Game Tree Search on Gomoku

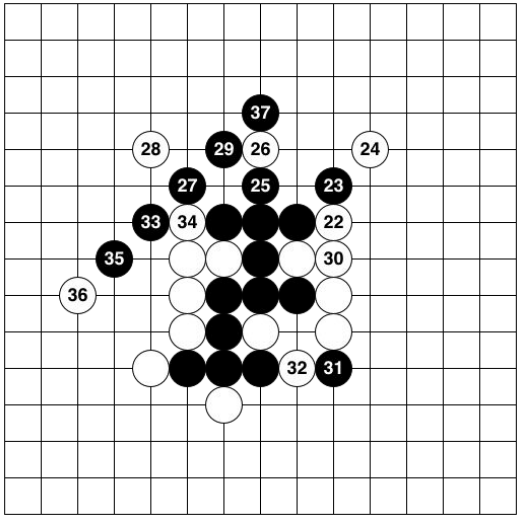
Group: 30
310554039 曾揚
0816153 陳琮方

Outline

- Introduction
- Tree Search & Parallelism
- Result
- Demo
- Conclution

Gomoku

Gomoku



- Five in a Row, 五子棋
- 19x19 Standard
- Player alteranate turn stone in their color
- Win with five in a row!

Goal

- Play with Computer
- Faster & Stronger

Tree Search & Parallelism

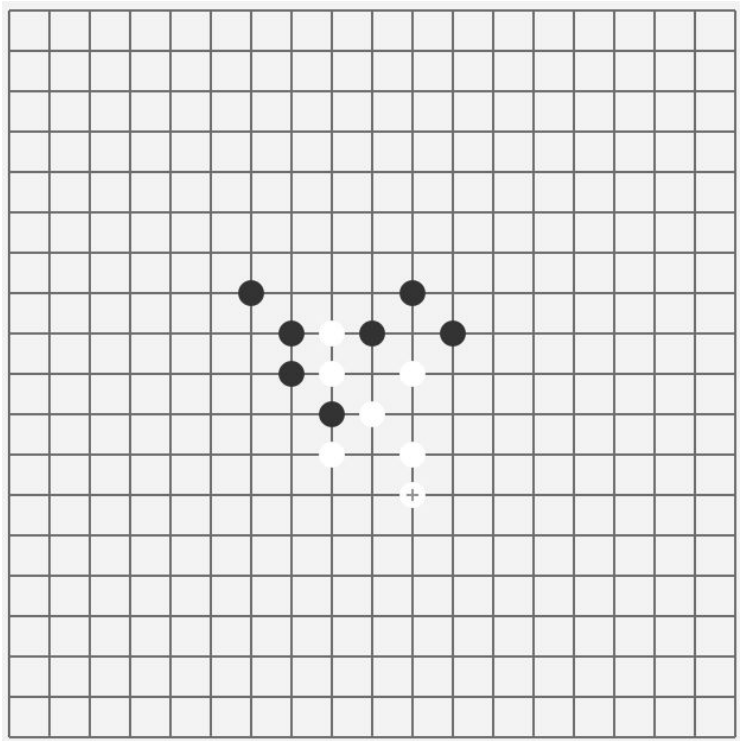
Approach

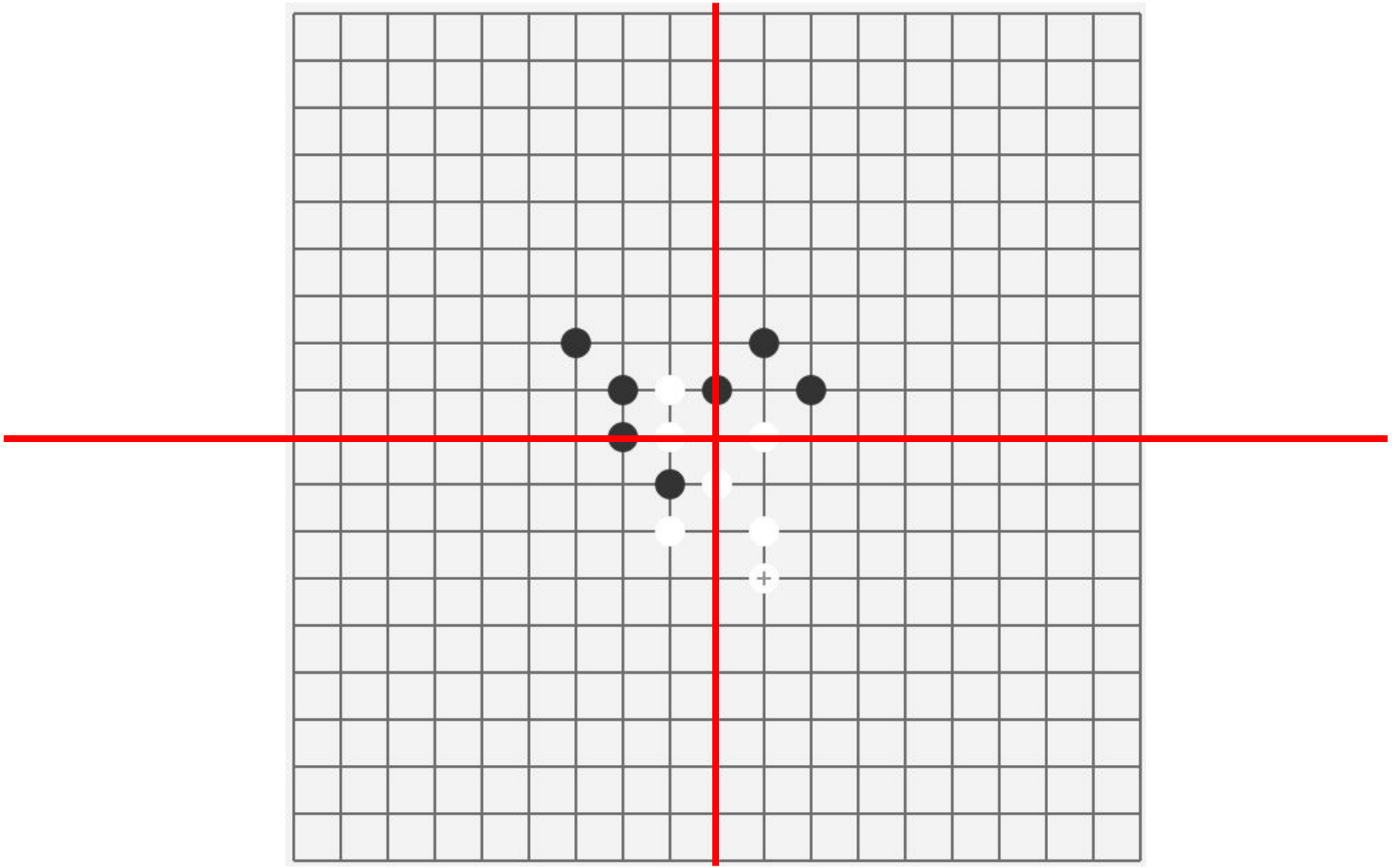
- Min-Max & Nega-Max
  - With Board Evaluation
- Monte Carlo Tree Search (MCTS)

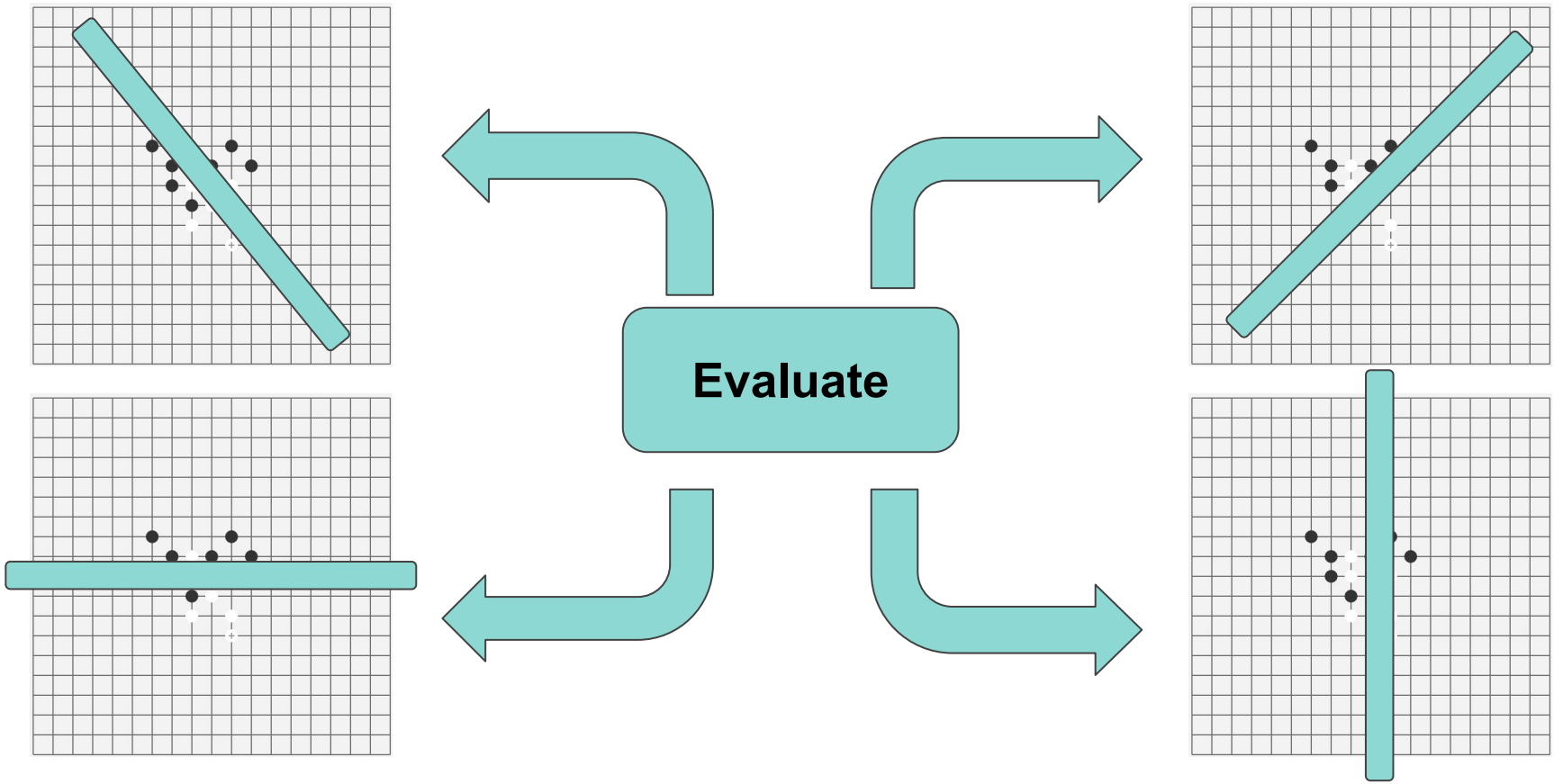
Board Evaluation

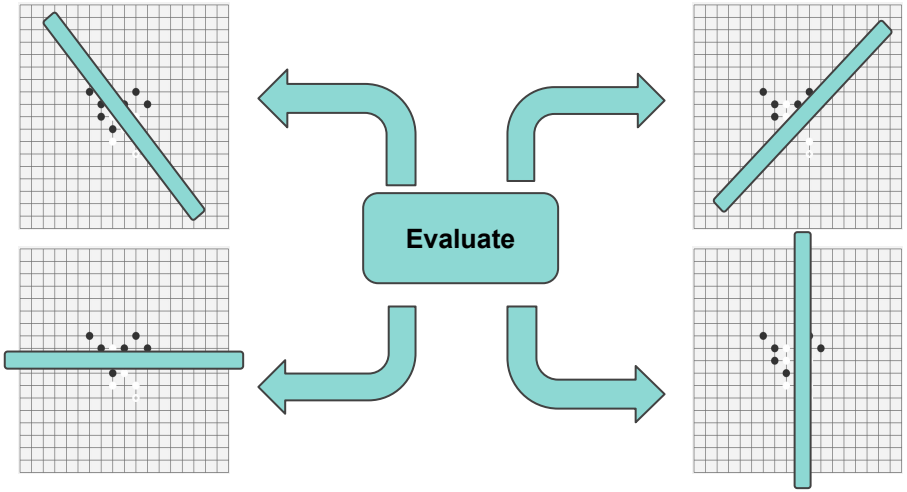
- 3 in a row
- 4 in a row
- Block/Non-Block
- Horizontal, Vertical, Diagonal

