A Simple Algorithm for Solving Qualitative Probabilistic Parity Games

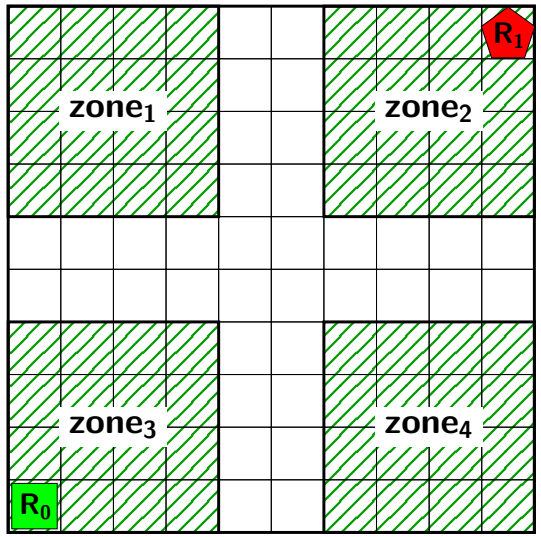
Sven Schewe

University of Liverpool

joint work with Ernst Moritz Hahn, Andrea Turrini & Lijun Zhang

Probabilistic Parity Games

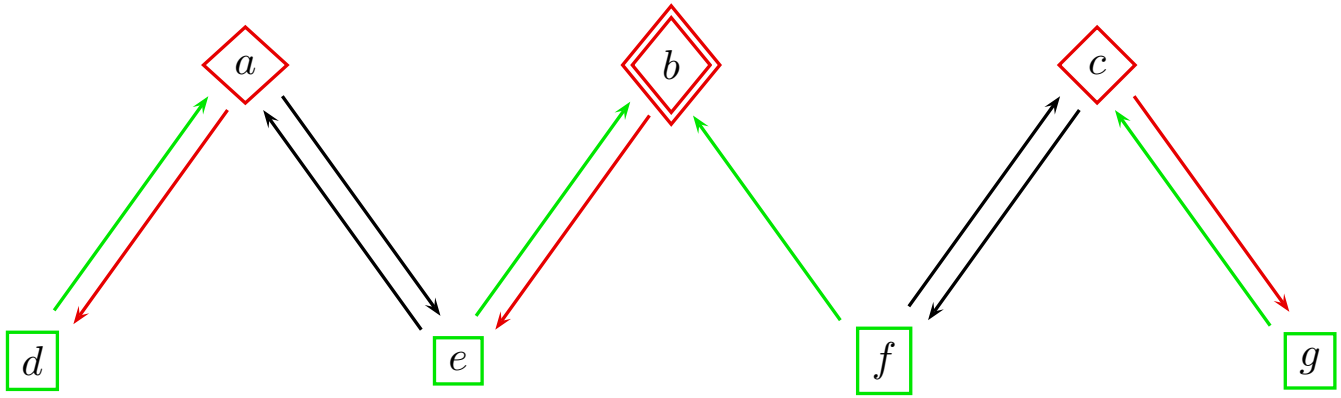
- Good abstraction for systems with
  - choices under our control
  - choices under environment control
  - randomised choices
- Suitable for linear time properties



$\langle\langle R_0 \rangle\rangle \mathcal{P}_{\geq 1}[\mathbf{GF}(\mathrm{zone}_1 \wedge \mathbf{F}\mathrm{zone}_2)]$

- Our goal is to win the game with probability 1.

Reachability Games

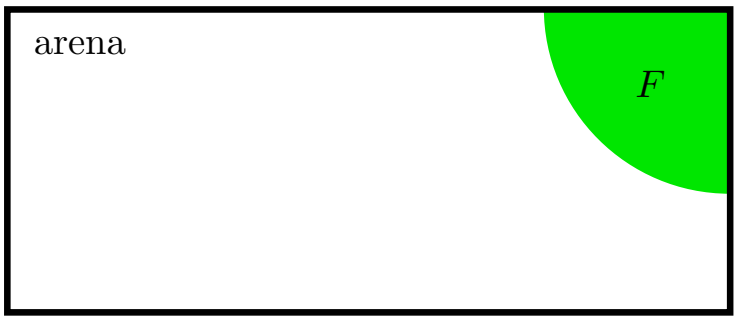


Reachability Game $\mathcal{R} = \langle V_0, V_1, E, F \rangle$

- 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
- $F \subseteq (V_0 \cup V_1)$ is a set of final / accepting game positions

Played by placing a pebble on the arena
– on V_0 player 0 chooses a successor, on V_1 player 1
$\Rightarrow$ infinite play π $\qquad S = (V_0 \cup V_1) \setminus F$
$\pi \notin S^\omega \rightsquigarrow$ player 0 wins, $\pi \in S^\omega \rightsquigarrow$ player 1 wins

