Parity Games

Sven Schewe

University of Liverpool

AVACS alumni presentation, September 29^{th}, 2015

Beautiful games you cannot stop playing

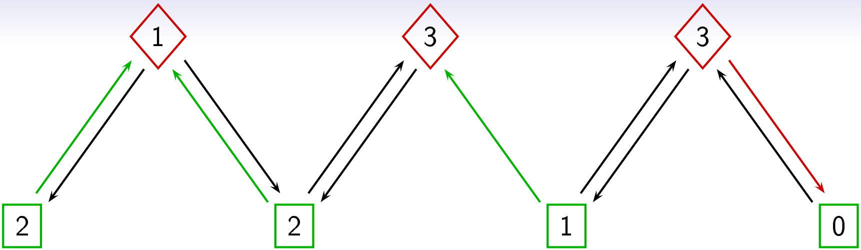
1. Parity Games with Few Colours
2. Parity Games with Many Colours
3. Parity Games with Few Colours
4. Parity Games with Few Colours
5. Parity Games with Few Colours
6. Parity Games with Bounded Treewidth
7. Strategy Improvement Algorithms

Beautiful games you cannot stop playing

1. Parity Games with Few Colours
2. Parity Games with Many Colours
3. Parity Games with Few Colours
4. Parity Games with Few Colours
5. Parity Games with Few Colours
6. Parity Games with Bounded Treewidth
7. Strategy Improvement Algorithms

Beautiful games you cannot stop playing

1. Parity Games with Few Colours
2. Parity Games with Many Colours
3. Parity Games with Few Colours
4. Parity Games with Few Colours
5. Parity Games with Few Colours
6. Parity Games with Bounded Treewidth
7. Strategy Improvement Algorithms



Parity Game $\mathcal{P} = \langle V_0, V_1, E, \alpha \rangle$

- V_0, and V_1 are disjoint finite sets of game positions
- $E \subseteq V_0 \cup V_1 \times V_0 \cup V_1$ is a set of edges, and
- $\alpha : V_0 \cup V_1 \to \mathbb{N}$ is a colouring function

Played by placing a pebble on the arena
– on V_0 player 0 chooses a successor, on V_1 player 1
$\Rightarrow$ infinite play, highest colour occurring infinite often
even $\rightsquigarrow$ player 0 wins, odd $\rightsquigarrow$ player 1 wins

Applications

- (non)emptiness game for parity tree automata
- acceptance game for parity tree automata
- satisfiability checking for CTL*, ATL*, μ-calculus, ATμC . . .
- open synthesis for LTL, CTL*, ATL*, μ-calculus, ATμC . . .

- μ-calculus model checking & extensions (e.g., graded μ-calculus, alternating-time μ-calculus)
- CTL* model checking (three colours), ATL* model checking
- module checking

Simple & Symmetric

Simple Reduction [Zwick+Paterson 96]

Parity Games

$\Downarrow$

Mean Payoff Games

$\Downarrow$

Discounted Payoff Games

$\Downarrow$

Simple Stochastic Games

Symmetric Problem

Until recently, only a single deterministic symmetric algorithm

Fixed Point, [Zwick+Paterson 96]

Obvious Facts and Open Questions

Obvious Facts

- symmetric

$\Rightarrow$ in class $\cap$ co-class

- **single** fixed point of DPG can be guessed

$\Rightarrow$ in UP $\cap$ co-UP [Jurdziński 00]

Less Obvious Facts

- PLS [Beckmann and Moller 08]
- $n^{O(\sqrt{n})}$ [Jurdziński, Zwick, and Paterson 08]
- PPAD [Etessami and Yannakakis 10]

Obvious Facts and Open Questions

Obvious Facts

- symmetric

$\Rightarrow$ in class $\cap$ co-class

- **single** fixed point of DPG can be guessed

$\Rightarrow$ in UP $\cap$ co-UP [Jurdziński 00]

Open Problems

- P?
- RP / ZPP?
- pay-off games: $2^{O(\sqrt{n})}$?, $2^{o(n)}$?

Overview

- Reachability Games
- Büchi Games
- Parity Games
  - McNaughton — few colours
  - Jurdziński, Paterson, and Zwick
  - Browne & al. / Jurdziński
  - their synthesis
- bounded tree-width & Co
- strategy improvement

Part I

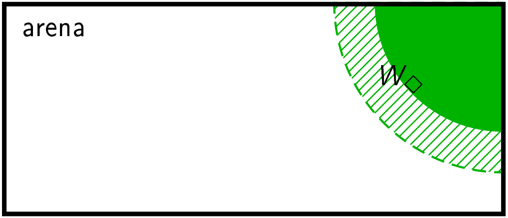
Reachability & Büchi Games

Solving Reachability Games

Algorithm – for $\mathcal{R} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set $W_\Diamond$ to $\Diamond$-attractor(F)
- set $W_\Box$ to $V \setminus W_\Diamond$

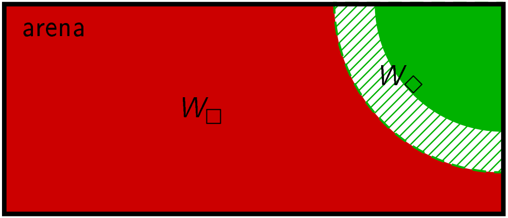
Solving Reachability Games



Algorithm – for $\mathcal{R} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set $W_\Diamond$ to $\Diamond$-attractor(F)
- set $W_\Box$ to $V \setminus W_\Diamond$

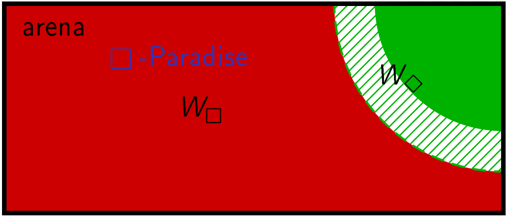
Solving Reachability Games



Algorithm – for $\mathcal{R} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set $W_\Diamond$ to $\Diamond$-attractor(F)
- set $W_\square$ to $V \setminus W_\Diamond$

Traps and Paradises



Traps and Paradises

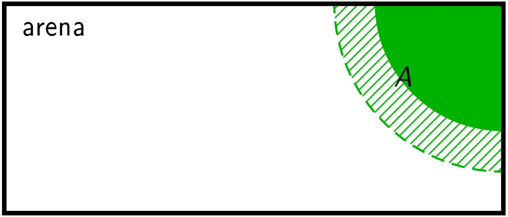
- A $\Diamond$ -trap is a set of states where $\Diamond$ cannot get out.
  E.g.: $W_\square$
- Remark: $W_\Diamond = W_\Diamond^\infty$ is usually no $\square$ -trap.
- A $\square$ -paradise is a $\Diamond$ -trap such that $\square$ can win without leaving it
  Example: $W_\square$

Solving Büchi Games

Algorithm – for $\mathcal{B} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set A to $\Diamond$-attractor(F)
- $U_\Box = V \setminus A$ is a $\Box$-paradise (strategy: stay there)
- $V_\Box = \Box$-attractor$(U_\Box)$ is a $\Box$-paradise (go to $U_\Box$, stay)
- $W_\Diamond$ for $\mathcal{B}$ is $W_\Diamond$ for $\mathcal{B} \setminus V_\Box$
- solve $\mathcal{B} \setminus V_\Box$

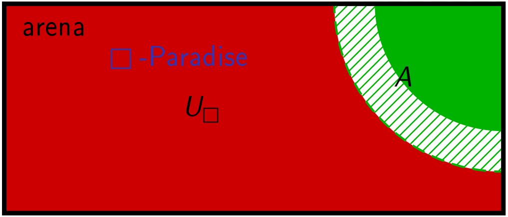
Solving Büchi Games



Algorithm – for $\mathcal{B} = \langle V_0, V_1, E, F\rangle$

- start with the final states F
- set A to $\Diamond$-attractor(F)
- $U_\Box = V \setminus A$ is a $\Box$-paradise (strategy: stay there)
- $V_\Box = \Box$-attractor($U_\Box$) is a $\Box$-paradise (go to $U_\Box$, stay)
- $W_\Diamond$ for $\mathcal{B}$ is $W_\Diamond$ for $\mathcal{B} \setminus V_\Box$
- solve $\mathcal{B} \setminus V_\Box$