Speed Up Board Evaluation

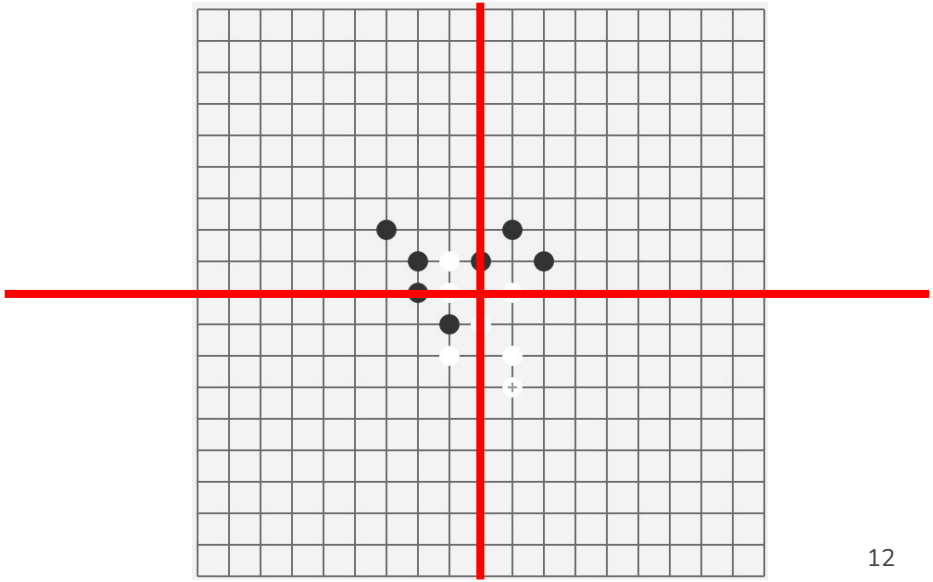





Evaluate

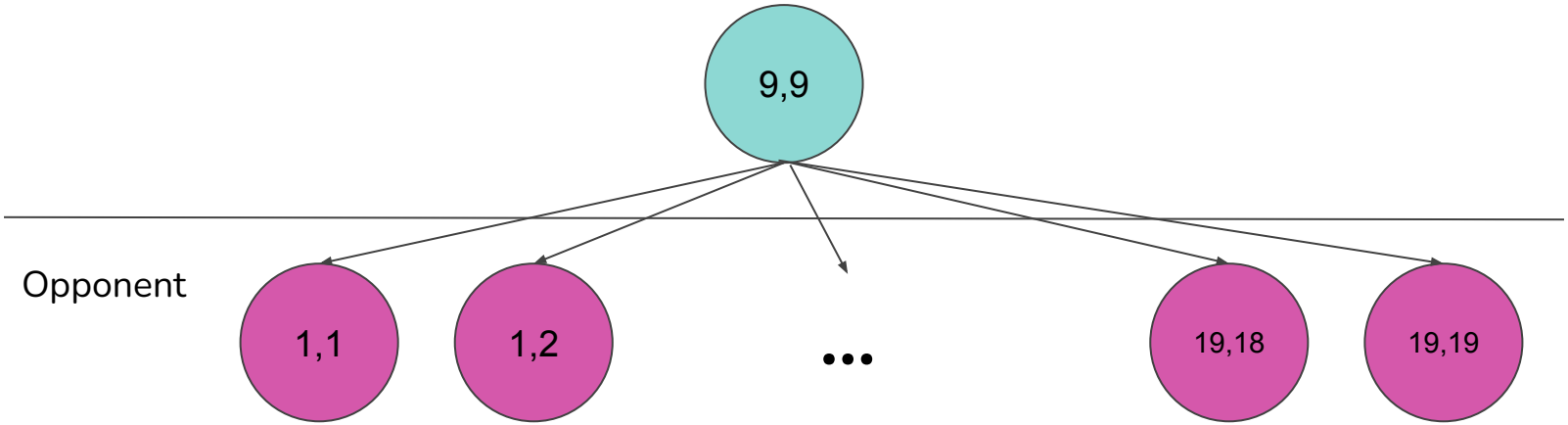


The Speed-Up is limited!