Solving Reachability Games



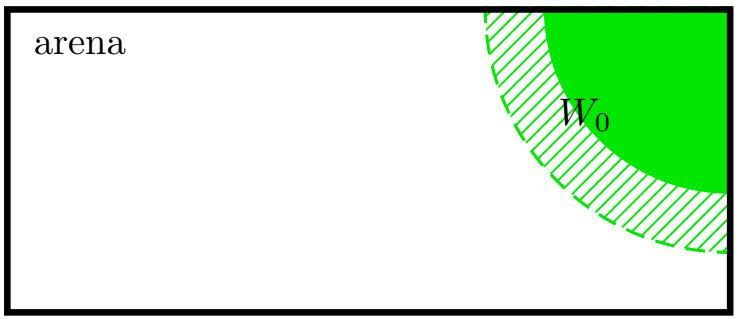
Algorithm – for $\mathcal{R} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set W_0 to 0-attractor(F)
- set W_1 to $V \setminus W_0$

σ-attractor(X): fix-point of X union

- for player σ: positions in V_σ that can reach X
- for player $1 - \sigma$: positions in $V_{1-\sigma}$ that can only reach X

Solving Reachability Games



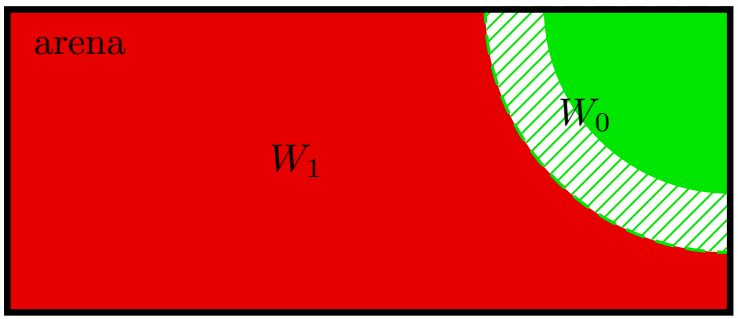
Algorithm – for $\mathcal{R} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set W_0 to 0-attractor(F)
- set W_1 to $V \setminus W_0$

σ-attractor(X): fix-point of X union

- for player σ: positions in V_σ that can reach X
- for player $1 - \sigma$: positions in $V_{1-\sigma}$ that can only reach X

Solving Reachability Games



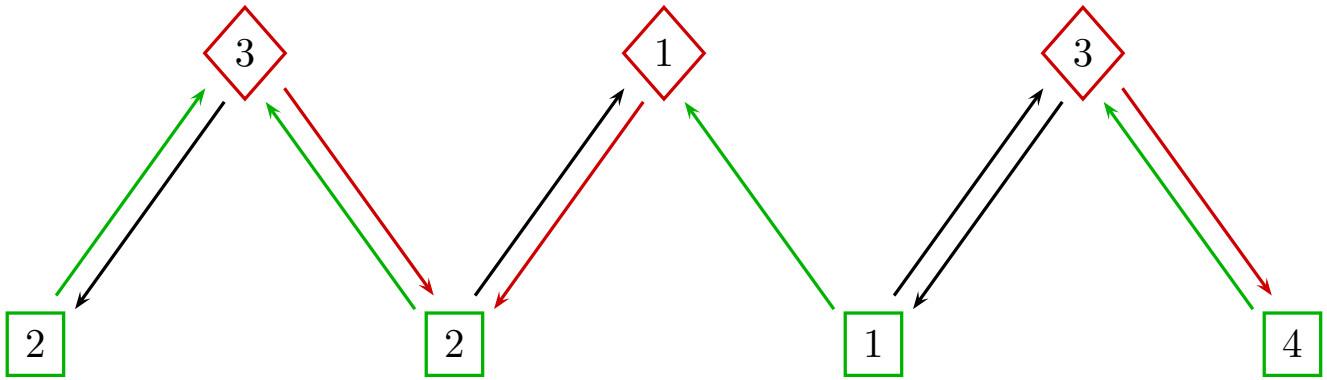
Algorithm – for $\mathcal{R} = \langle V_0, V_1, E, F \rangle$

- start with the final states F
- set W_0 to 0-attractor(F)
- set W_1 to $V \setminus W_0$

σ-attractor(X): fix-point of X union

- for player σ: positions in V_σ that can reach X
- for player $1 - \sigma$: positions in $V_{1-\sigma}$ that can only reach X

Parity Games



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
- $\alpha \colon (V_0 \cup V_1) \to \mathbb{N}$ is a priority function

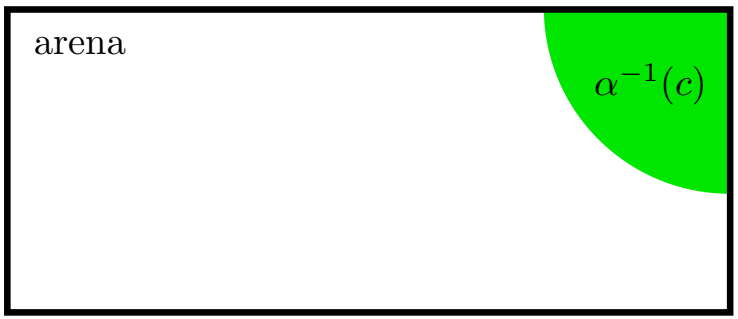
Played by placing a pebble on the arena
– on V_0 player 0 chooses a successor, on V_1 player 1
$\Rightarrow$ infinite play, lowest priority occurring infinite often
even $\rightsquigarrow$ player 0 wins, odd $\rightsquigarrow$ player 1 wins