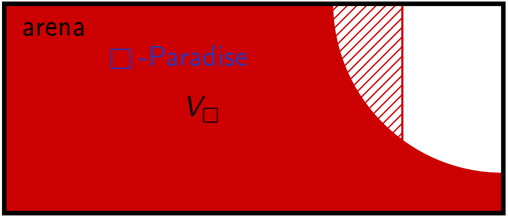
Solving Büchi Games



Algorithm – for $\mathcal{B} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set A to $\Diamond$-attractor(F)
- $U_\square = V \setminus A$ is a $\square$-paradise (strategy: stay there)
- $V_\square = \square$-attractor($U_\square$) is a $\square$-paradise (go to $U_\square$, stay)
- $W_\Diamond$ for $\mathcal{B}$ is $W_\Diamond$ for $\mathcal{B} \setminus V_\square$
- solve $\mathcal{B} \setminus V_\square$

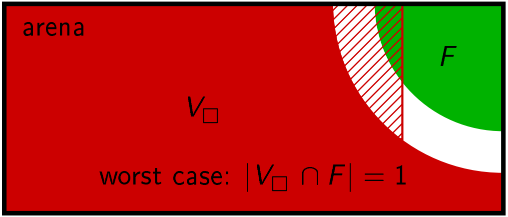
Solving Büchi Games



Algorithm – for $\mathcal{B} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set A to $\Diamond$-attractor(F)
- $U_\Box = V \setminus A$ is a $\Box$-paradise (strategy: stay there)
- $V_\Box = \Box$-attractor$(U_\Box)$ is a $\Box$-paradise (go to $U_\Box$, stay)
- $W_\Diamond$ for $\mathcal{B}$ is $W_\Diamond$ for $\mathcal{B} \setminus V_\Box$
- solve $\mathcal{B} \setminus V_\Box$

Solving Büchi Games



Remark

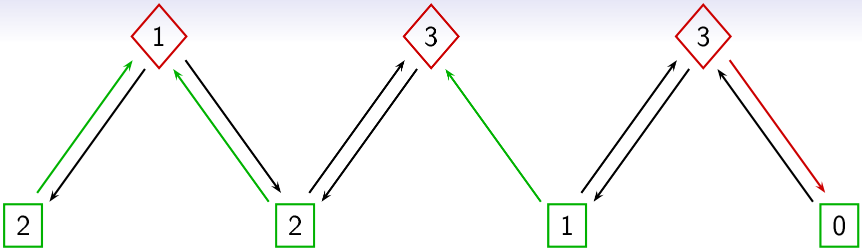
- 'outdated' approach
- $O(n^2)$ [Chaterjee and Henzinger 12]

Part II

Parity Games

Overview

# colours	3	4	5	6	7	8
McNaughton	$O(m\,n^2)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^5)$	$O(m\,n^6)$	$O(m\,n^7)$
Browne & al.	$O(m\,n^3)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^4)$	$O(m\,n^5)$	$O(m\,n^5)$
Jurdziński	$O(m\,n^2)$	$O(m\,n^2)$	$O(m\,n^3)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^4)$
w.o. strategy / [GW15]	$O(m\,n)$		$O(m\,n^2)$		$O(m\,n^3)$	
Big Steps [S07]	$O(m\,n)$	$O(m\,n^{1\frac{1}{2}})$	$O(m\,n^2)$	$O(m\,n^{2\frac{1}{3}})$	$O(m\,n^{2\frac{3}{4}})$	$O(m\,n^{3\frac{1}{16}})$
[CHL15]	$O(n^{2.5})$	$O(n^3)$	$O(n^{3\frac{1}{3}})$	$O(n^{3\frac{3}{4}})$	$O(n^{4\frac{1}{16}})$	$O(n^{4\frac{9}{20}})$



Parity Game $\mathcal{P} = \langle V_0, V_1, E, \alpha \rangle$

- V_0, and V_1 are disjoint finite sets of game positions
- $E \subseteq V_0 \cup V_1 \times V_0 \cup V_1$ is a set of edges, and
- $\alpha : V_0 \cup V_1 \to \mathbb{N}$ is a colouring function

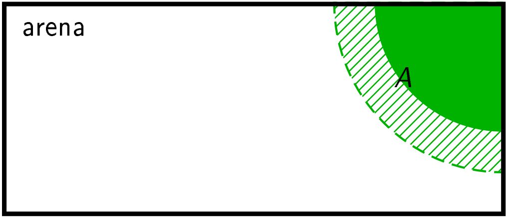
Played by placing a pebble on the arena
– on V_0 player 0 chooses a successor, on V_1 player 1
$\Rightarrow$ infinite play, highest colour occurring infinite often
even $\rightsquigarrow$ player 0 wins, odd $\rightsquigarrow$ player 1 wins

McNaughton's Algorithm

McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal colour, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$(\alpha^{-1}(c))$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

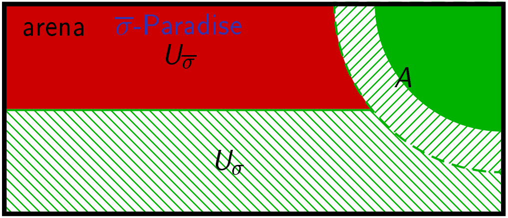
McNaughton's Algorithm



McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal colour, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

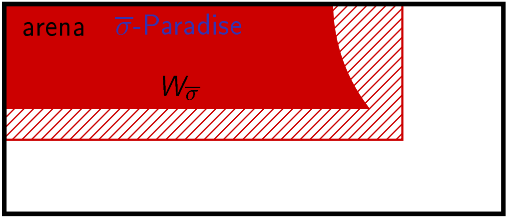
McNaughton's Algorithm



McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal colour, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot\cup U_0, W_1 \dot\cup U_1)$

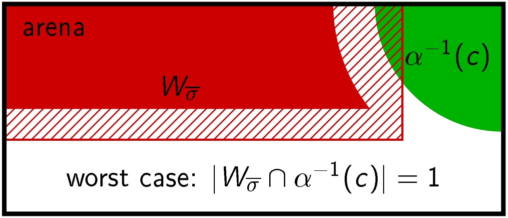
McNaughton's Algorithm



McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal colour, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot\cup U_0, W_1 \dot\cup U_1)$

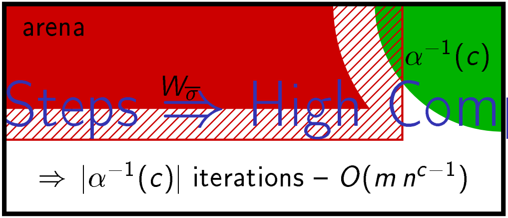
McNaughton's Algorithm



McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal colour, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$(\alpha^{-1}(c))$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

McNaughton's Algorithm—Weakness

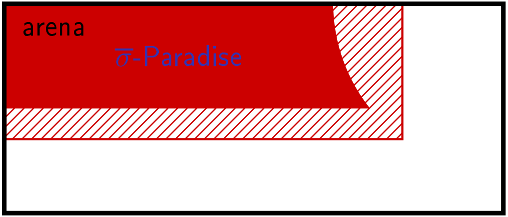


Small Steps $\Rightarrow$ High Complexity

McNaughton's Algorithm – for $P = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal colour, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- set A to σ-attractor$\big(\alpha^{-1}(c)\big)$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}})$, and set W_σ to $\emptyset$
- set (U_0, U_1) to McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

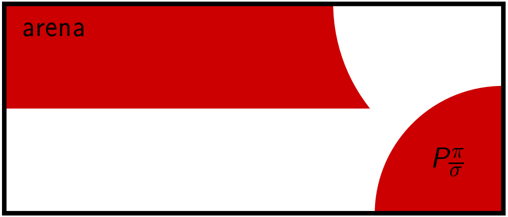
σ-Paradise



Definition – σ-Paradise

- Subset P_σ of the positions, s.t. player σ has a strategy to
  - stay in P_σ ($\overline{\sigma}$-trap)
  - that is winning for all states in P_σ.
- σ-Paradises are closed under
  - union, and
  - σ-attractor.

σ/π-Paradise



Definition – σ/π-Paradise

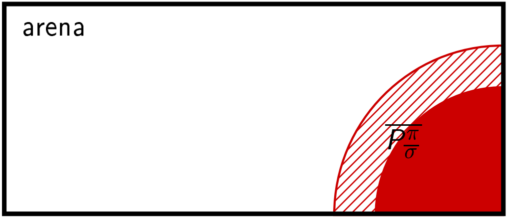
- Paradise P_σ^π that contains all σ-paradises of size $\leq \pi$.
- σ/π-Paradises are closed under
  - union with any σ-paradise, and
  - σ-attractor.