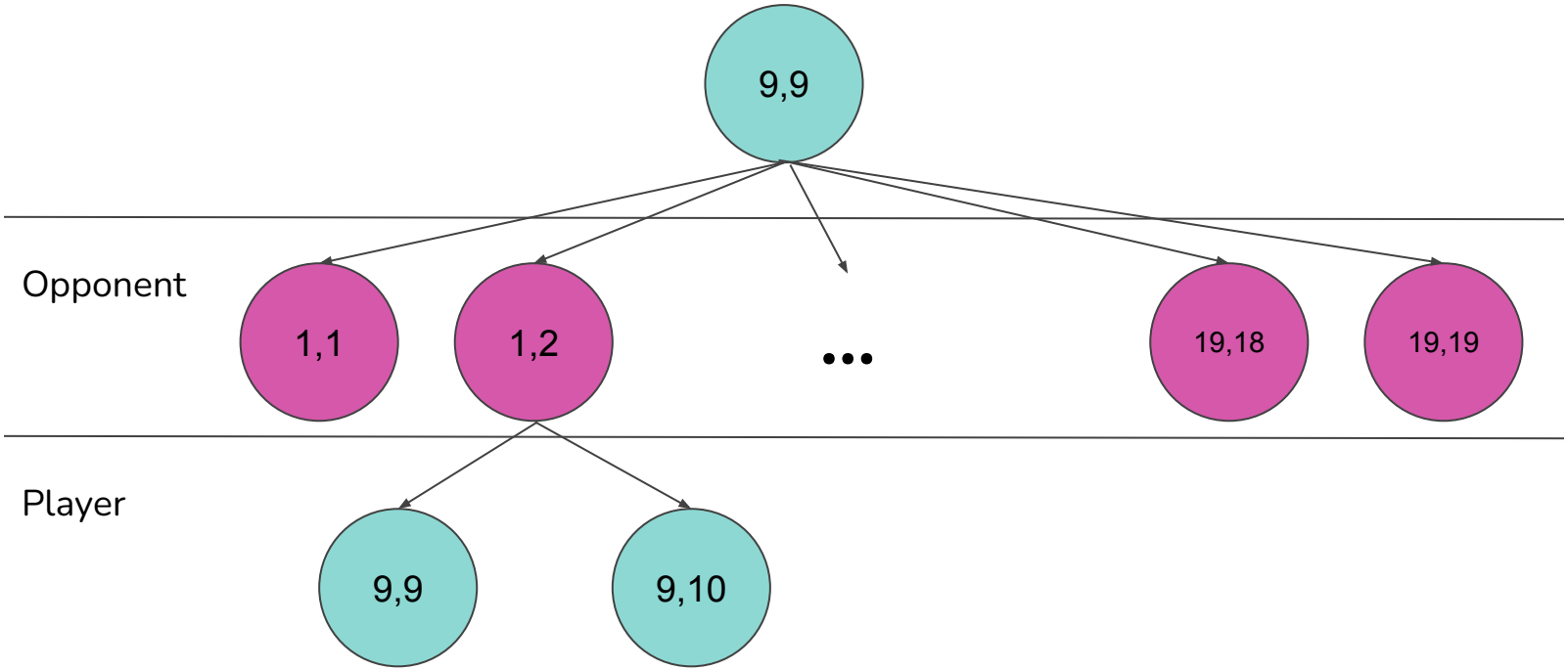


Min-Max Search

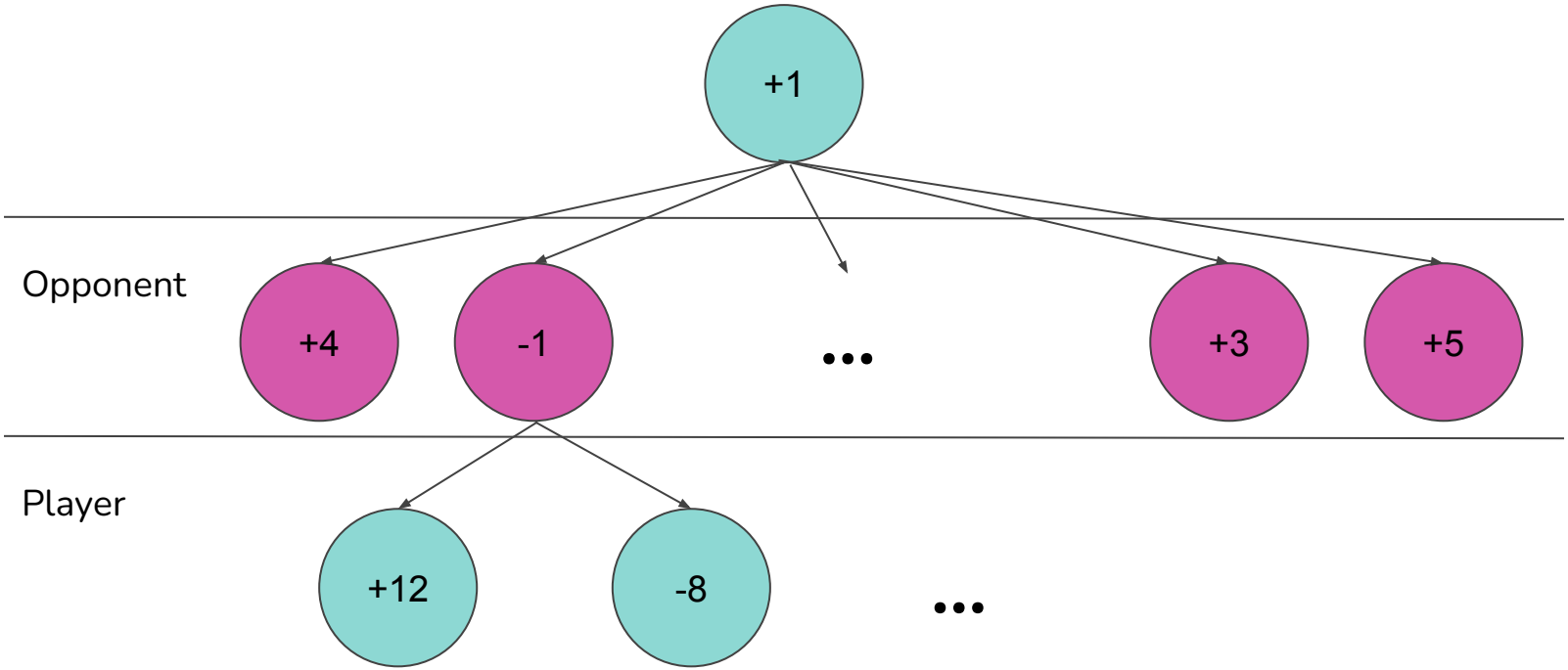
Min-Max Search



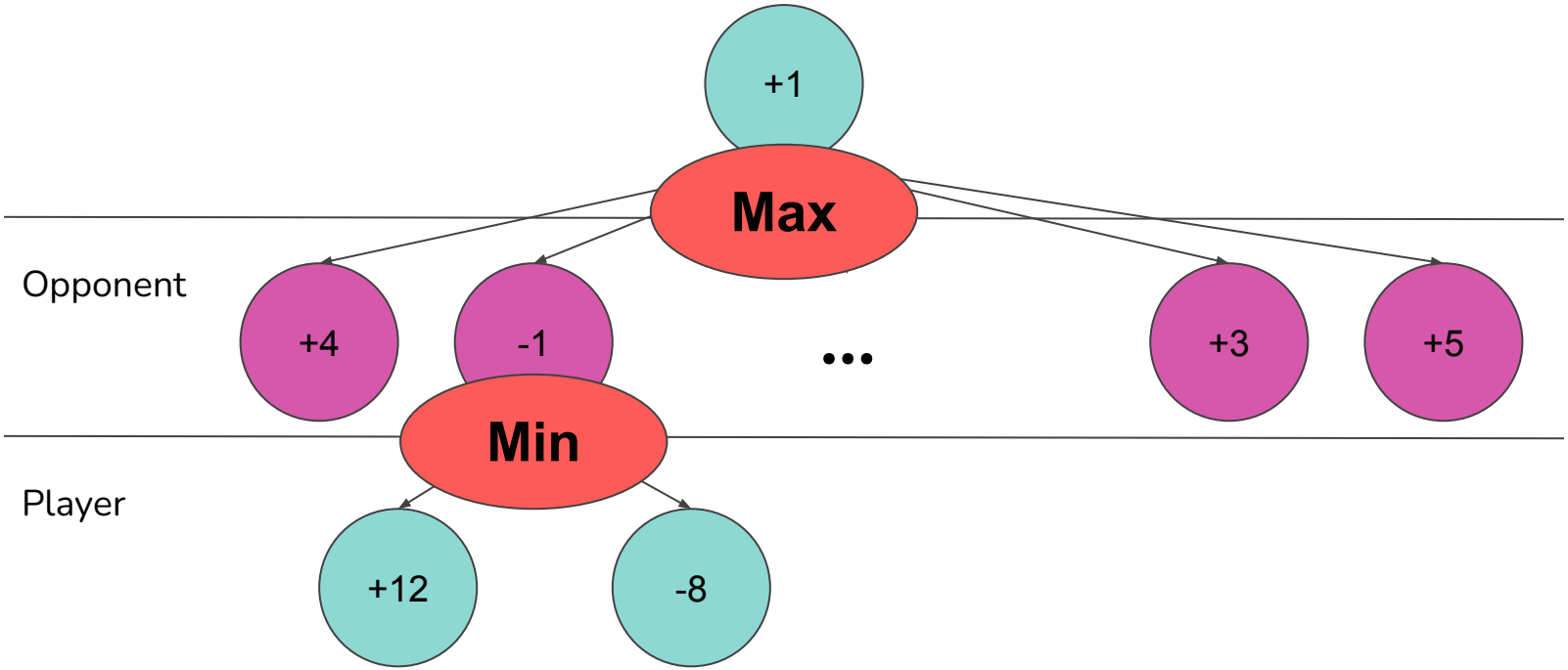
Min-Max Search



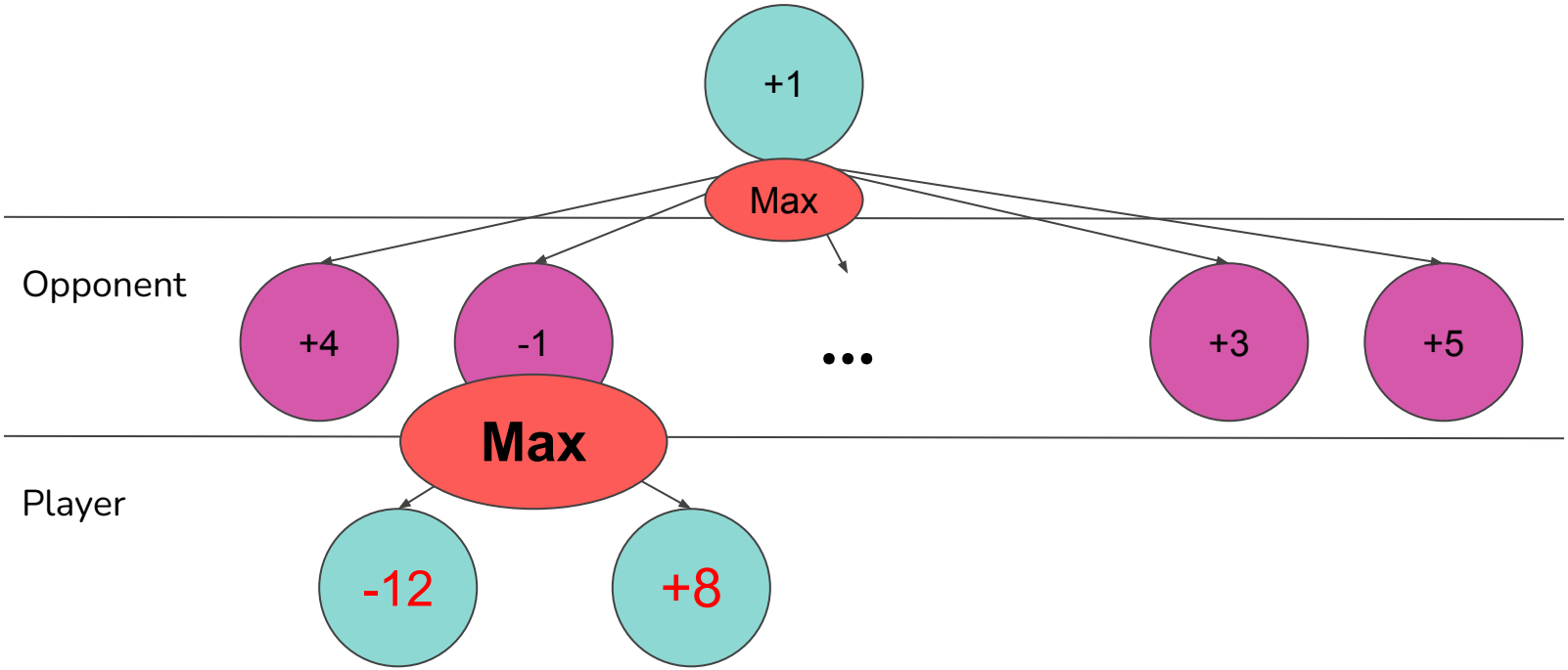
Min-Max Search



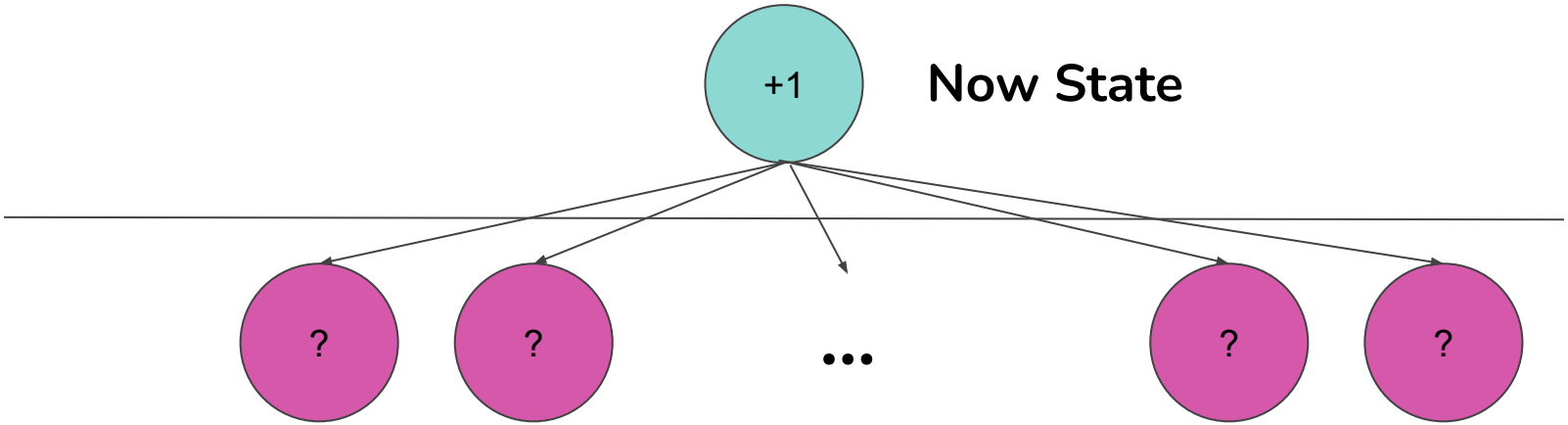
Min-Max Search