State of the Art

# priorities	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]	$O(n^{2.5})$	$O(n^3)$	$O(n^{3\frac{1}{3}})$	$O(n^{3\frac{3}{4}})$	$O(n^{4\frac{1}{16}})$	$O(n^{4\frac{9}{20}})$

- but if you don't tell the games that they are hard McNaughton is by far the fastest

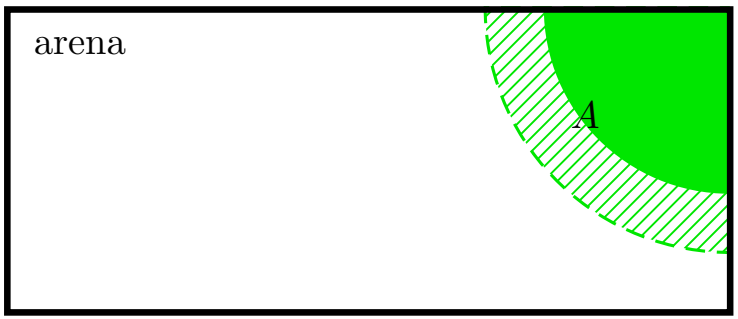
McNaughton's Algorithm



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

- set c to the minimal priority, 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 $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

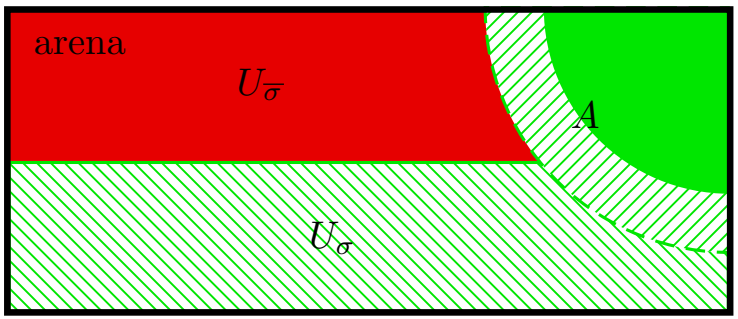
McNaughton's Algorithm



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

- set c to the minimal priority, 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 $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

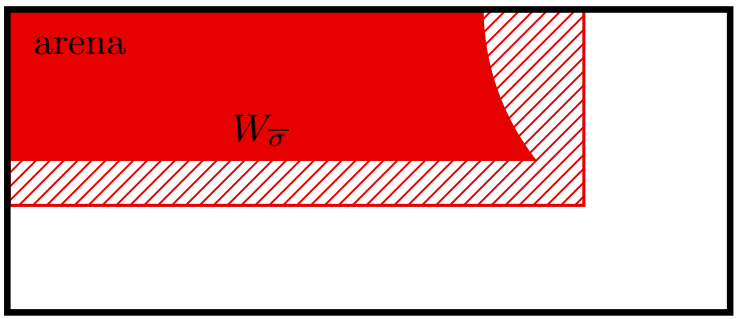
McNaughton's Algorithm



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

- set c to the minimal priority, 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 $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

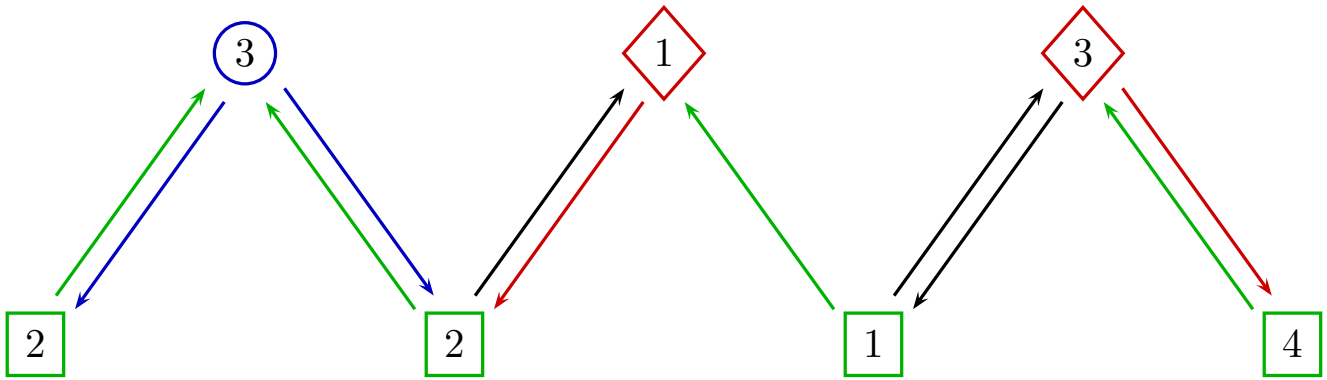
McNaughton's Algorithm



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

- set c to the minimal priority, 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 $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