Big-Step Algorithm

BigStep Algorithm – for $\mathcal{P} = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal color, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- **compute $\overline{\sigma}/\pi$-paradise $P^{\pi}_{\overline{\sigma}}$, and set $\overline{P^{\pi}_{\overline{\sigma}}}$ to $\overline{\sigma}$-attractor$(P^{\pi}_{\overline{\sigma}})$**
- **set $\mathcal{P}'$ to $\mathcal{P} \setminus \overline{P^{\pi}_{\overline{\sigma}}}$**
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to BigStep$(\mathcal{P}' \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}}) \cup \overline{P^{\pi}_{\overline{\sigma}}}$, and set W_σ to $\emptyset$
- set (U_0, U_1) to BigStep$(\mathcal{P} \setminus W_{\overline{\sigma}})$, return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

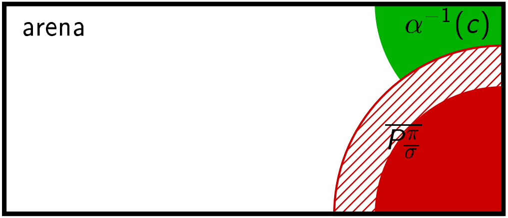
Big-Step Algorithm



BigStep Algorithm – for $\mathcal{P} = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal color, σ to c modulo 2, and $\overline{\sigma}$ to $1-\sigma$
- **compute $\overline{\sigma}/\pi$-paradise $P^{\pi}_{\overline{\sigma}}$, and set $\overline{P^{\pi}_{\overline{\sigma}}}$ to $\overline{\sigma}$-attractor$(P^{\pi}_{\overline{\sigma}})$**
- **set $\mathcal{P}'$ to $\mathcal{P} \setminus \overline{P^{\pi}_{\overline{\sigma}}}$**
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to BigStep$(\mathcal{P}' \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}}) \cup \overline{P^{\pi}_{\overline{\sigma}}}$, and set W_σ to $\emptyset$
- set (U_0, U_1) to BigStep$(\mathcal{P} \setminus W_{\overline{\sigma}})$, return $(W_0 \dot\cup U_0, W_1 \dot\cup U_1)$

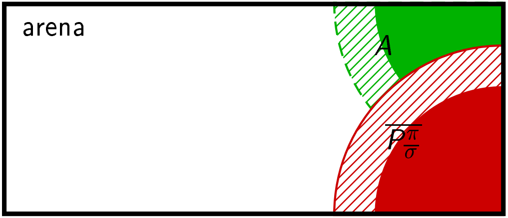
Big-Step Algorithm



BigStep Algorithm – for $\mathcal{P} = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal color, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- **compute $\overline{\sigma}/\pi$-paradise $P^{\pi}_{\overline{\sigma}}$, and set $\overline{P^{\pi}_{\overline{\sigma}}}$ to $\overline{\sigma}$-attractor$(P^{\pi}_{\overline{\sigma}})$**
- **set $\mathcal{P}'$ to $\mathcal{P} \setminus \overline{P^{\pi}_{\overline{\sigma}}}$**
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to BigStep$(\mathcal{P}' \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}}) \cup \overline{P^{\pi}_{\overline{\sigma}}}$, and set W_σ to $\emptyset$
- set (U_0, U_1) to BigStep$(\mathcal{P} \setminus W_{\overline{\sigma}})$, return $(W_0 \dot\cup U_0, W_1 \dot\cup U_1)$

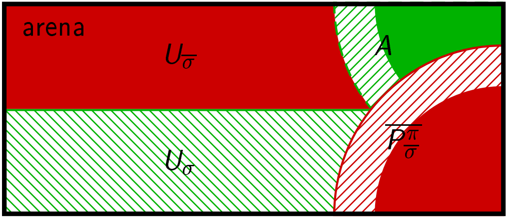
Big-Step Algorithm



BigStep Algorithm – for $\mathcal{P} = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal color, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- **compute $\overline{\sigma}/\pi$-paradise $P_{\overline{\sigma}}^{\pi}$, and set $\overline{P_{\overline{\sigma}}^{\pi}}$ to $\overline{\sigma}$-attractor$(P_{\overline{\sigma}}^{\pi})$**
- **set $\mathcal{P}'$ to $\mathcal{P} \setminus \overline{P_{\overline{\sigma}}^{\pi}}$**
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to BigStep$(\mathcal{P}' \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}}) \cup \overline{P_{\overline{\sigma}}^{\pi}}$, and set W_σ to $\emptyset$
- set (U_0, U_1) to BigStep$(\mathcal{P} \setminus W_{\overline{\sigma}})$, return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

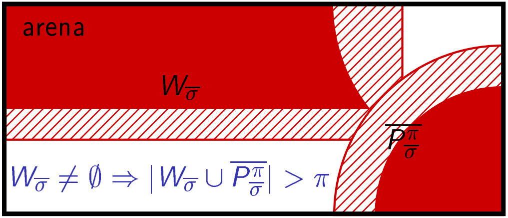
Big-Step Algorithm



BigStep Algorithm – for $\mathcal{P} = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal color, σ to c modulo 2, and $\overline{\sigma}$ to $1 - \sigma$
- **compute $\overline{\sigma}/\pi$-paradise $P^\pi_{\overline{\sigma}}$, and set $\overline{P^\pi_{\overline{\sigma}}}$ to $\overline{\sigma}$-attractor$(P^\pi_{\overline{\sigma}})$**
- set $\mathcal{P}'$ to $\mathcal{P} \setminus \overline{P^\pi_{\overline{\sigma}}}$
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to BigStep$(\mathcal{P}' \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}}) \cup \overline{P^\pi_{\overline{\sigma}}}$, and set W_σ to $\emptyset$
- set (U_0, U_1) to BigStep$(\mathcal{P} \setminus W_{\overline{\sigma}})$, return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

Big-Step Algorithm



BigStep Algorithm – for $\mathcal{P} = \langle V_0, V_1, E, \alpha \rangle$

- set c to the maximal color, σ to c modulo 2, and $\overline{\sigma}$ to $1-\sigma$
- **compute $\overline{\sigma}/\pi$-paradise $P^{\pi}_{\overline{\sigma}}$, and set $\overline{P^{\pi}_{\overline{\sigma}}}$ to $\overline{\sigma}$-attractor$(P^{\pi}_{\overline{\sigma}})$**
- set $\mathcal{P}'$ to $\mathcal{P} \setminus \overline{P^{\pi}_{\overline{\sigma}}}$
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to BigStep$(\mathcal{P}' \setminus A)$
- set $W_{\overline{\sigma}}$ to $\overline{\sigma}$-attractor$(U_{\overline{\sigma}}) \cup \overline{P^{\pi}_{\overline{\sigma}}}$, and set W_{σ} to $\emptyset$
- set (U_0, U_1) to BigStep$(\mathcal{P} \setminus W_{\overline{\sigma}})$, return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

Jurdziński, Paterson, and Zwick

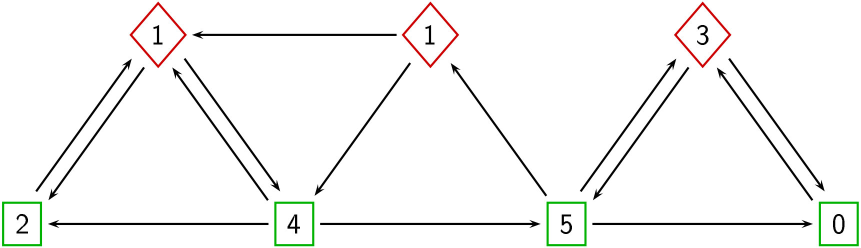
- invented this approache
- used it to establish a deterministic $n^{O(\sqrt{n})}$ bound

Brute Force (roughly)

- try all sets of size up to $\pi \in O(\sqrt{n})$
- there are some $n^{O(\sqrt{n})}$ many
- each level has up to $O(\sqrt{n})$ many calls
- call tree of size $n^{O(\sqrt{n})}$

drawback: c is, in fact, usually tiny compared to $\sqrt{n}$

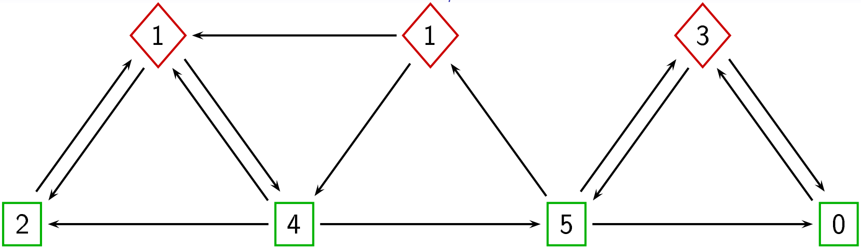
Browne & al., Jurdziński



If you follow a winning strategy of even on W_0, then . . .