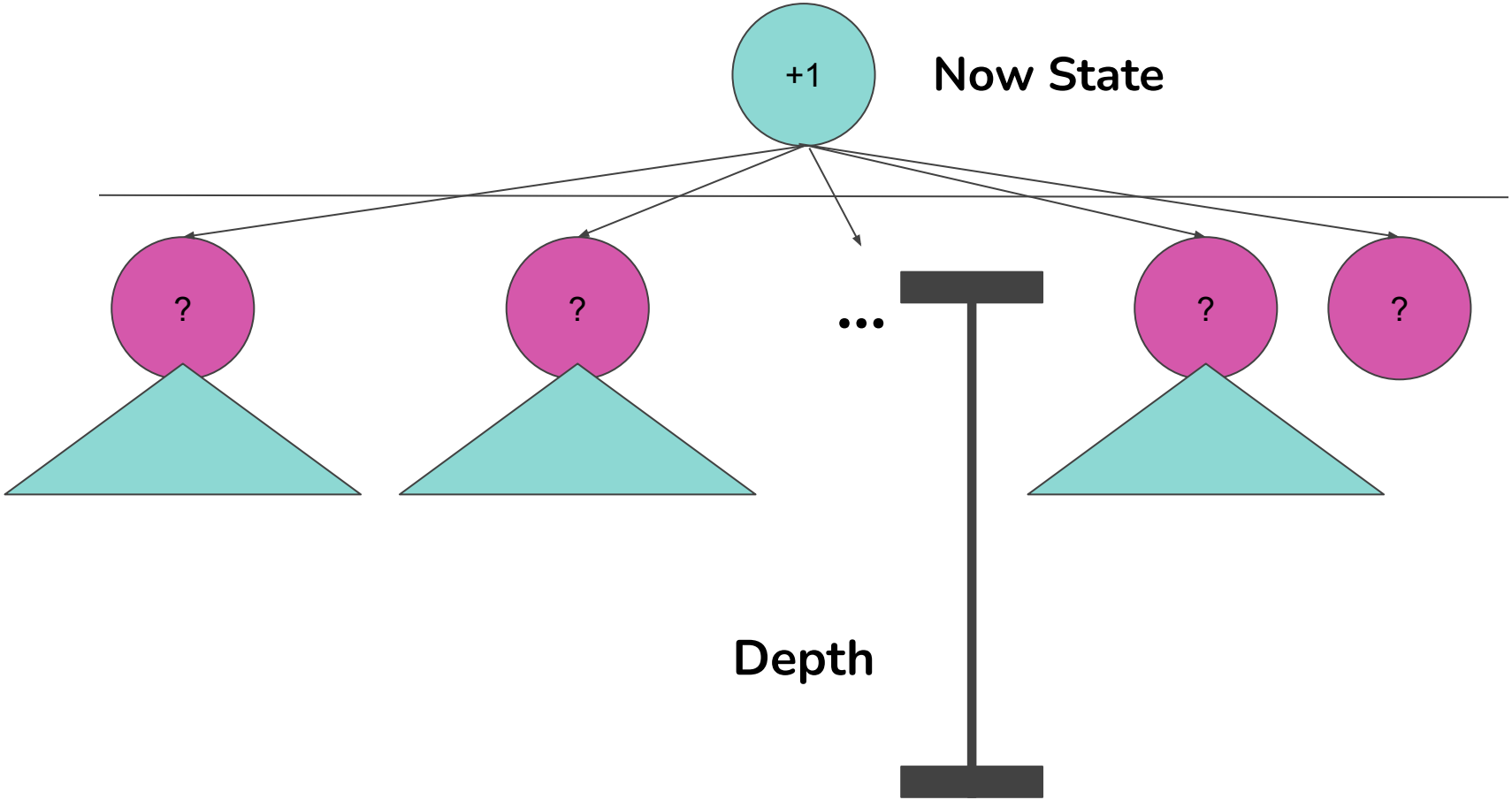


Nega-Max Search

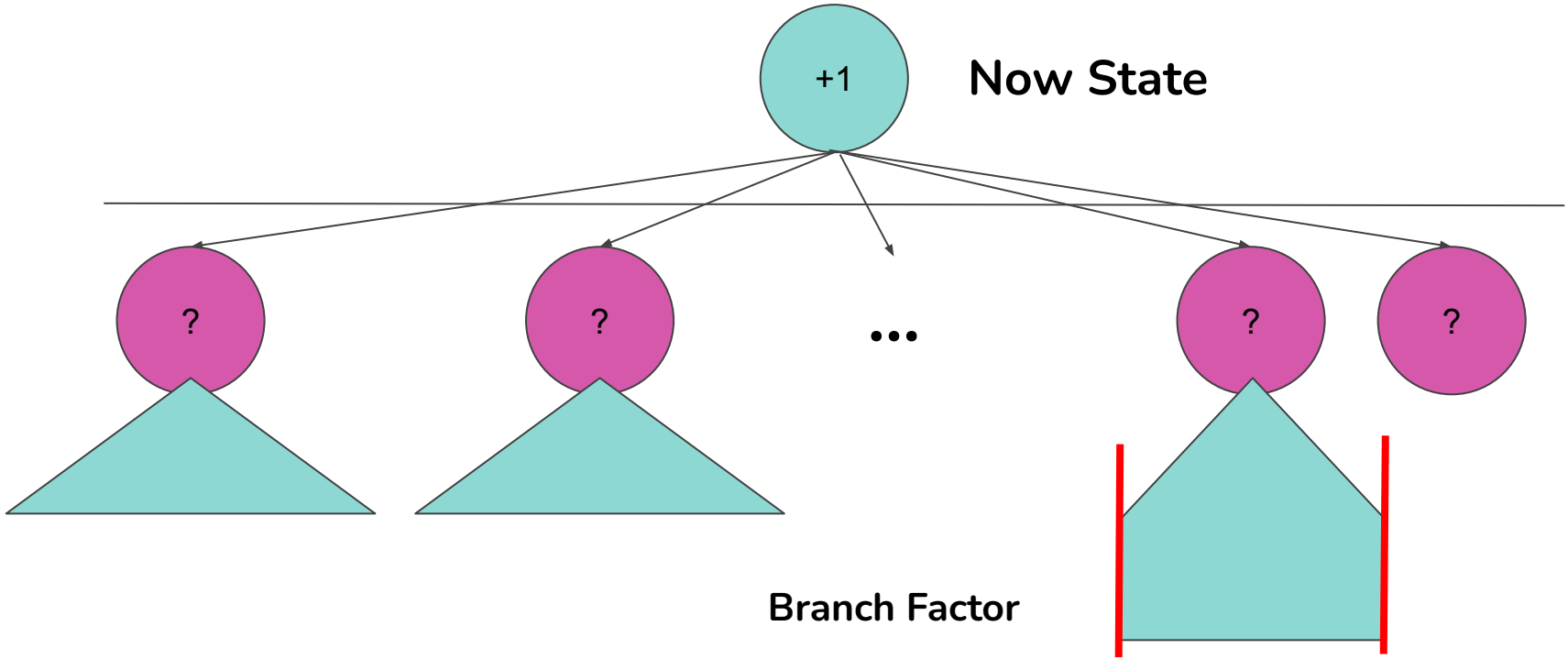


In Real

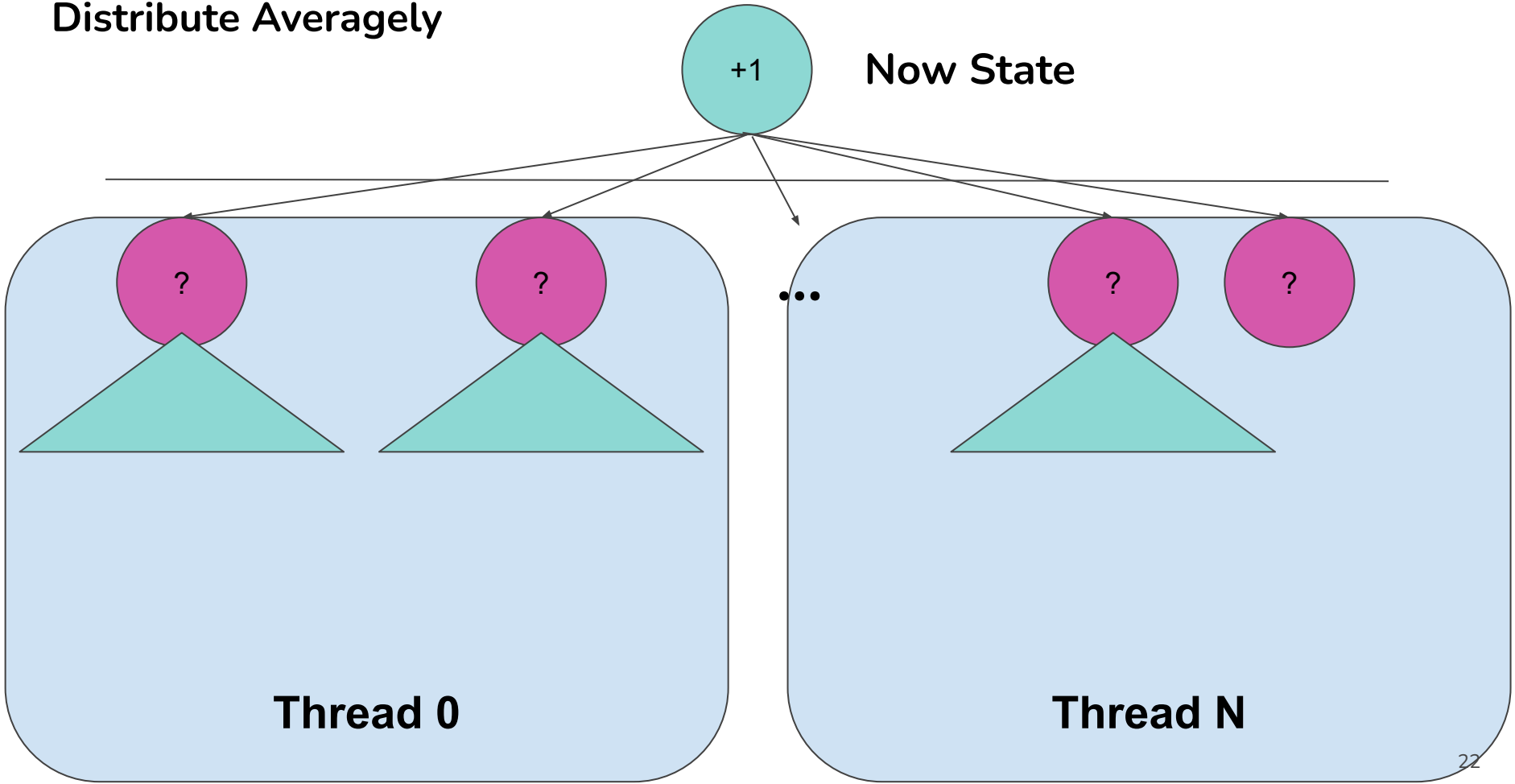



+1
Now State
?
?
...
?
?
Depth

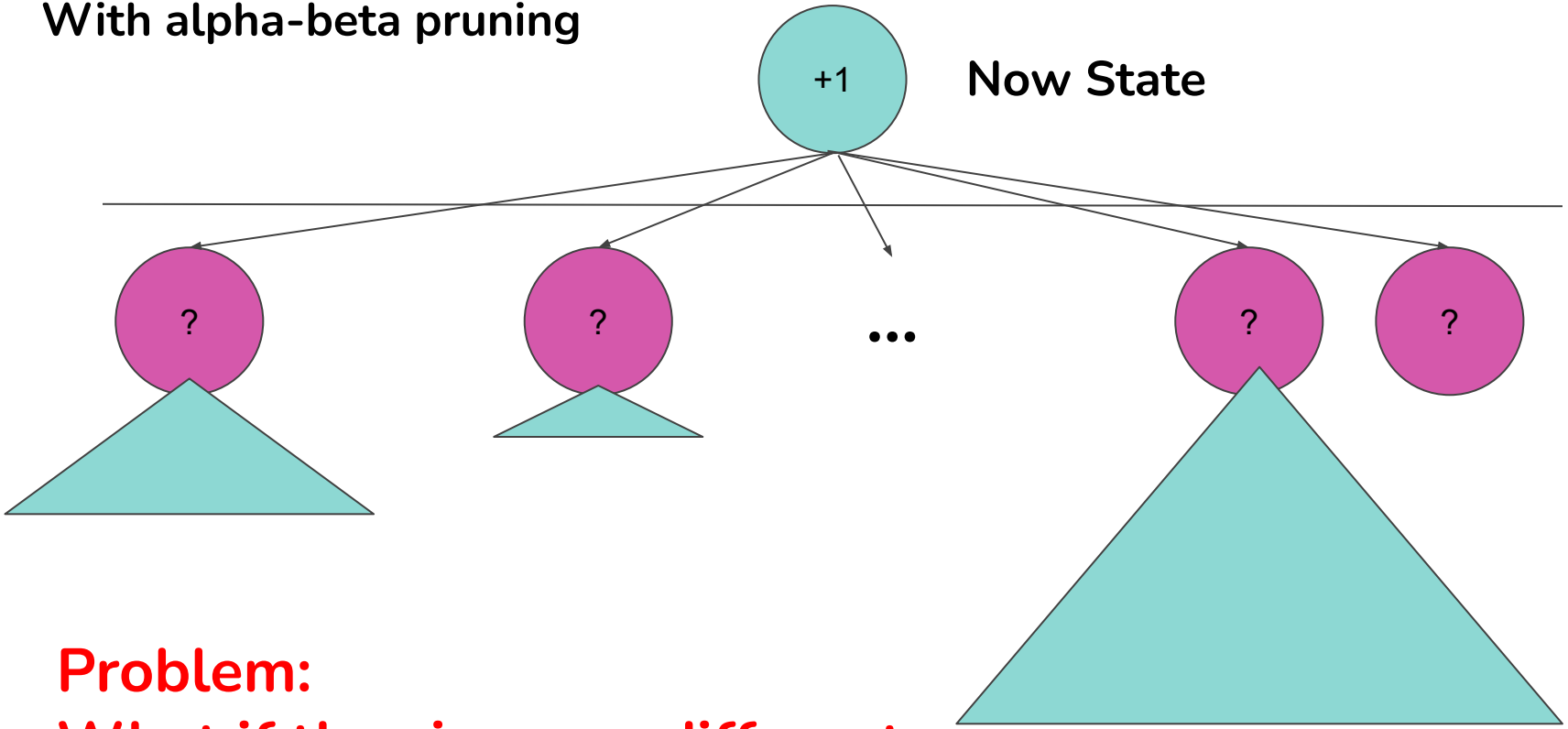
Speed-Up Min-Max



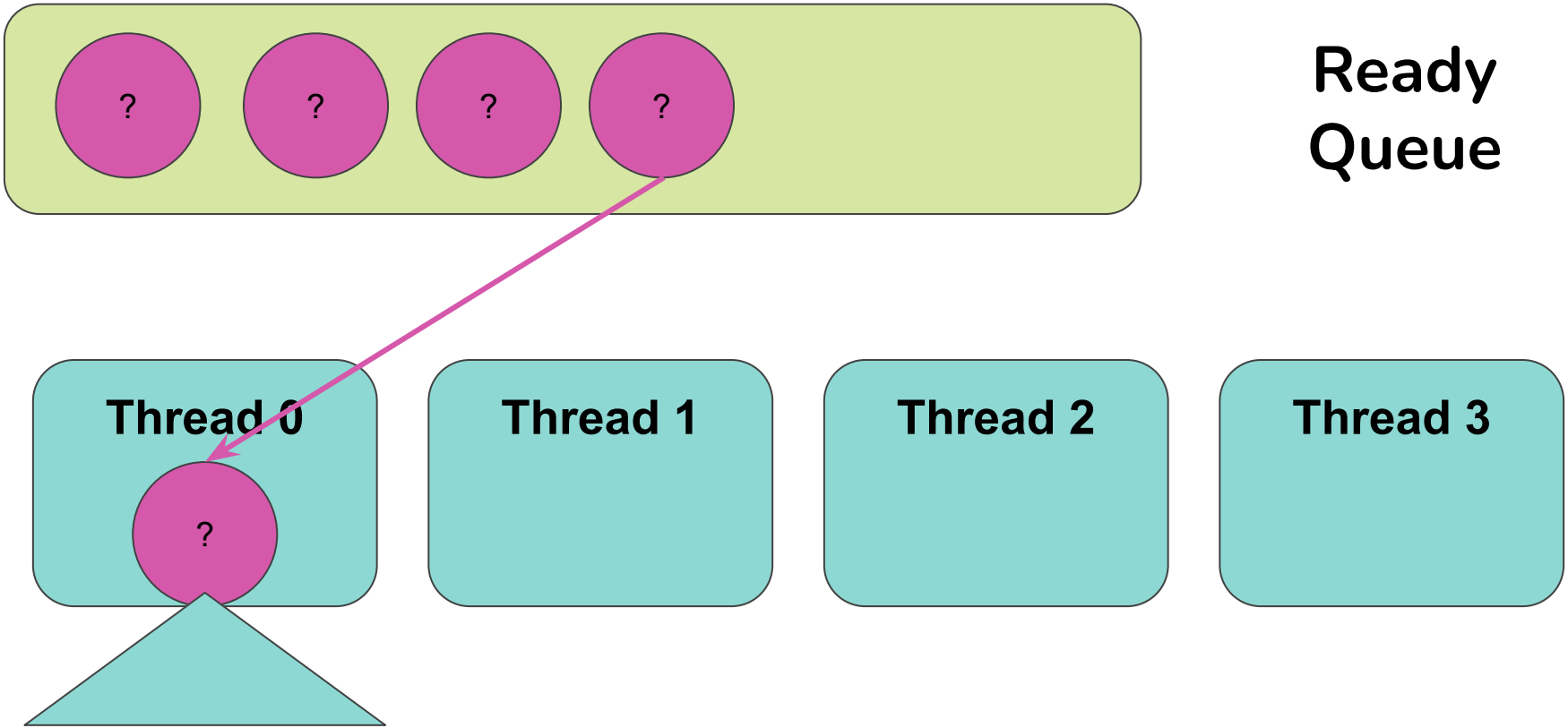
Distribute Averagely