Probabilistic Parity Games



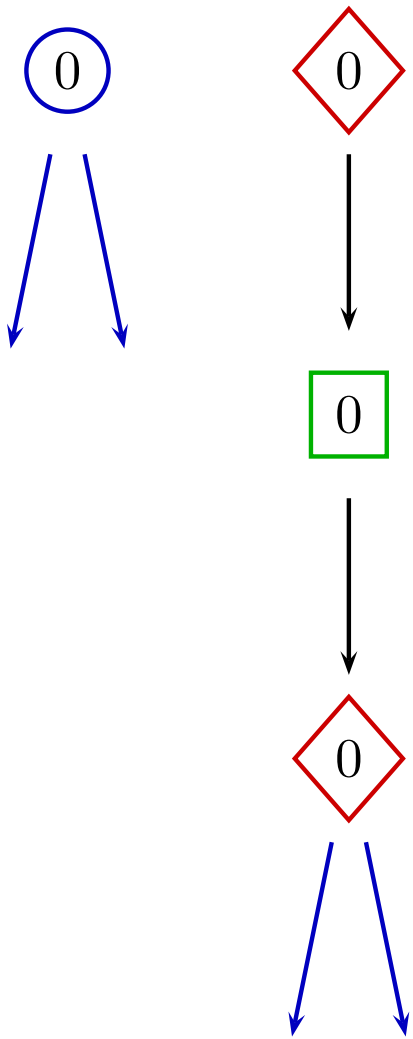
Probabilistic Parity Game $\mathcal{P} = \langle V_0, V_1, V_R, E, \alpha \rangle$

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

Played by placing a pebble on the arena
– on V_0 player 0 chooses a successor, on V_1 player 1
– on V_R a successor is chosen randomly
$\Rightarrow$ infinite play, lowest priority occurring infinite often
even $\rightsquigarrow$ player 0 wins, odd $\rightsquigarrow$ player 1 wins

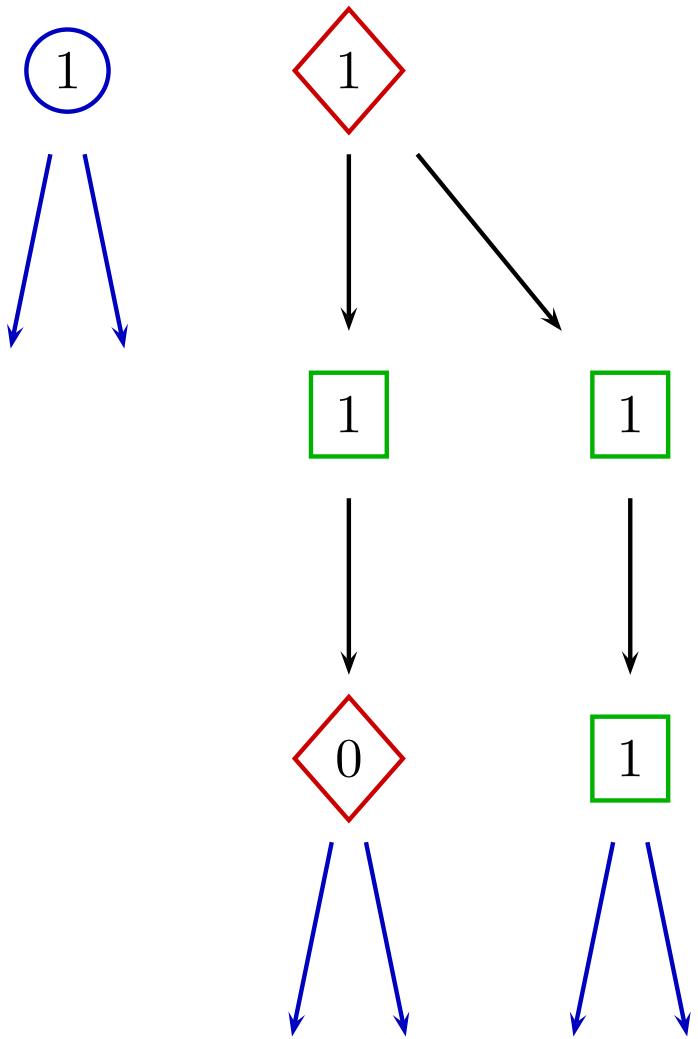
Removing the Random Player: Gadget Construction

