[4] $u(Even) = +1$ and $u(Odd) = -1$

# The graph is locally tree like w.h.p.

- For any $v \in \mathcal{V}$ and $h \geq 0$, let $T_h(v)$ be the (random) subgraph with nodes the descendants of $v$ within distance $h$ or less.

# The graph is locally tree like w.h.p.

- For any $v \in \mathcal{V}$ and $h \geq 0$, let $T_h(v)$ be the (random) subgraph with nodes the descendants of $v$ within distance $h$ or less.
- The following proposition shows that w.h.p. $T_{h(|\mathcal{V}|)}(v)$ is a tree if $h(|\mathcal{V}|)$ is sufficiently small with respect to $|\mathcal{V}|$.

## Proposition

*If* $h(|\mathcal{V}|) = o(\ln |\mathcal{V}|)$ *then* $\{T_{h(|\mathcal{V}|)}(v)$ *is a tree*$\}$ *occurs with high probability.*

# Searching for self-winning nodes with exploration

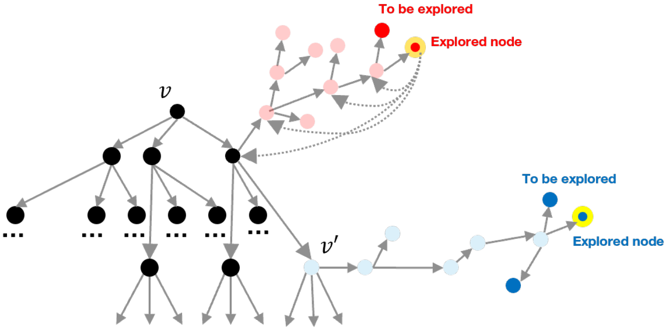


Figure: Representation of explorations.

# Searching for self-winning nodes with exploration

- A graph to explore has $|\mathcal{V}'|$ nodes and $Bin(d, q)$ degree.

# Searching for self-winning nodes with exploration

- A graph to explore has $|\mathcal{V}'|$ nodes and $Bin(d, q)$ degree.
- $N_T$ =number of cycles in the graph among nodes observed by the process after $T$ exploration steps.
- $k =$ number of initial nodes.
- $\eta(d, q(1-\varepsilon))^k =$ extinction probability of a branching process with offspring distrib. $Bin(d, q(1-\epsilon))$, $\epsilon > 0$.

# Searching for self-winning nodes with exploration

- A graph to explore has $|\mathcal{V}'|$ nodes and $Bin(d, q)$ degree.
- $N_T$ =number of cycles in the graph among nodes observed by the process after $T$ exploration steps.
- $k$ = number of initial nodes.
- $\eta(d, q(1-\varepsilon))^k$ = extinction probability of a branching process with offspring distrib. $Bin(d, q(1-\epsilon))$, $\epsilon > 0$.

## Proposition

*If* $\varepsilon > 0$ *and* $T = \frac{\varepsilon}{d}|\mathcal{V}'|$ *then* $\lim_{|\mathcal{V}'|\to\infty} \mathbb{P}(N_T = 0) \leq \eta(d, q(1-\varepsilon))^k$.

# Result

**Main result**

If $d\eta\left(d-1,\frac{1}{4}\right) < 1$ then `SWCP` solves a random $d$-out regular parity game w.h.p.

- Ongoing revision, the paper will be updated soon.

# Table of Contents

# Numerical results

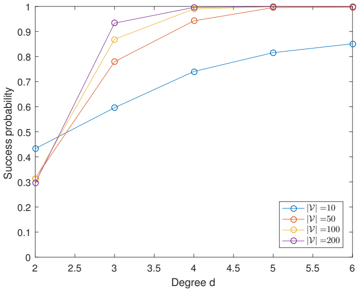


Figure: Probability that SWCP successfully evaluates a node.

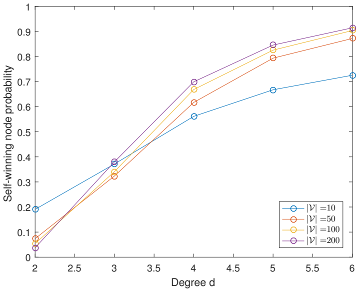


Figure: Proportion of self-winning nodes.