- player odd cannot force $> |\alpha^{-1}(c)|$ occurences of any odd colour c without a higher even colour in between
- player even can force $> |\alpha^{-1}(c)|$ occurences of some (not a particular!) even colour c without a higher odd colour in between

Browne & al., Jurdziński



Rules: Jurdziński: backwards, order on counter vector

- we start at some **initial positions** with counters for, say, the odd colours only, inially **set to** 0
- each player chooses how to continue on her vertices
- if we pass an odd colour c, the counter is increased
- if we pass an even colour c, all counters for **smaller** colours are **re-set**
- player odd wins if a counter exceeds $|\alpha^{-1}(c)|$

Big Steps – What if c is Small?

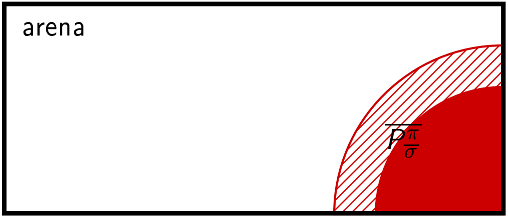
— the common case —

Stop counting at π (simple!)

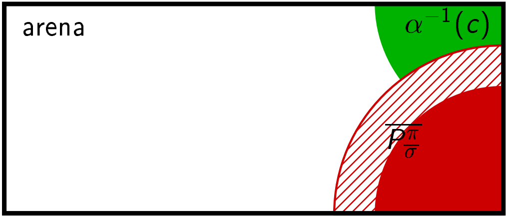
- $\lceil 0.5c \rceil$ many counters
- their sum bounded by π
- $\leq \binom{\pi + \lceil 0.5c \rceil}{\pi} \approx \underline{\pi^{\lceil 0.5c \rceil}}$ values
- covers all σ-paradises $P_{\overline{\sigma}}$ with $|P_{\overline{\sigma}}| \leq \pi$
- Complexity: $O\left(c\, m\, \pi^{\lceil 0.5c \rceil}\right)$

Big-Step Algorithm

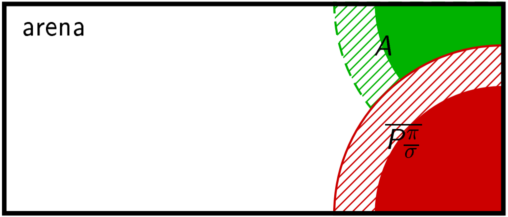
Big-Step Algorithm



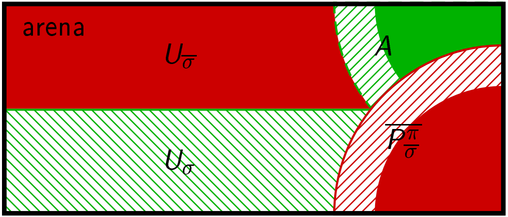
Big-Step Algorithm



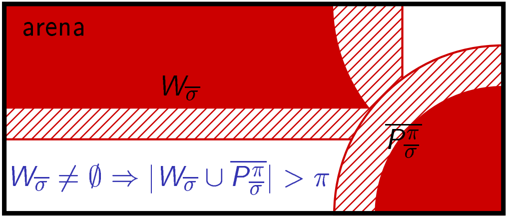
Big-Step Algorithm



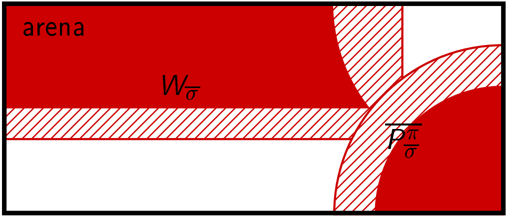
Big-Step Algorithm



Big-Step Algorithm



Solving Parity Games in Big Steps – Complexity



number of colours	3	4	5	6	7	8
paradise construction	-	$O(m\,n)$	$O(m\,n^{1\frac{1}{2}})$	$O(m\,n^2)$	$O(m\,n^{2\frac{1}{3}})$	$O(m\,n^{2\frac{3}{4}})$
chosen parameter $\pi_c(n)$	-	$n^{\frac{1}{2}}$	$n^{\frac{1}{2}}$	$n^{\frac{2}{3}}$	$n^{\frac{7}{12}}$	$n^{\frac{11}{16}}$
number of iterations $\frac{n}{\pi_c(n)}$	-	$n^{\frac{1}{2}}$	$n^{\frac{1}{2}}$	$n^{\frac{1}{3}}$	$n^{\frac{5}{12}}$	$n^{\frac{5}{16}}$
solving complexity	$O(m\,n)$	$O(m\,n^{1\frac{1}{2}})$	$O(m\,n^2)$	$O(m\,n^{2\frac{1}{3}})$	$O(m\,n^{2\frac{3}{4}})$	$O(m\,n^{3\frac{1}{16}})$

State of the Art

# colours	3	4	5	6	7	8
McNaughton	$O(m\,n^2)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^5)$	$O(m\,n^6)$	$O(m\,n^7)$
Browne & al.	$O(m\,n^3)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^4)$	$O(m\,n^5)$	$O(m\,n^5)$
Jurdziński	$O(m\,n^2)$	$O(m\,n^2)$	$O(m\,n^3)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^4)$
w.o. strategy / [GW15]	$O(m\,n)$		$O(m\,n^2)$		$O(m\,n^3)$	
Big Steps [S07]	$O(m\,n)$	$O(m\,n^{1\frac{1}{2}})$	$O(m\,n^2)$	$O(m\,n^{2\frac{1}{3}})$	$O(m\,n^{2\frac{3}{4}})$	$O(m\,n^{3\frac{1}{16}})$
[CHL15]	$O(n^{2.5})$	$O(n^3)$	$O(n^{3\frac{1}{3}})$	$O(n^{3\frac{3}{4}})$	$O(n^{4\frac{1}{16}})$	$O(n^{4\frac{9}{20}})$

- Significantly improved complexity bound
  - from $O\left(c\,m\left(\frac{n}{\lfloor 0.5c \rfloor}\right)^{\lfloor 0.5\,c \rfloor}\right)$ to $O\left(m\left(\frac{\kappa\,n}{c}\right)^{\gamma(c)}\right)$ for
    $\gamma(c) = \frac{1}{3}c + \frac{1}{2} - \frac{1}{3c} - \frac{1}{\lceil \frac{c}{2} \rceil \lfloor \frac{c}{2} \rfloor}$ if c is even, and
    $\gamma(c) = \frac{1}{3}c + \frac{1}{2} - \frac{1}{\lceil \frac{c}{2} \rceil \lfloor \frac{c}{2} \rfloor}$ if c is odd
- Second improvement that reduces the growth in # colours

Part III

Bounded Treewidth & Co

Other Parameter

Parity games are in P for other parameters than # colours

- tree-width [Obdrzálek 03]
- DAG-width [Berwanger, Dawar, Hunter, and Kreutzer 06]
- clique-width [Obdrzálek 07]

Hope

Can this be a foundation for a tractable algorithm?