random vertex with priority 0



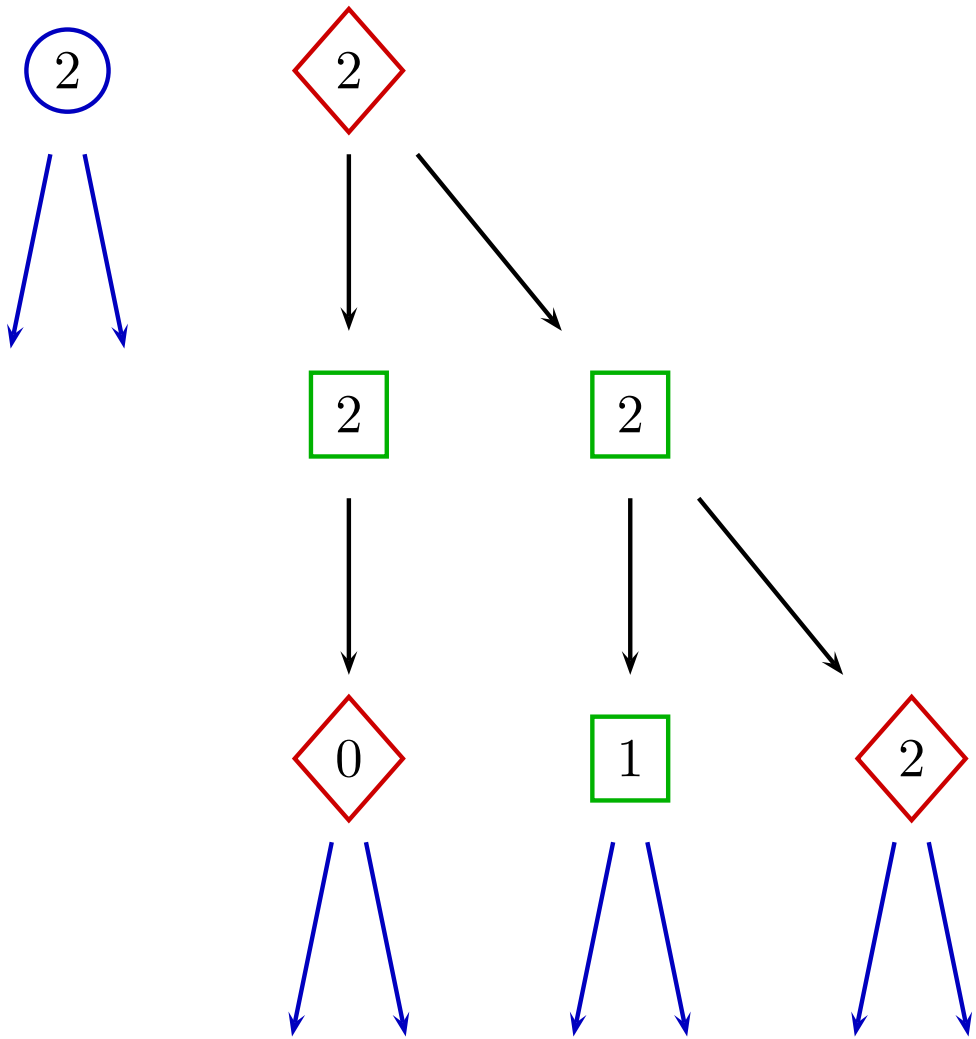
Removing the Random Player: Gadget Construction

random vertex with priority 1



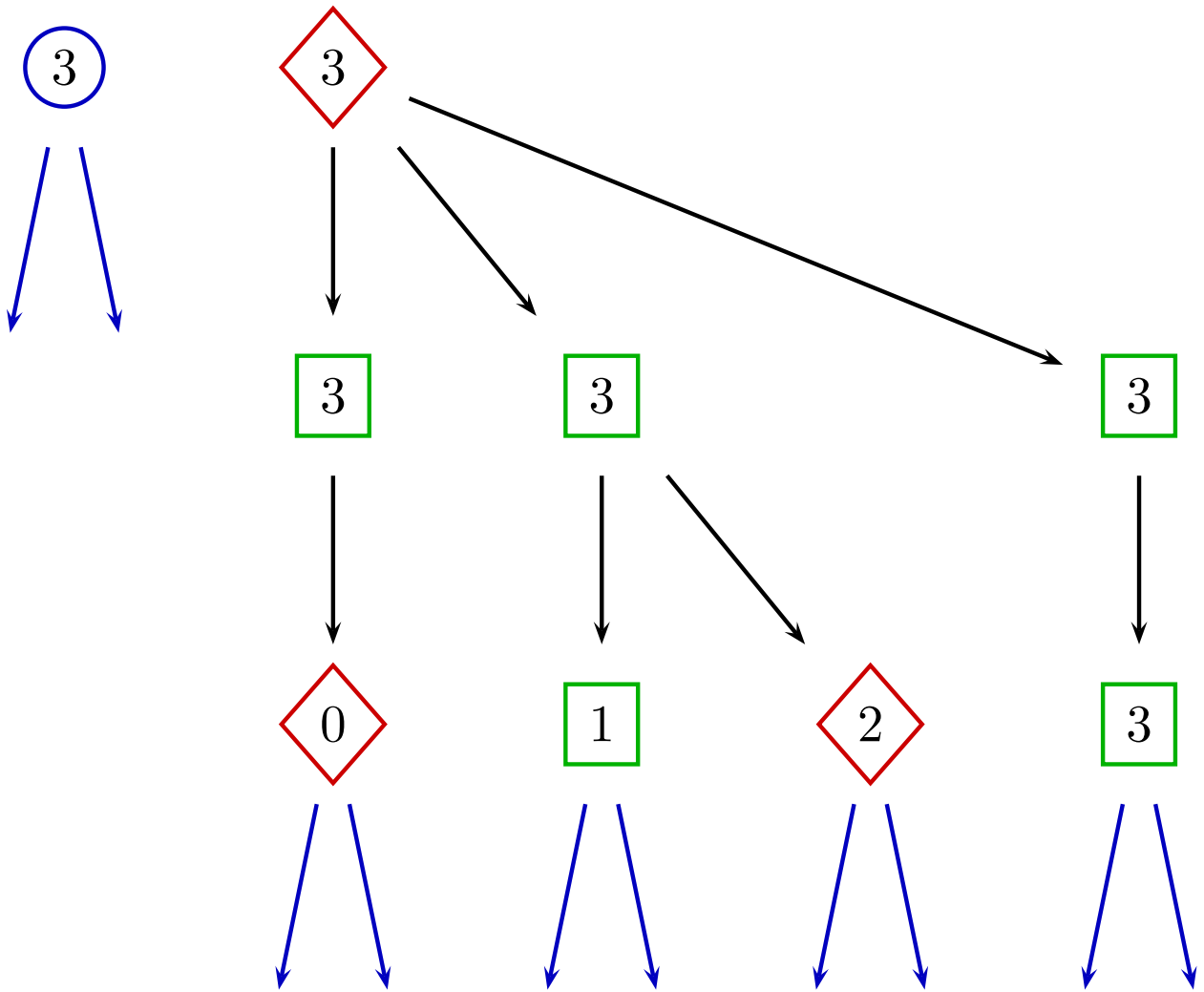
Removing the Random Player: Gadget Construction