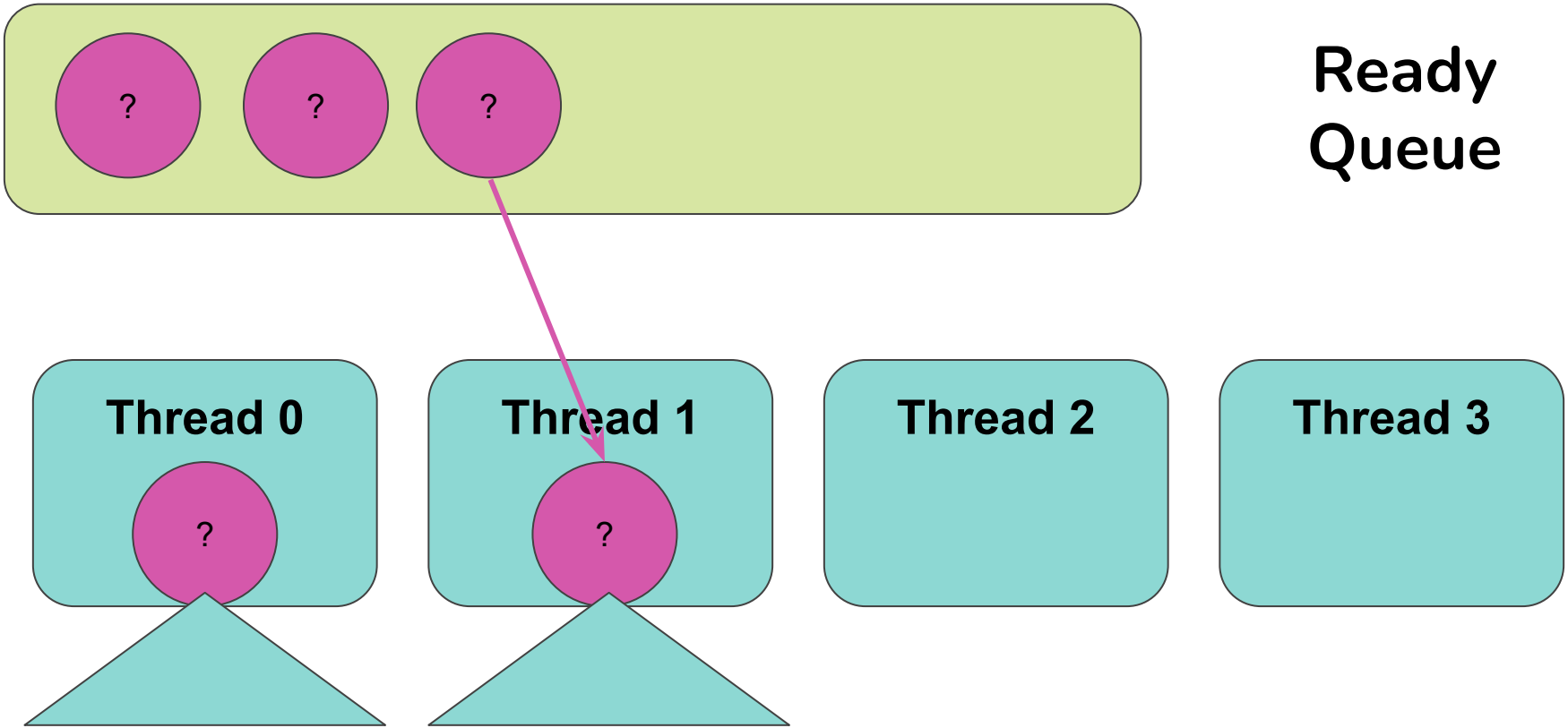


With alpha-beta pruning

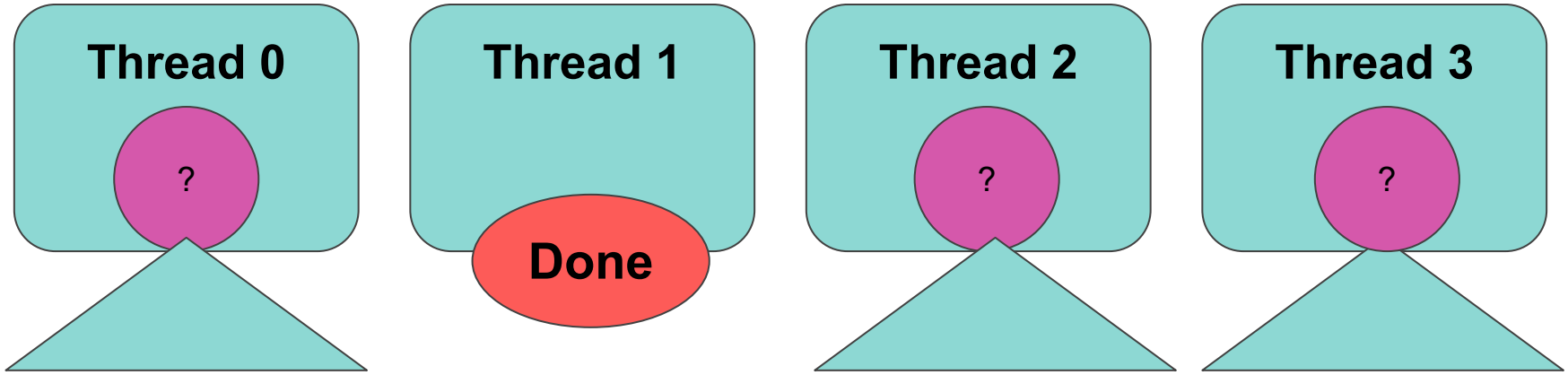


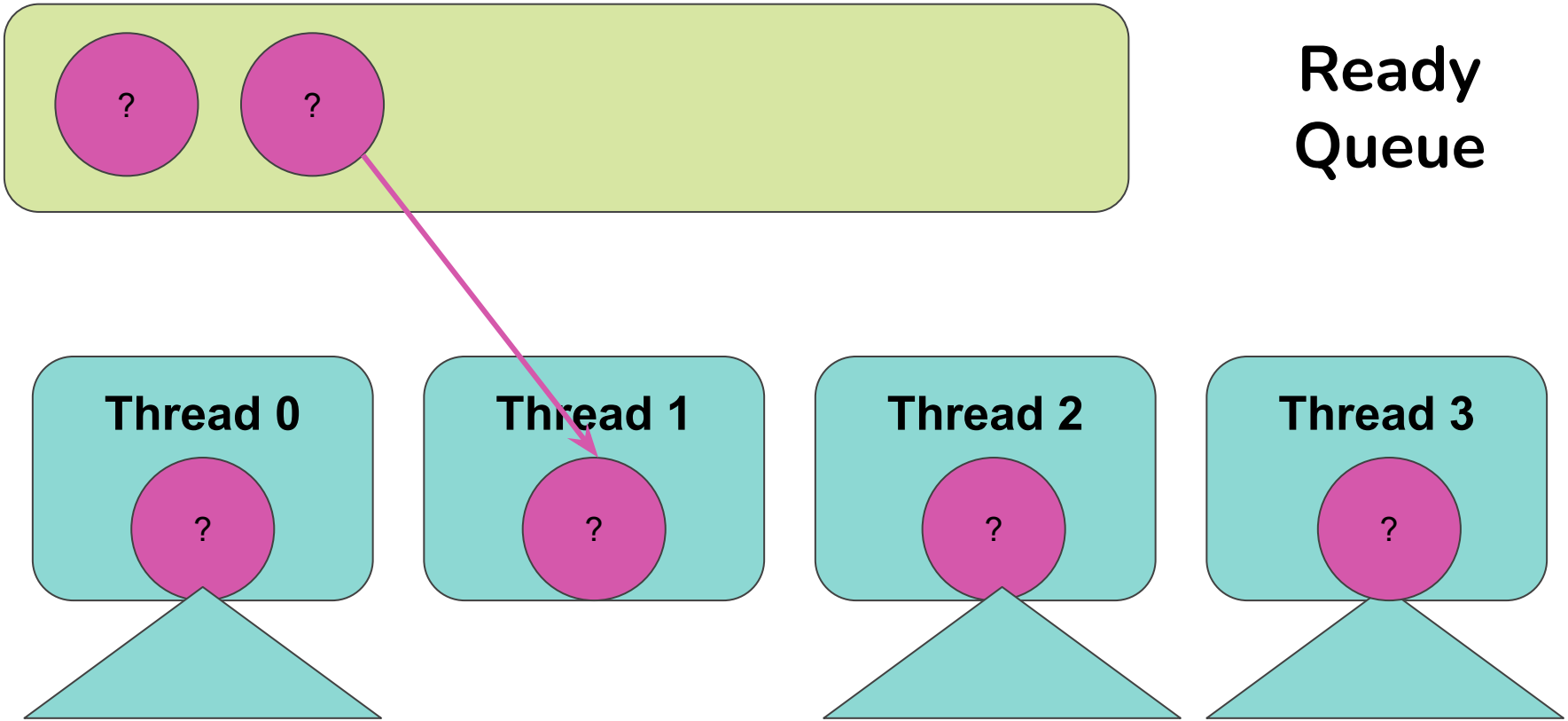
Problem:
What if the sizes are different


?
?
?
?
Ready Queue
Thread 0
Thread 1
Thread 2
Thread 3
?

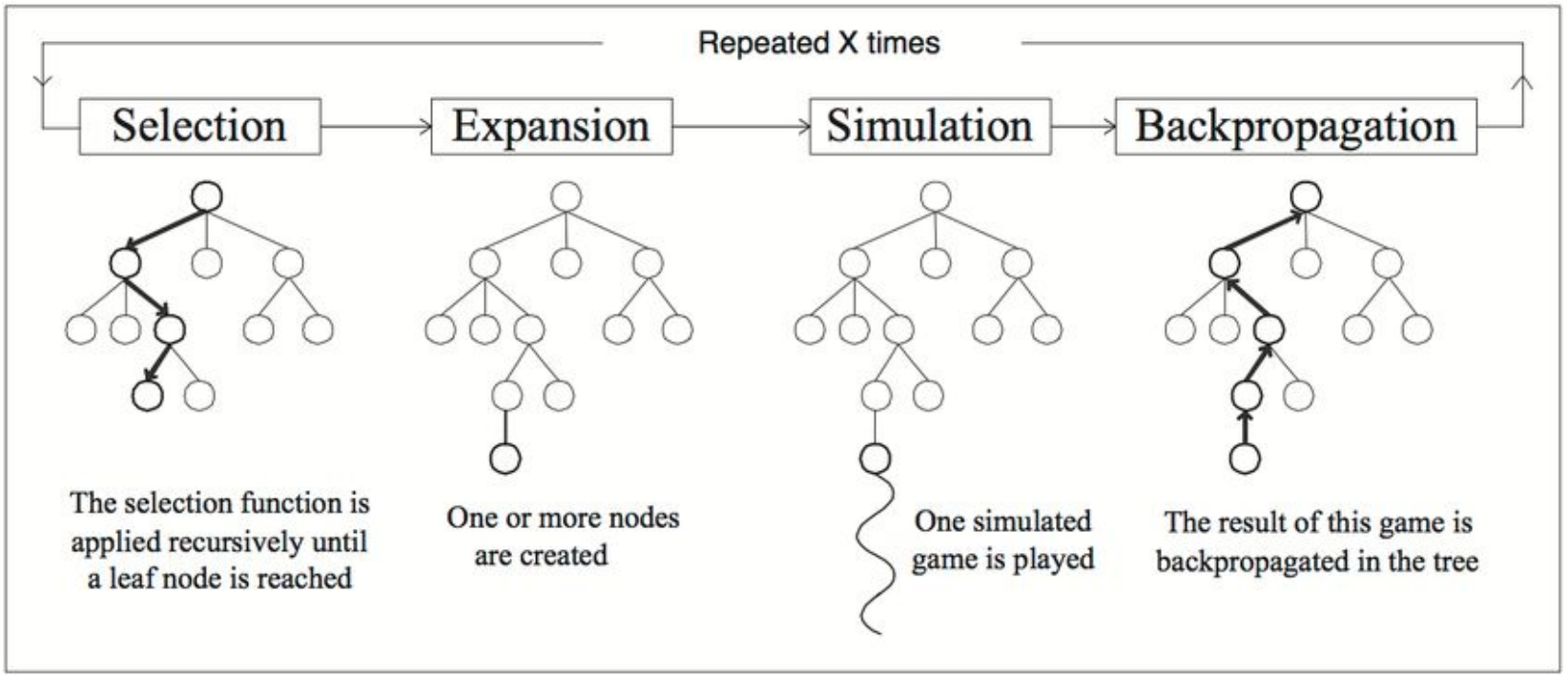

?
?
?
Ready Queue