A 'Positive' Result

Fearnley and Schewe 2013

- NC^2 for bounded tree-width k
+ improved bound $O\left(n\, c^{2(k+1)^2}\right) \rightsquigarrow O\left((n\, k^2\, k!(c+1)^{3k+1}\right)$
+ fixed parameter tractable for bounded DAG-width

Improved by Ganardi 2015

- LogCFL for bounded tree-width
- LogCFL for bounded cleaque-width
- LogDCFL for tree-width 2

A 'Positive' Result

Fearnley and Schewe 2013

- NC^2 for bounded tree-width k

+ improved bound $O\big(n\, c^{2(k+1)^2}\big) \rightsquigarrow O\big((n\, k^2\, k!(c+1)^{3k+1}\big)$
+ fixed parameter tractable for bounded DAG-width

Improved by Ganardi 2015

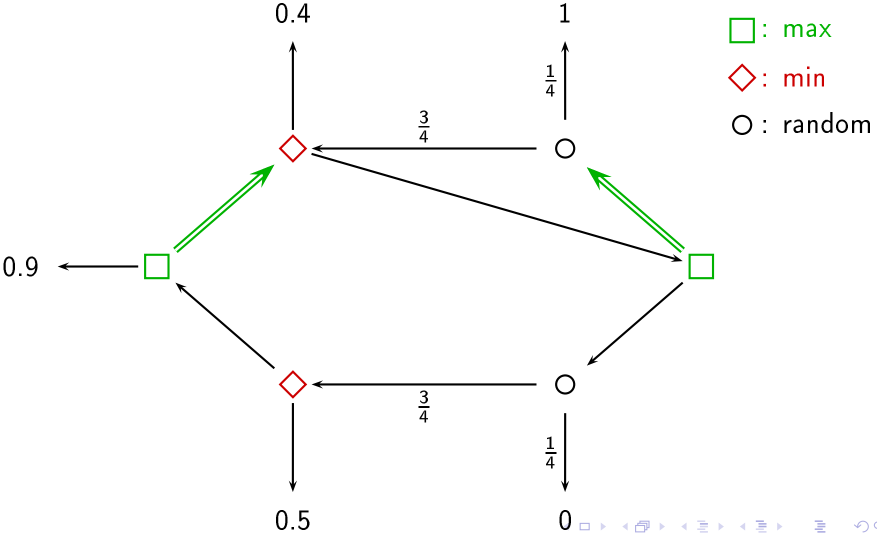
- LogCFL for bounded tree-width
- LogCFL for bounded cleaque-width
- LogDCFL for tree-width 2

Part IV

Strategy Improvement

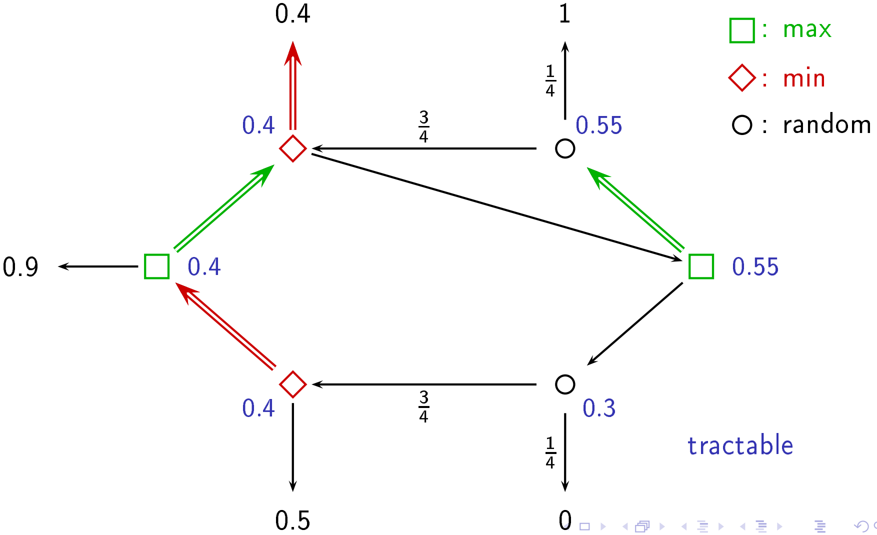
Classic Strategy Improvement

fix strategy



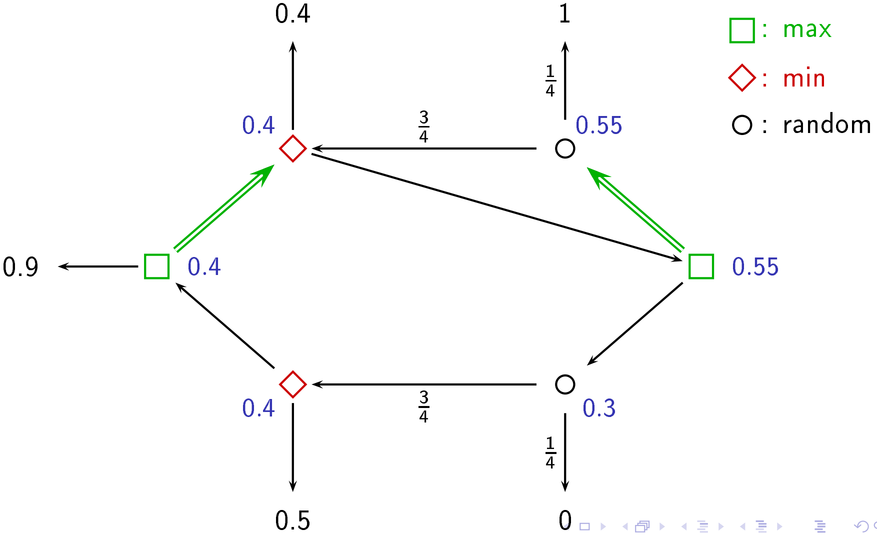
Classic Strategy Improvement

find best response and evaluate



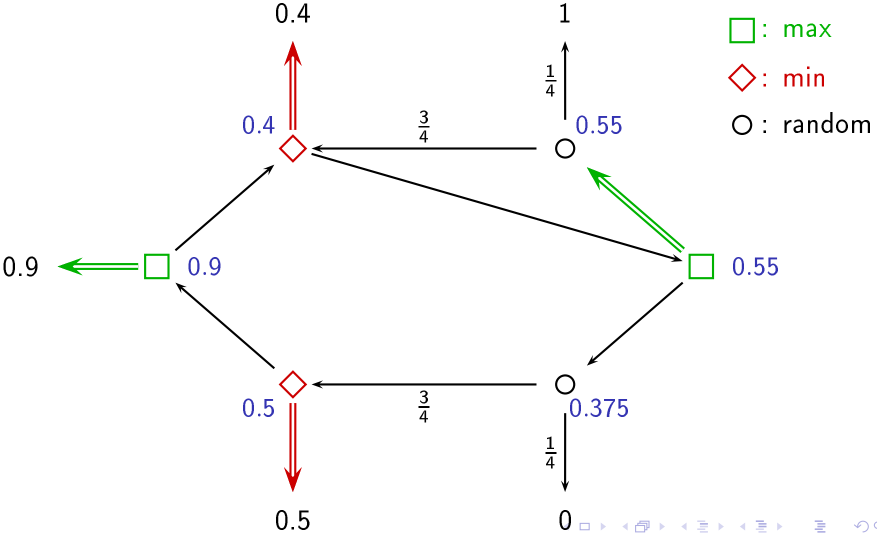
Classic Strategy Improvement

apply local improvements



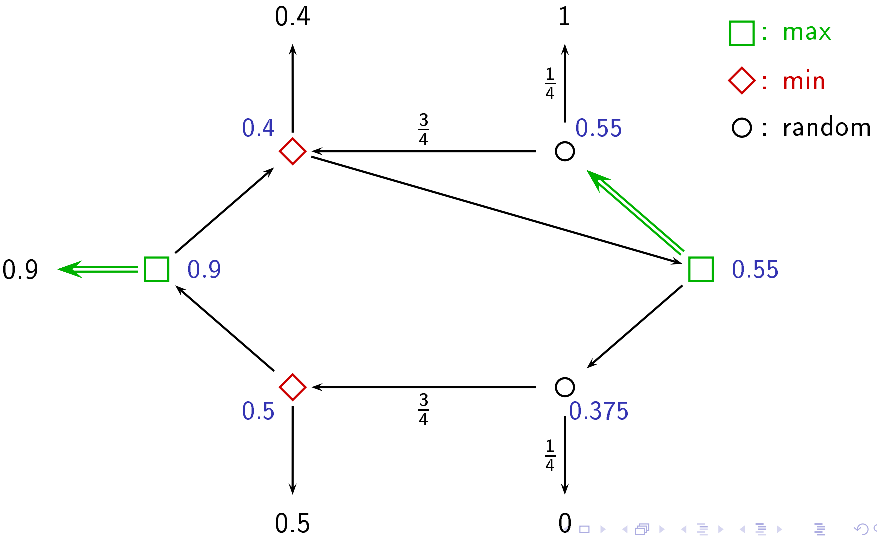
Classic Strategy Improvement

find best response & evaluate



Classic Strategy Improvement

no local improvent: done



CSI – failed hope

- was long hoped to be tractable
- many update policies

∀ exponential lower bounds [Friedmann 11,. . .]
– use static update policy

∃ PSPACE powerful [Fearnley+Savani 15]

SYMMETRYЯTƎMMYS

Symmetry and Complexity [Jurdziński 98]

1. guess valuation
2. verify

⇒ one value: UP
symmetry: UP∩CoUP

Iterated Fixed Point [Emerson+Lei 86] parity games

- similar treatment
- best performing algorithm

Optimal Strategy Improvement [Schewe 08]
parity games, MPG mean partitions

- some symmetry
- fab performance

Why not?