# Table of Contents

# Conclusion

- Study random d-out regular parity games.
- Proposed SWCP running in time $O\left(|\mathcal{V}|^2 + |\mathcal{V}||\mathcal{E}|\right)$ to solve random d-out regular parity games (satisfying a degree condition) w.h.p.
- Next steps
  - Relax the assumptions to solve "more parity games".
  - Generalize the approach to other games.

Thank you !

Benerecetti, M., Dell'Erba, D., and Mogavero, F. (2018).
Solving parity games via priority promotion.
*Formal Methods in System Design*, 52.

Calude, C. S., Jain, S., Khoussainov, B., Li, W., and Stephan, F. (2017).
Deciding parity games in quasipolynomial time.
In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 252–263, New York, NY, USA. Association for Computing Machinery.

Emerson, E. A., Jutla, C. S., and Sistla, A. P. (1993).
On model-checking for fragments of $\mu$-calculus.
In *Computer Aided Verification*, pages 385–396. Springer Berlin Heidelberg.

Grädel, E., Thomas, W., and Wilke, T., editors (2002).
*Automata Logics, and Infinite Games: A Guide to Current Research*.
Springer-Verlag, Berlin, Heidelberg.

Jurdzinski, M. and Lazić, R. (2017).
Succinct progress measures for solving parity games.
In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '17. IEEE Press.

Jurdziński, M., Paterson, M., and Zwick, U. (2008).
A deterministic subexponential algorithm for solving parity games.
*SIAM Journal on Computing*, 38(4):1519–1532.

Lehtinen, K. (2018).
A modal $\mu$ perspective on solving parity games in quasi-polynomial time.
In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, page 639–648.

McNaughton, R. (1993).
Infinite games played on finite graphs.
*Annals of Pure and Applied Logic*, 65(2):149 – 184.

Mezard, M. and Montanari, A. (2009).

*Information, Physics and Computation*.
Oxford University Press.

Parys, P. (2019).
Parity games: Zielonka's algorithm in quasi-polynomial time.
*CoRR*, abs/1904.12446.

van Dijk, T. (2018).
Oink: An implementation and evaluation of modern parity game solvers.
In Beyer, D. and Huisman, M., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 291–308, Cham. Springer International Publishing.

Zielonka, W. (1998).
Infinite games on finitely coloured graphs with applications to automata on infinite trees.
*Theoretical Computer Science*, 200(1):135 – 183.

# Parity games

Definition ([Zielonka, 1998][Grädel et al., 2002][?])

A parity game $(\mathcal{A}, Acc, v_0)$ is a two-players (called *Even* and *Odd*) infinite-duration zero-sum game with perfect information played on an "arena" (a.k.a. game graph) $\mathcal{A} = (\mathcal{V}, \mathcal{V}_{Even}, \mathcal{V}_{Odd}, \mathcal{E}, \mathcal{C}, \Omega)$

- $(\mathcal{V}, \mathcal{E})$ is a finite [a] graph such that $\mathcal{V} = \mathcal{V}_{Even} \cup \mathcal{V}_{Odd}$, $\mathcal{V}_{Even} \cap \mathcal{V}_{Odd} = \emptyset$ and $\forall v \in \mathcal{V}$, $\{v' \in \mathcal{V} | (v, v') \in \mathcal{E}\}$ is finite and non-empty,
- $\mathcal{C} = \{1, \ldots, c\}$ is the finite set of priorities (a.k.a. colours),
- $\Omega : \mathcal{V} \rightarrow \mathcal{C}$ is a priority function (a.k.a. colouring mapping) mapping each node to a priority (we assume that there are no "uncoloured" node),
- $v_0$ is the initial position of the token on the graph[b],
- $Acc = \{v_0 v_1 \ldots \in \mathcal{V}^\omega : \limsup_{i\to\infty} \Omega(v_i) \text{ is even}\}$ is the winning condition for Even.

[a] We assume finiteness but many works typically consider infinite graphs.

[b] Sometimes not in the definition of the game.