Thread 0
Thread 1
Thread 2
Thread 3
?
?

Ready Queue

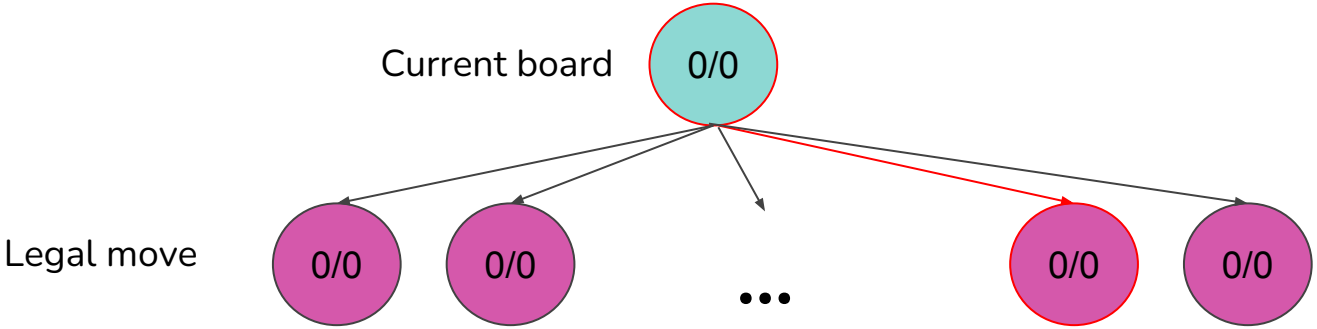



?
?
Ready Queue
Thread 0
?
Thread 1
?
Thread 2
?
Thread 3
?

Monte-Carlo Tree Search (MCTS)



MCTS - Selection

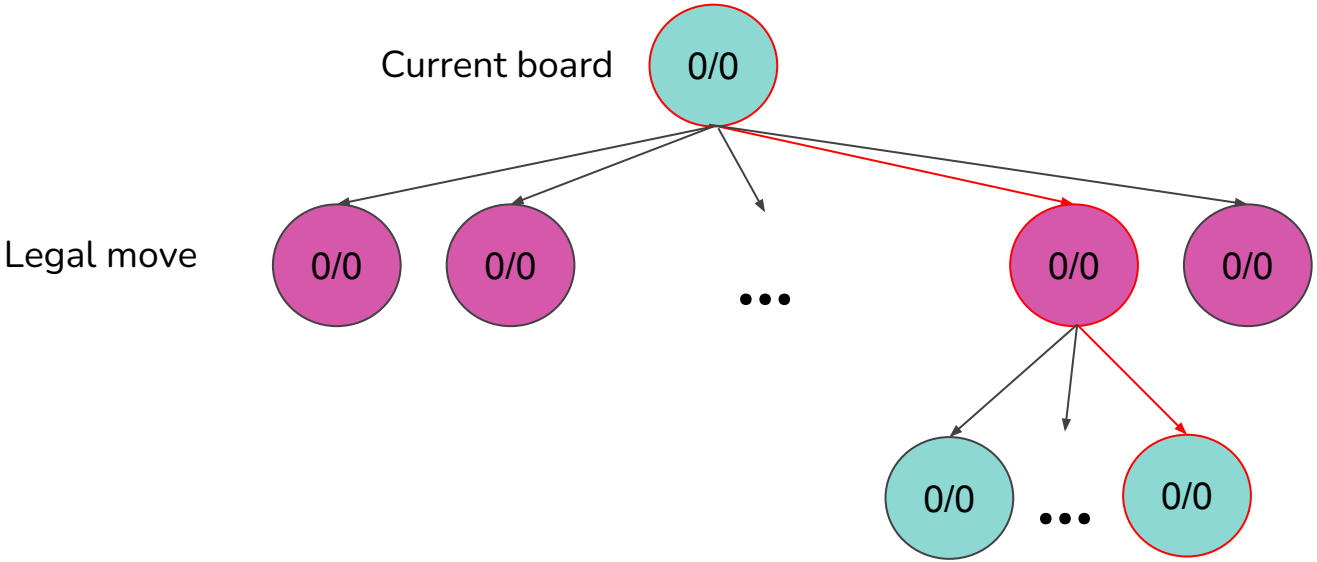


Upper Confidence bounds applies to Tree (UCT method) :