random vertex with priority 2



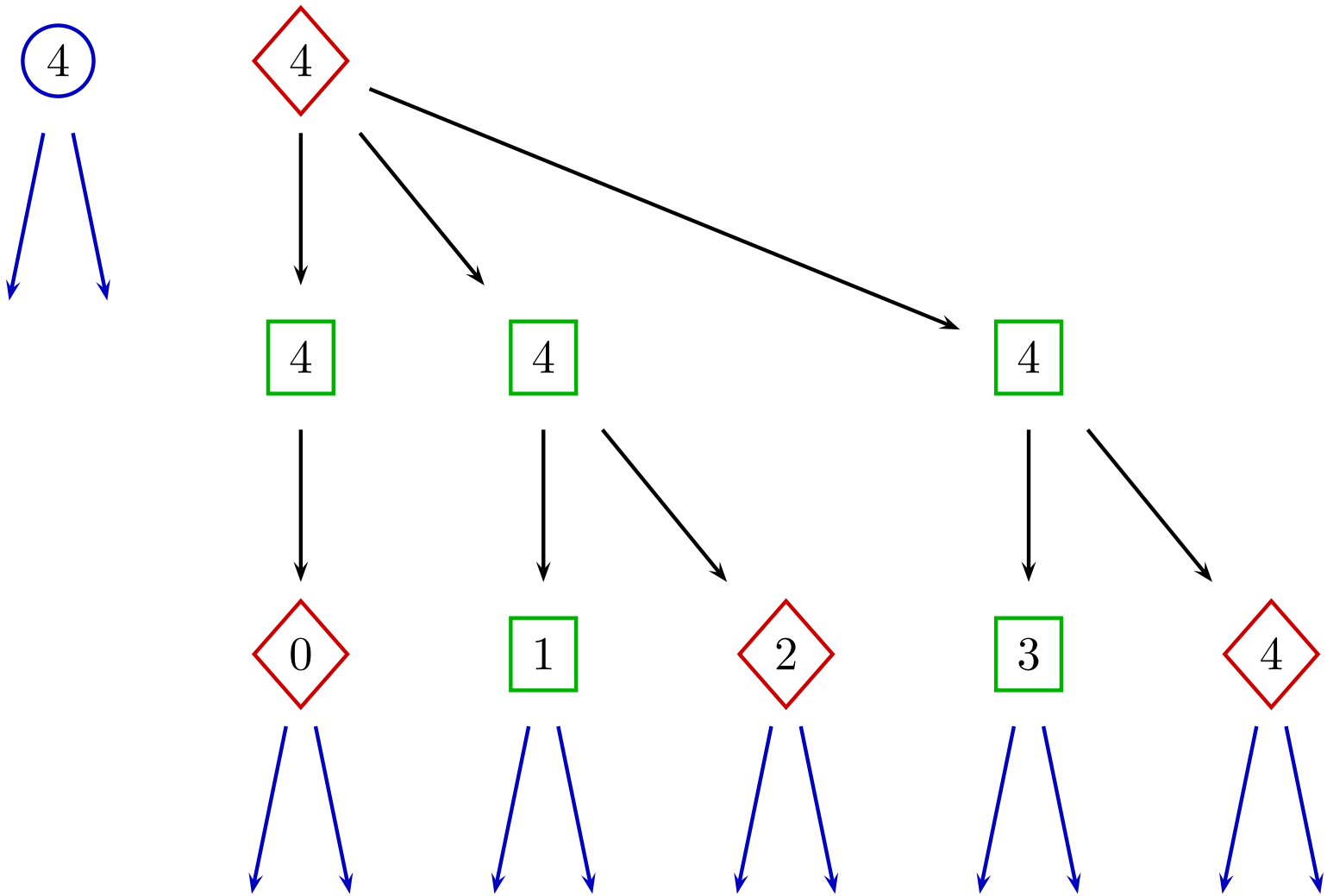
Removing the Random Player: Gadget Construction