Naive symmetric strategy improvement

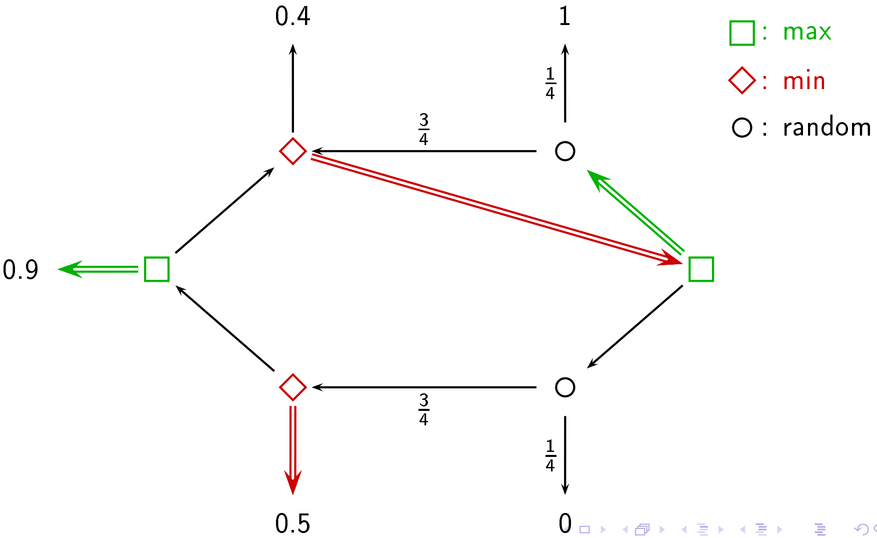
Question: Why has SSI not been thoroughly studied?

Answer: Anne Condon has proved it wrong [Condon 93]