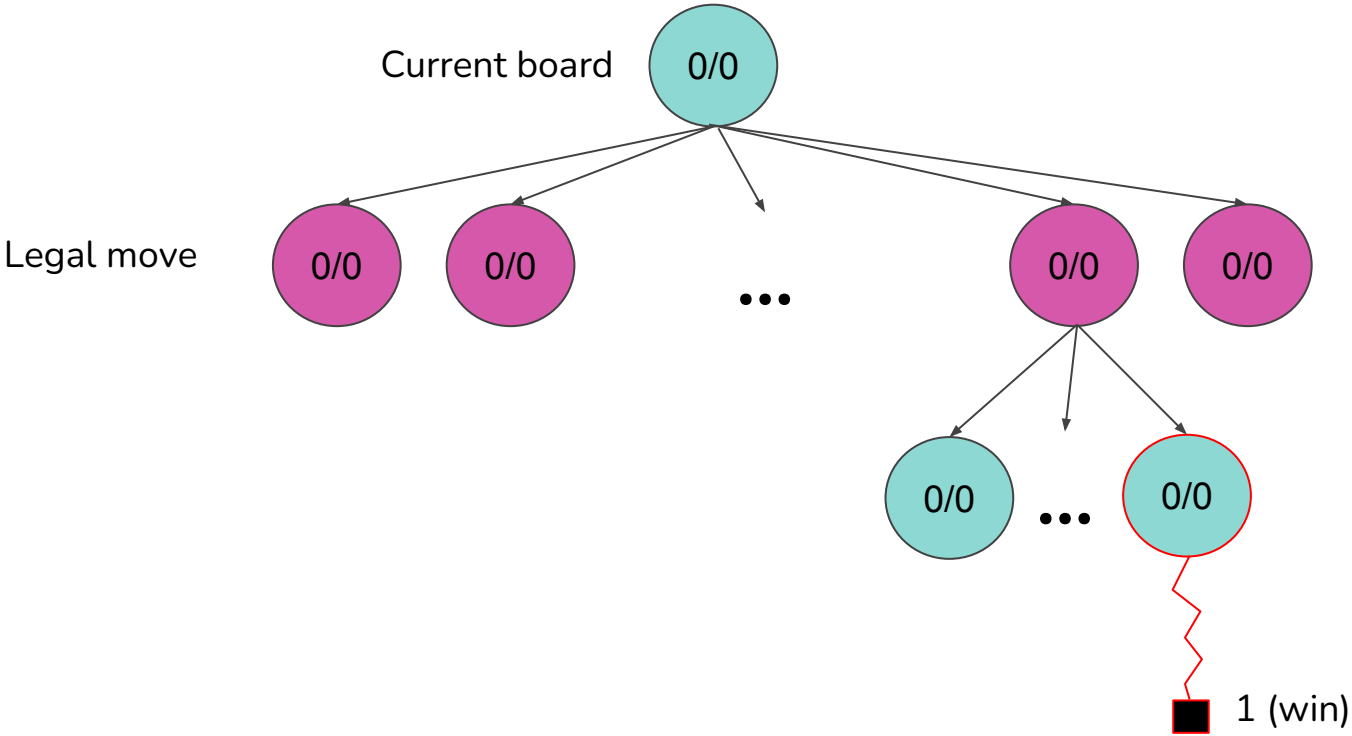
$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln t}{n_i}}$$

- w_i stands for the number of wins after the i-th move;
- n_i stands for the number of simulations after the i-th move;
- c is the exploration parameter—theoretically equal to $\sqrt{2}$; in practice usually chosen empirically;
- t stands for the total number of simulations for the node considered. It is equal to the sum of all the n_i.