random vertex with priority 3



Removing the Random Player: Gadget Construction

random vertex with priority 4



Different Attractors

Weak attractors

- reach goal almost surely

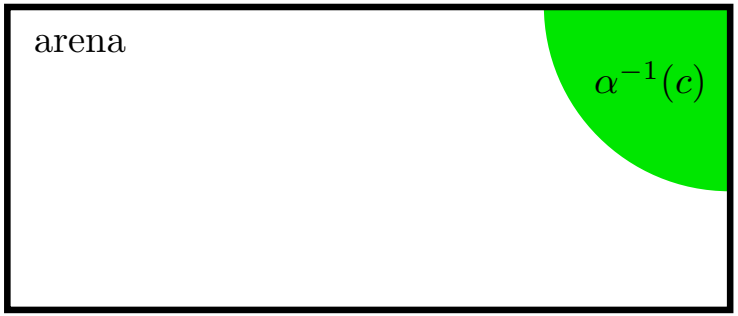
Strong attractors

- reach goal with positive probability

$\Rightarrow$ treat probabilistic nodes as "yours"

Note: weak attractor(G) $\subseteq$ strong attractor(G)

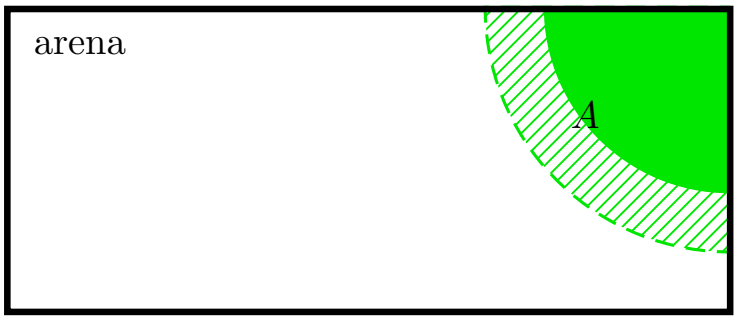
Probabilistic McNaughton



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

- set c to the minimal priority and σ to c modulo 2
- treat random nodes as player σ vertices
- set A to σ-attractor$(\alpha^{-1}(c))$
- set (U_0, U_1) to Prob-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 Prob-McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

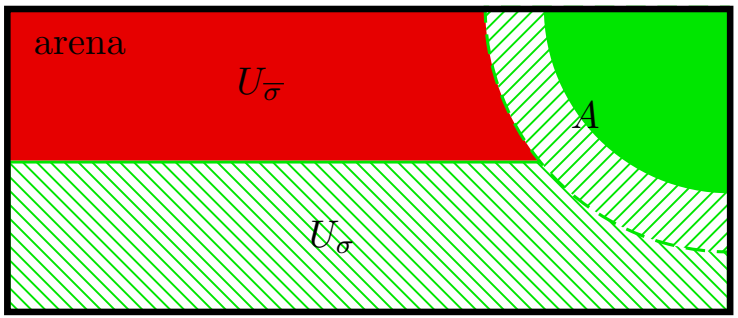
Probabilistic McNaughton



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

- set c to the minimal priority and σ to c modulo 2
- treat random nodes as player σ vertices
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to Prob-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 Prob-McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

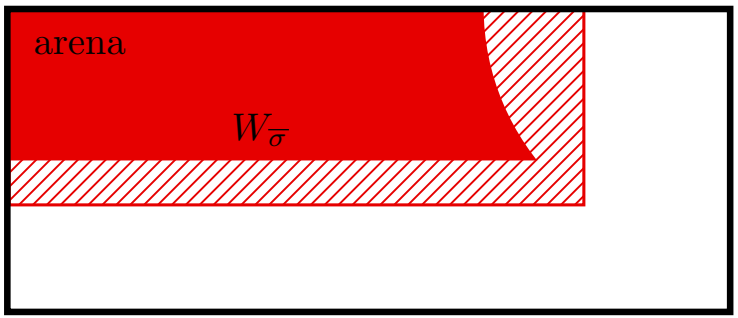
Probabilistic McNaughton



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

- set c to the minimal priority and σ to c modulo 2
- treat random nodes as player σ vertices
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to Prob-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 Prob-McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

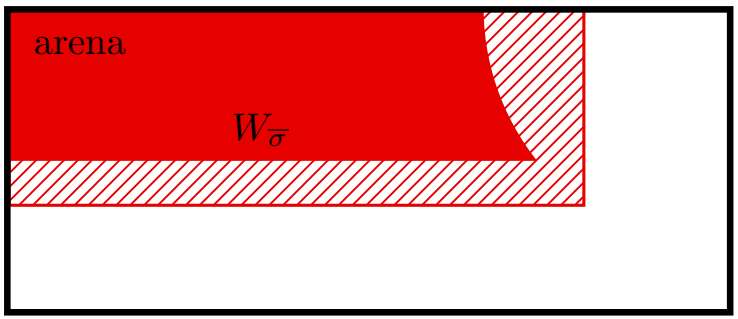
Probabilistic McNaughton – $\sigma = 0, \overline{\sigma} = 1$