1. Cuncurrent Switch
2. Alternating Best Response

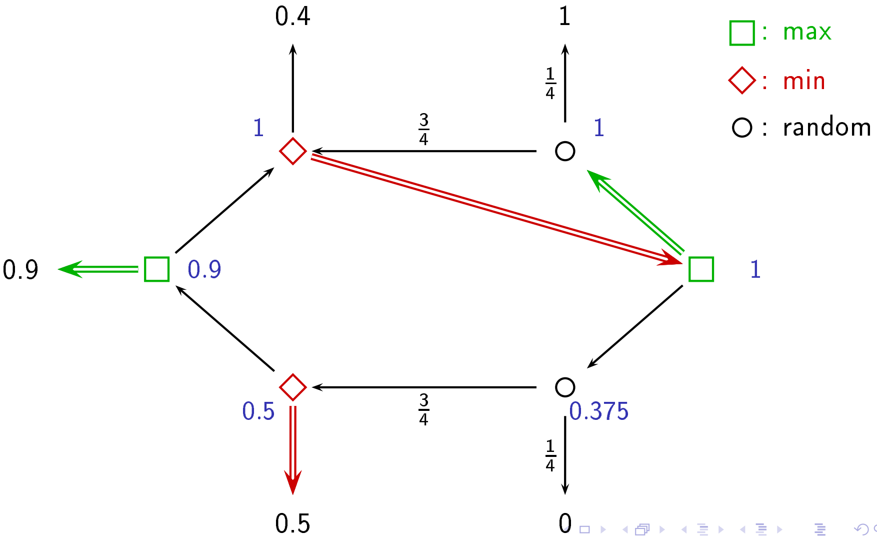
Concurrent Switch

starting strategies



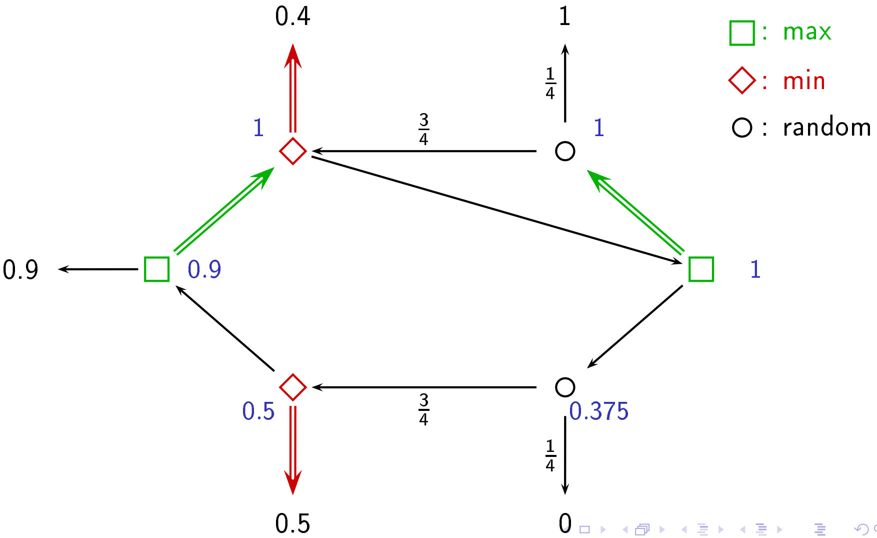
Concurrent Switch

evaluate



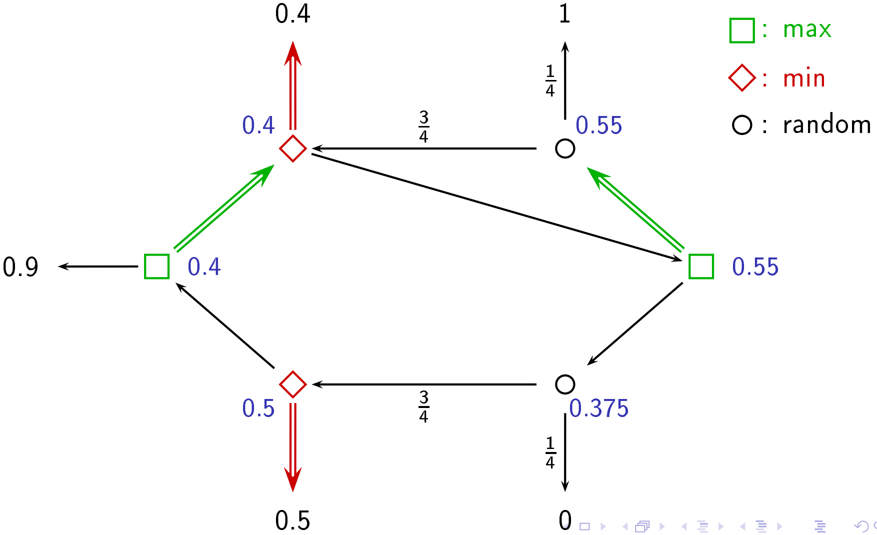
Concurrent Switch

update strategies



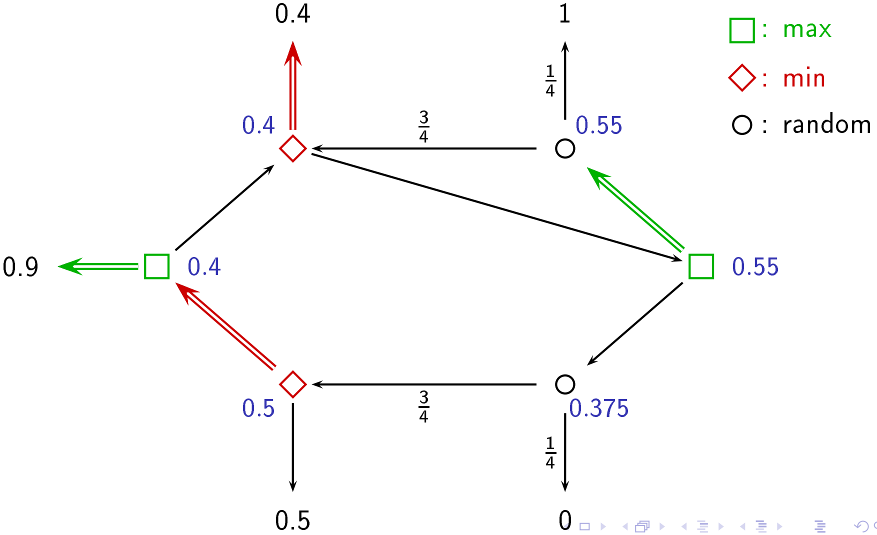
Concurrent Switch

update evaluation



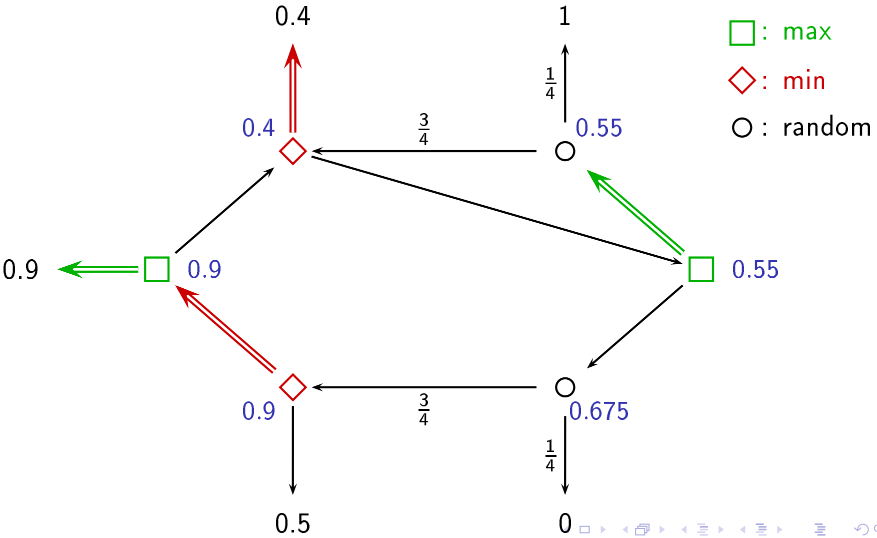
Concurrent Switch

update strategies



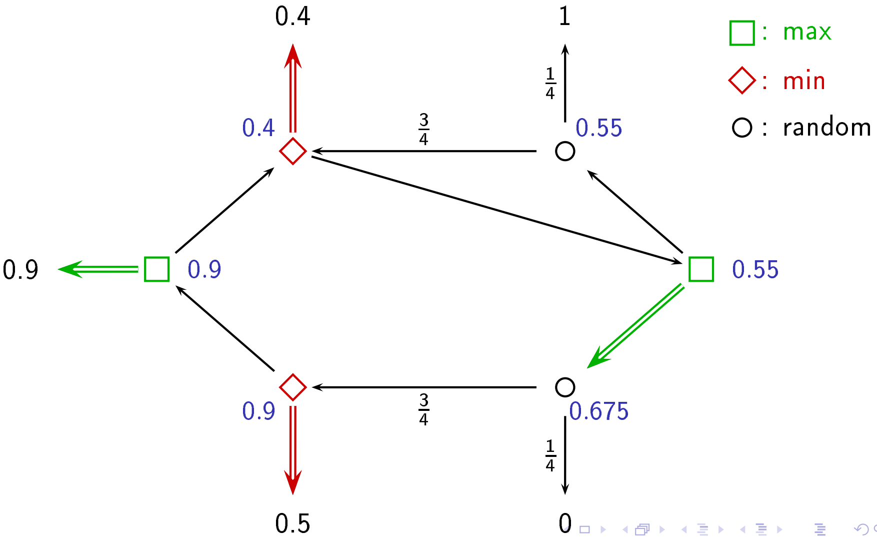
Concurrent Switch

update evaluation



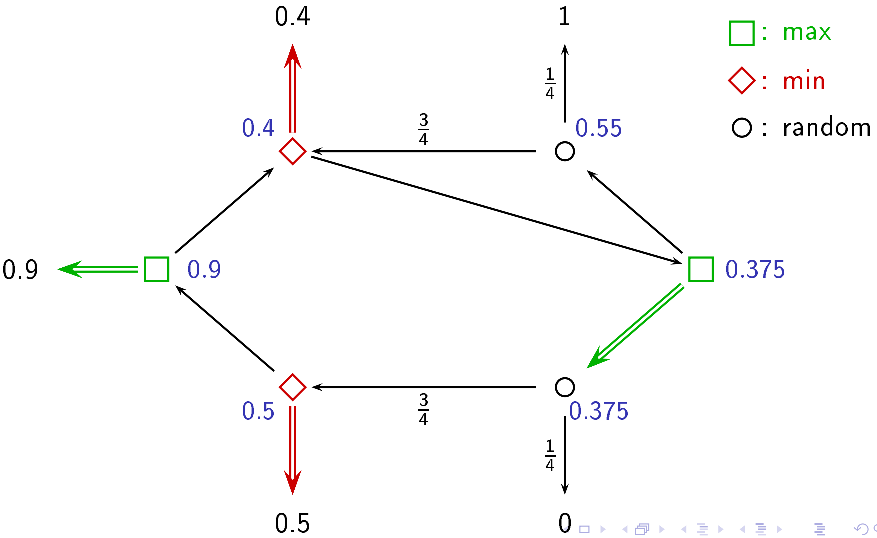
Concurrent Switch

update strategy



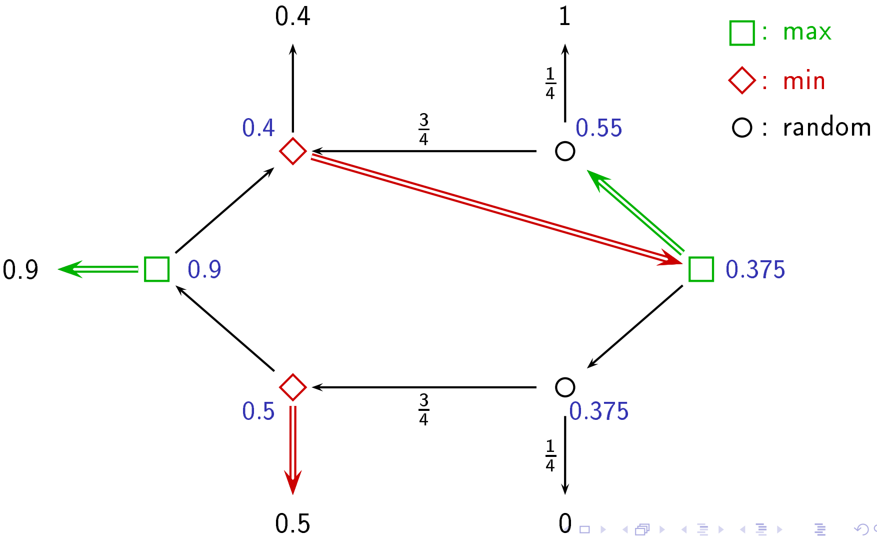
Concurrent Switch

update evaluation



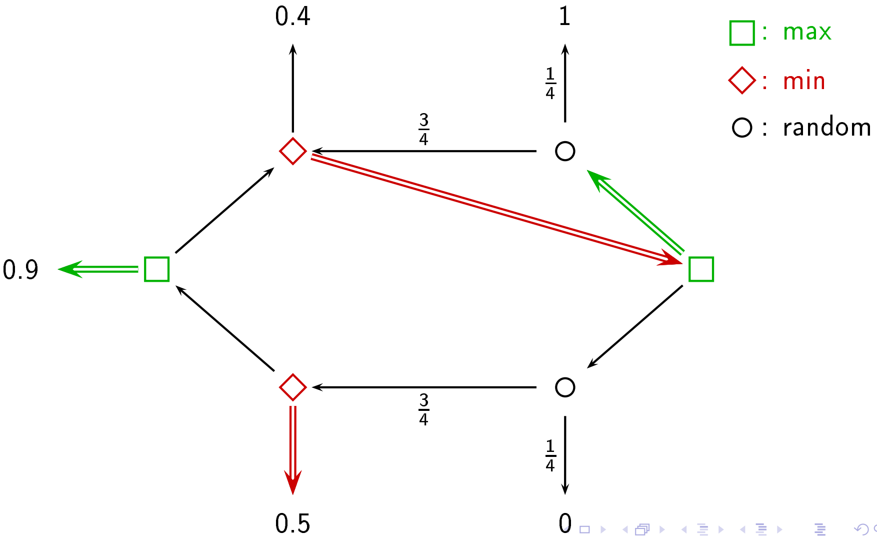
Concurrent Switch

update strategy (cycle)