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

- set c to the minimal priority and σ to c modulo 2
- treat random nodes as player σ vertices
- set A to σ-attractor$\left(\alpha^{-1}(c)\right)$
- set (U_0, U_1) to Prob-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 Prob-McNaughton$(\mathcal{P} \setminus W_{\overline{\sigma}})$
- return $(W_0 \dot{\cup} U_0, W_1 \dot{\cup} U_1)$

Probabilistic McNaughton – $\sigma = 1, \overline{\sigma} = 0$



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

- use weak attractor
  (instead of the strong attractor)
- co-game of the weak attractor might have sub-stochastic vertices
- they are turned into "good" vertices, by
  - giving them priority 0 or
  - adding an accepting sink as successor

$\Rightarrow$ visiting them infinitely often wins

Experimental Results

property	n	sat	game construction vertices	game construction priorities	t_{prod}	gadget construction vertices	t_{gadg}	solving time t_{gMcN}	solving time t_{gJur}	solving time t_{pMcN}
Reachability	12	true	1 324 704	2	2	5 351 584	3	0	4	0
	16	true	4 418 592	2	9	17 996 832	13	3	24	1
$\langle\langle R_0 \rangle\rangle \mathcal{P}_{\geq 1}$	20	true	11 050 656	2	22	45 166 752	28	8	84	3
[$\mathbf{F}\,\mathrm{zone}_1$	24	true	23 211 552	2	51	95 045 152	58	18	219	7
$\wedge\ \mathbf{F}\,\mathrm{zone}_2$]	28	true	43 334 304	2	102	177 634 464	115	35	–MO–	14
	32	true	74 294 304	2	197	-	–MO–	-	-	24
	36	-	-	-	–MO–	-	-	-	-	-
	40	-	-	-	–MO–	-	-	-	-	-
Repeated	12	true	1 324 704	2	2	6 010 528	3	1	5	0
Reachability	16	true	4 418 592	2	9	20 085 792	15	3	25	1
	20	true	11 050 656	2	20	50 273 952	29	9	85	3
$\langle\langle R_0 \rangle\rangle \mathcal{P}_{\geq 1}$	24	true	23 211 552	2	44	105 643 552	59	21	227	8
[$\mathbf{GF}\,\mathrm{zone}_1$	28	true	43 334 304	2	88	197 278 368	121	38	–MO–	15
$\wedge\ \mathbf{GF}\,\mathrm{zone}_2$	32	true	74 294 304	2	180	-	–MO–	-	-	27
$\wedge\ \mathbf{GF}\,\mathrm{zone}_3$	36	-	-	-	–MO–	-	-	-	-	-
$\wedge\ \mathbf{GF}\,\mathrm{zone}_4$]	40	-	-	-	–MO–	-	-	-	-	-
Repeated	12	true	693 048	5	0	4 009 976	2	0	3	0
Ordered	16	true	2 264 184	5	3	13 206 072	9	2	16	0
Reachability	20	true	5 611 320	5	6	32 844 792	17	6	50	1
	24	true	11 729 784	5	14	68 787 512	62	13	129	4
$\langle\langle R_0 \rangle\rangle \mathcal{P}_{\geq 1}$	28	true	21 836 088	5	27	128 198 136	69	24	282	7
[$\mathbf{GF}(\mathrm{zone}_1 \wedge$	32	true	37 367 928	5	54	219 543 096	135	–MO–	–MO–	13
$\mathbf{F}\,\mathrm{zone}_2)$]	36	true	59 984 184	5	65	-	–MO–	-	-	21
	40	true	91 564 920	5	121	-	–MO–	-	-	36
Reach-avoid	12	true	1 616 400	4	2	7 656 720	4	1	–TO–	0
	16	true	5 452 848	4	14	25 820 208	15	6	–TO–	1
$\langle\langle R_0 \rangle\rangle \mathcal{P}_{\geq 1}$	20	true	13 703 184	4	35	64 877 328	33	19	–TO–	5
[$\neg\mathrm{zone}_1\ \mathbf{U}\ \mathrm{zone}_2$	24	true	28 855 728	4	79	136 606 128	116	40	–TO–	11
$\wedge\ \neg\mathrm{zone}_4\ \mathbf{U}\ \mathrm{zone}_2$	28	true	53 951 760	4	171	255 402 000	135	–MO–	–MO–	21
$\wedge\ \neg\mathrm{zone}_4\ \mathbf{U}\ \mathrm{zone}_1$	32	true	92 585 520	4	323	-	–MO–	-	-	38
$\wedge\ \mathbf{F}\mathrm{zone}_4$]	36	-	-	-	–MO–	-	-	-	-	-
	40	-	-	-	–MO–	-	-	-	-	-

Summary

- solving probabilistic parity games is as fast as solving parity games
- solving probabilistic parity games is as simple as solving parity games
- solving probabilistic parity games is as beautiful as solving parity games
- the same class of algorithms works best

Summary

Solving probabilistic parity games:

- fast
- simple
- beautiful

‘Comparison’ to GIST

vertices	edges	best	average	worst
1,000	5,000	0.63	1.17	1.59
10,000	50,000	39.38	51.43	62.61
50,000	250,000	2063.40	2513.18	2711.23