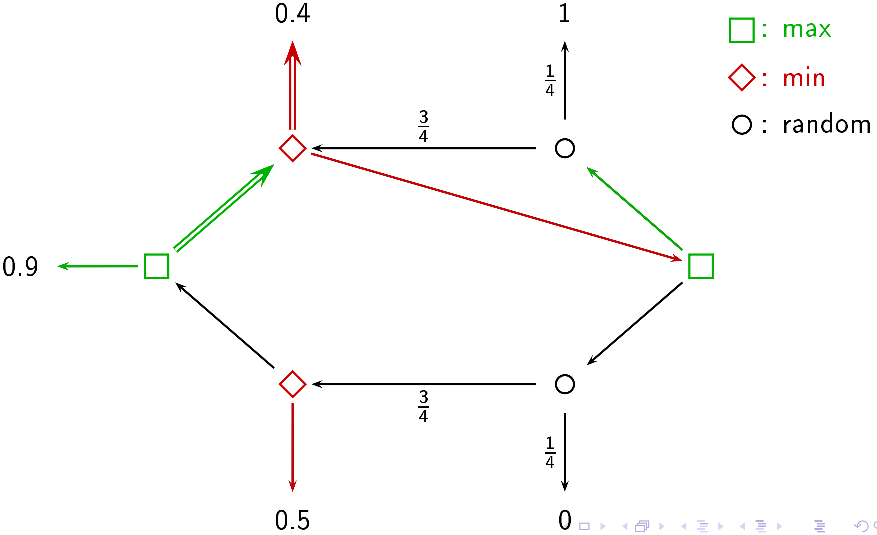
Symmetric Strategy Improvement

starting strategies



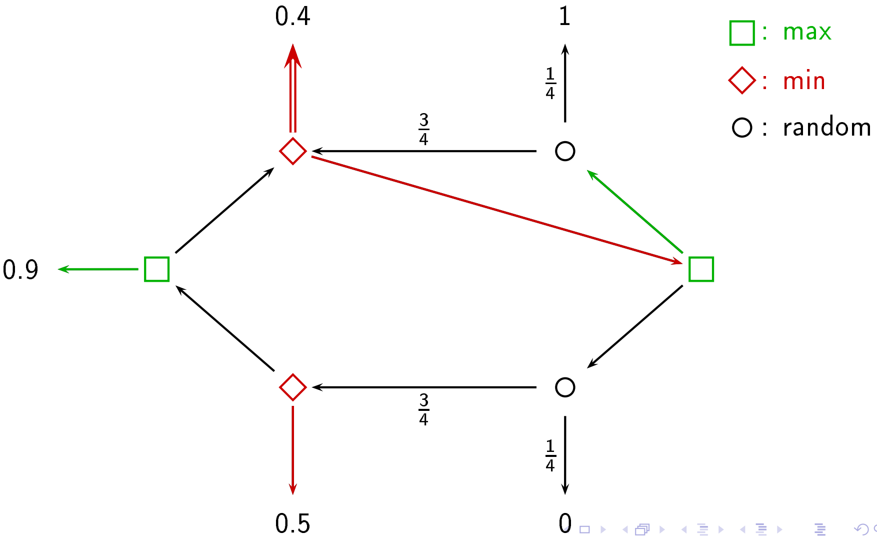
Symmetric Strategy Improvement

evaluate – best response



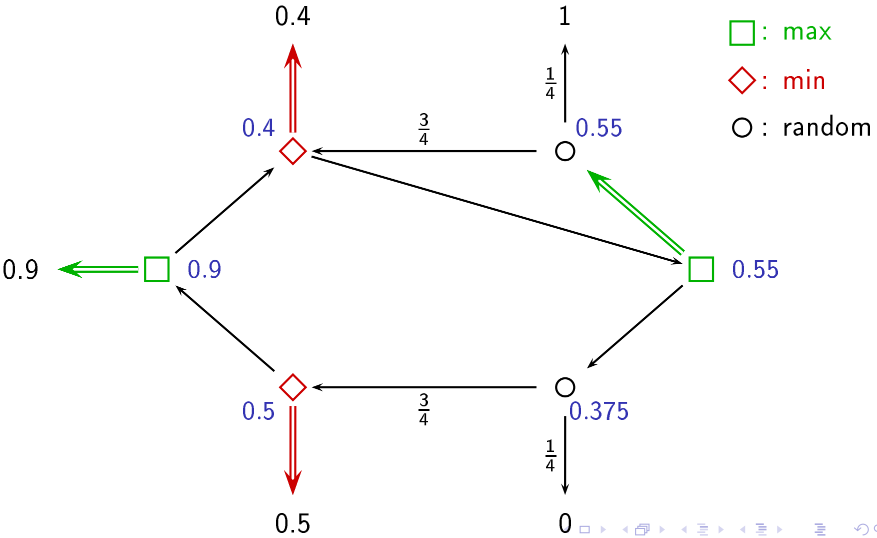
Symmetric Strategy Improvement

best response & improvement



Symmetric Strategy Improvement

update (done)



Symmetric Strategy Improvement

Can SSI help overcome problems of CSI?

Question: How about single player examples? [Fearnley 10]

Answer: Easy (but no surprise there)

Question: How about Friedmann's traps? [Friedmann 11,...]

Answer: Yes but this doesn't imply there are no traps

Question: Less iterations on random games?

Answer: Yes but probably not half

Question: Is SSI polynomial?

Answer: Look at the weather! Isn't it lovely?

Friedmann's Traps

Switch Rule	1	2	3	4	5	6	7	8	9	10
Cunningham	2	6	9	12	15	18	21	24	27	30
CunninghamSubexp	1	1	1	1	1	1	1	1	1	1
FearnleySubexp	4	7	11	13	17	21	25	29	33	37
FriedmannSubexp	4	9	13	15	19	23	27	31	35	39
RandomEdgeExpTest	1	2	2	2	2	2	2	2	2	2
RandomFacetSubexp	1	2	7	9	11	13	15	17	19	21
SwitchAllBestExp	4	5	8	11	12	13	15	17	18	19
SwitchAllBestSubExp	5	7	9	11	13	15	17	19	21	23
SwitchAllSubExp	3	5	7	9	10	11	12	13	14	15
SwitchAllExp	3	4	6	8	10	11	12	14	16	18
ZadehExp	-	6	10	14	18	21	25	28	32	35
ZadehSubexp	5	9	13	16	20	23	27	30	34	37

Parity Games

with few colours

# colours	3	4	5	6	7	8
McNaughton	$O(m\,n^2)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^5)$	$O(m\,n^6)$	$O(m\,n^7)$
Browne & al.	$O(m\,n^3)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^4)$	$O(m\,n^5)$	$O(m\,n^5)$
Jurdziński	$O(m\,n^2)$	$O(m\,n^2)$	$O(m\,n^3)$	$O(m\,n^3)$	$O(m\,n^4)$	$O(m\,n^4)$
w.o. strategy / [GW15]	$O(m\,n)$		$O(m\,n^2)$		$O(m\,n^3)$	
Big Steps [S07]	$O(m\,n)$	$O(m\,n^{1\frac{1}{2}})$	$O(m\,n^2)$	$O(m\,n^{2\frac{1}{3}})$	$O(m\,n^{2\frac{3}{4}})$	$O(m\,n^{3\frac{1}{16}})$
[CHL15]	$O(n^{2.5})$	$O(n^3)$	$O(n^{3\frac{1}{3}})$	$O(n^{3\frac{3}{4}})$	$O(n^{4\frac{1}{16}})$	$O(n^{4\frac{9}{20}})$

Parity Games

Further complexity resuts

- NP∩CoNP [NcNaughton 93]
- UP∩CoUP [Zwick and Paterson '96, Jurdziński 98]
- PLS [Beckmann and Moller 08]
- PPAD [Etessami and Yannakakis 10]
- $n^{O(\sqrt{n})}$ [Jurdziński, Zwick, and Paterson 08]
- in LogCFL for bounded tree- and clique-width [Ganardi 15]
- fixed parameter tractable for bounded DAG-width

Parity & Pay-Off Games

Strategy Improvement

- deterministic update [Puri 95, Vöge and Jurdziński 00]
- randomised updates [Ludwig 95, Björklund and Vorobyov 07]
- one-step optimal updates [S 08]
- they are all expensive [Friedmann 09, FHZ 11a]
- symmetric strategy improvement [STV 15]

Parity Games

. . . are simply **beautiful**!