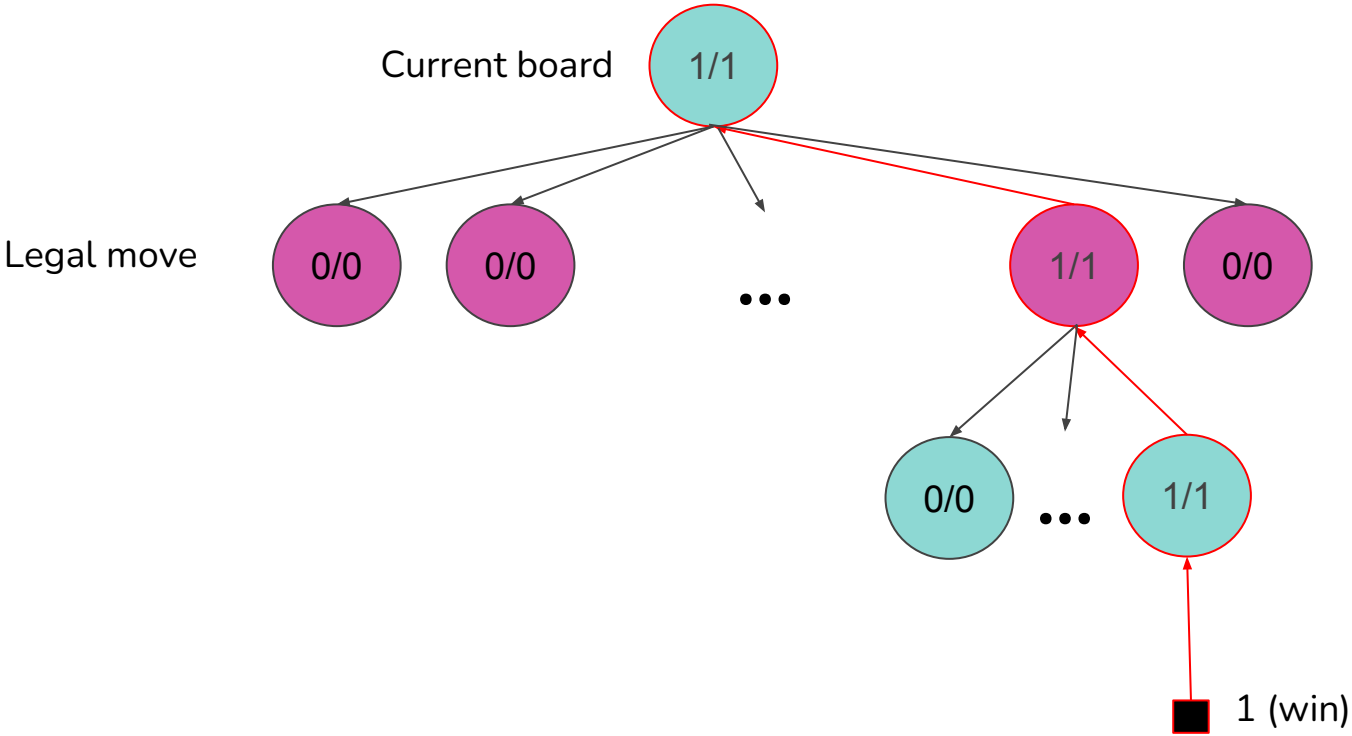
MCTS - Expansion



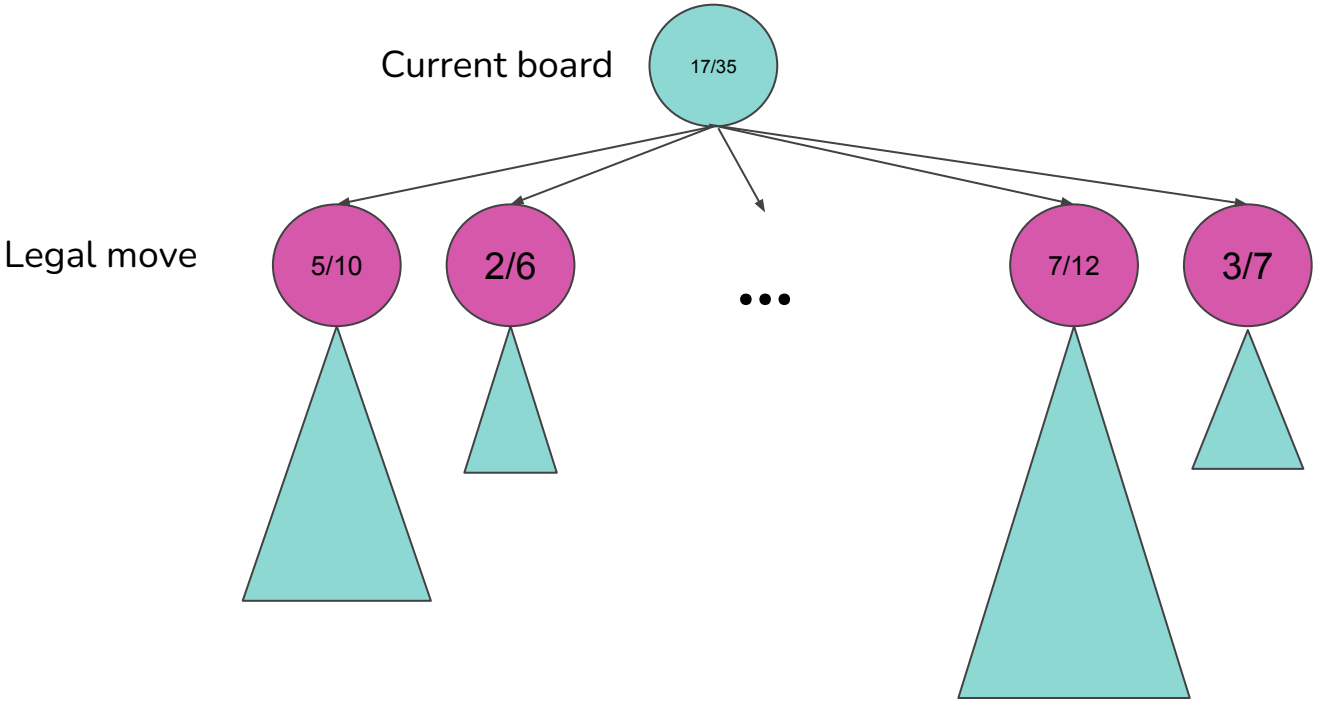
MCTS - Simulation



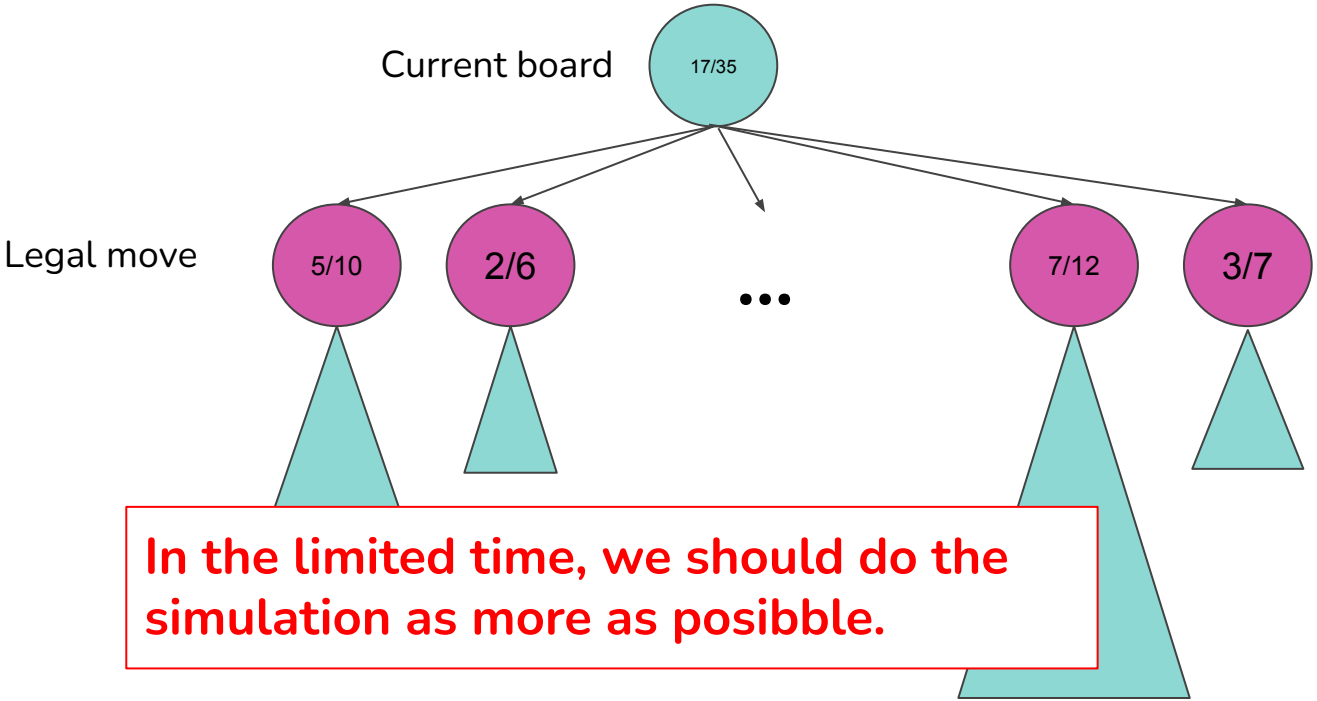
MCTS - Backpropagation



MCTS

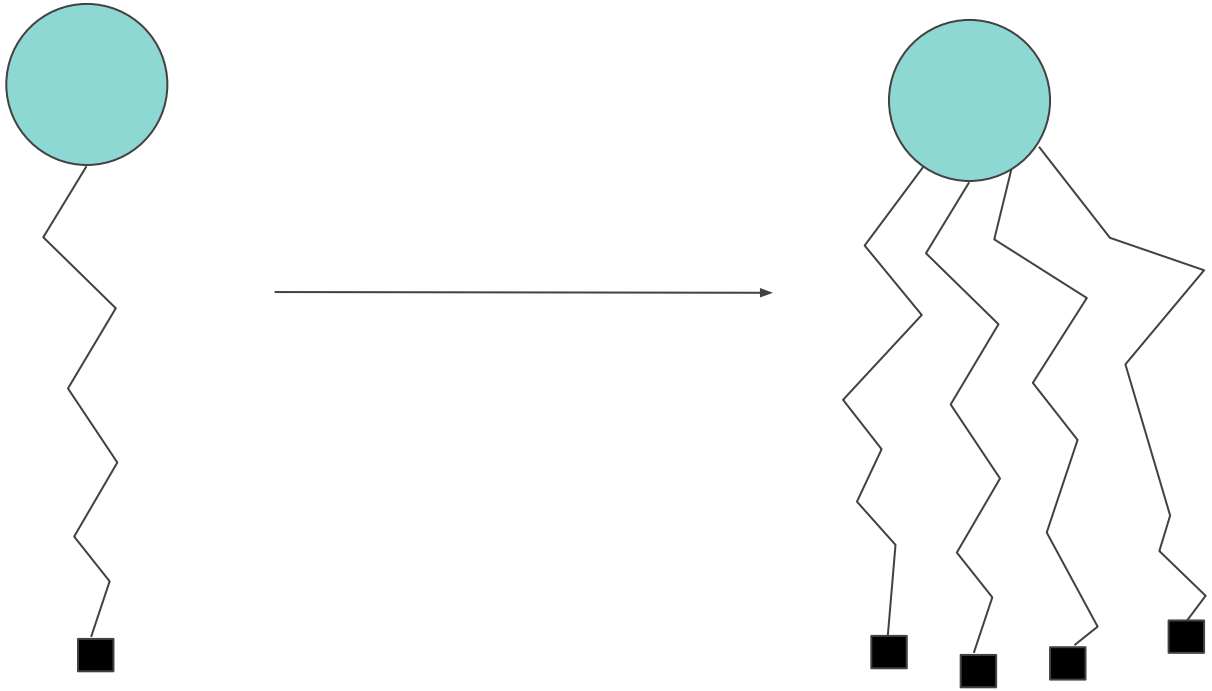


MCTS

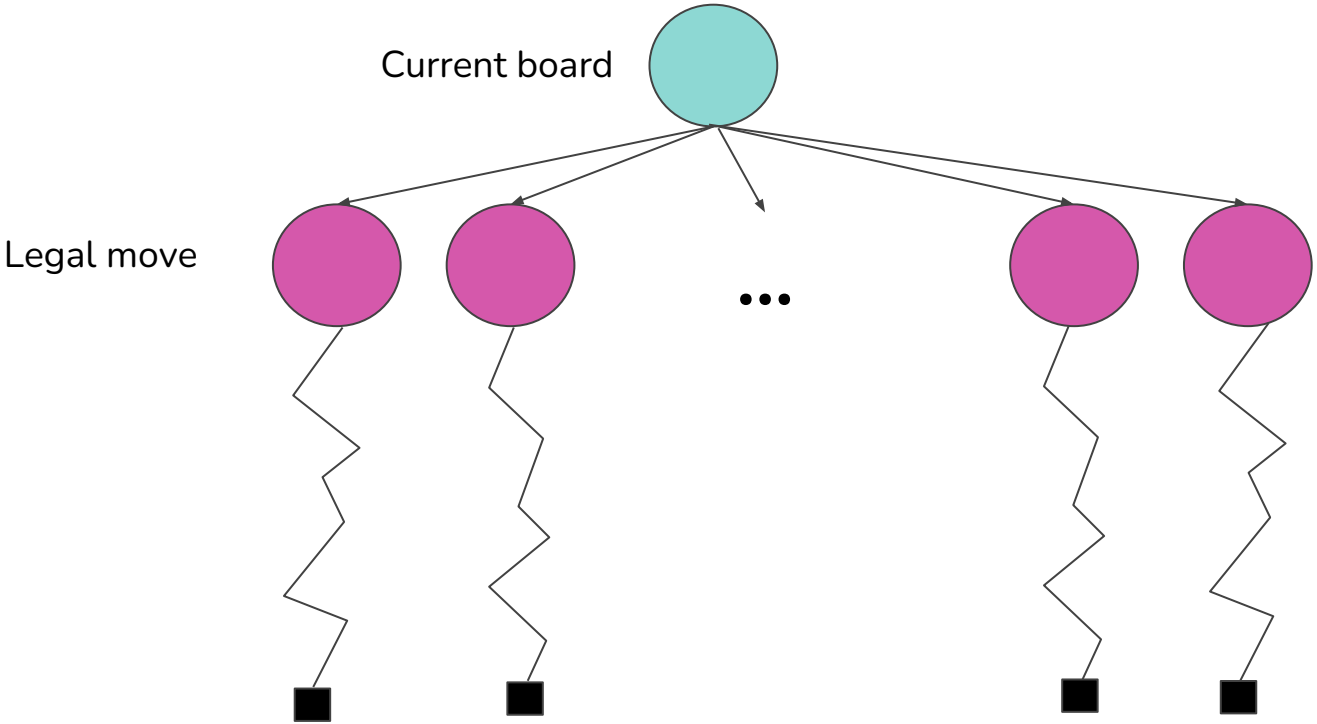


In the limited time, we should do the simulation as more as posibble.

MCTS - Leaf Parallelization



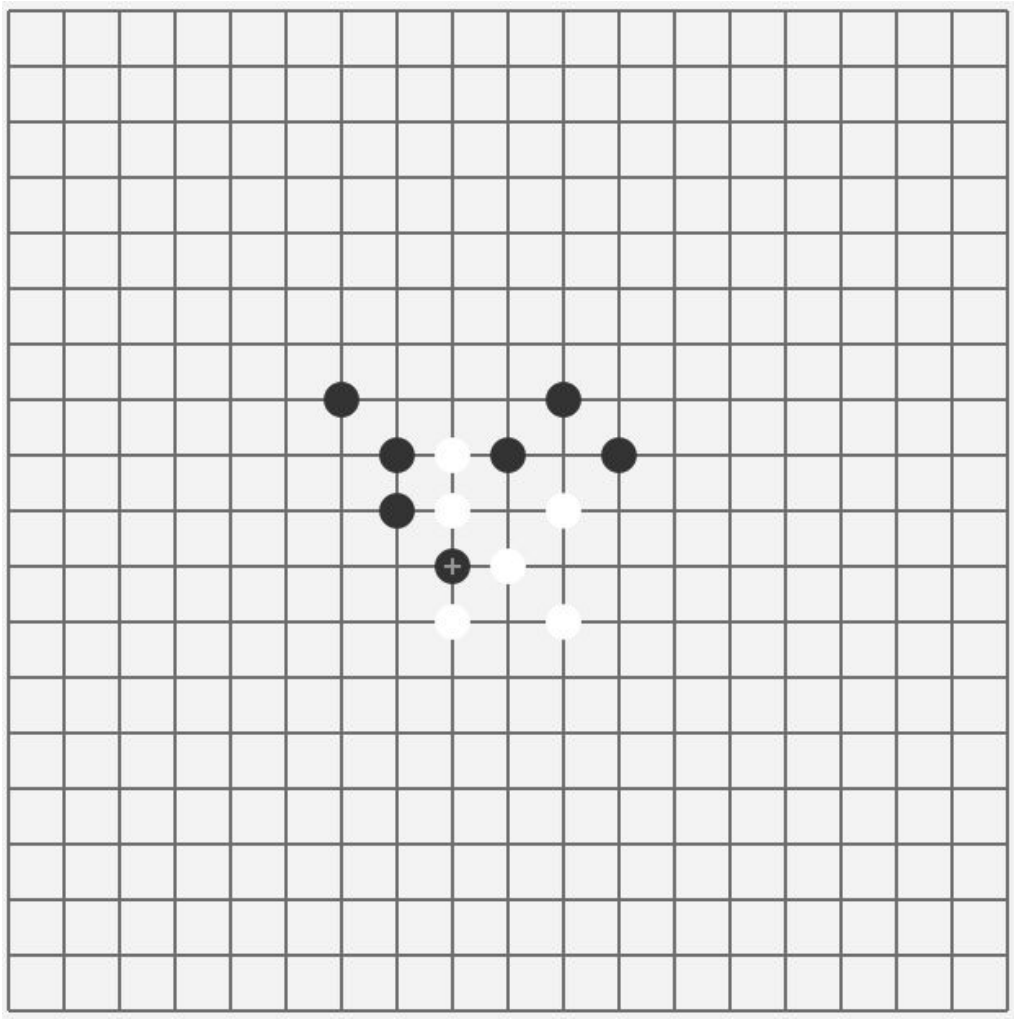
MCTS - Root Parallelization



Result

Platform

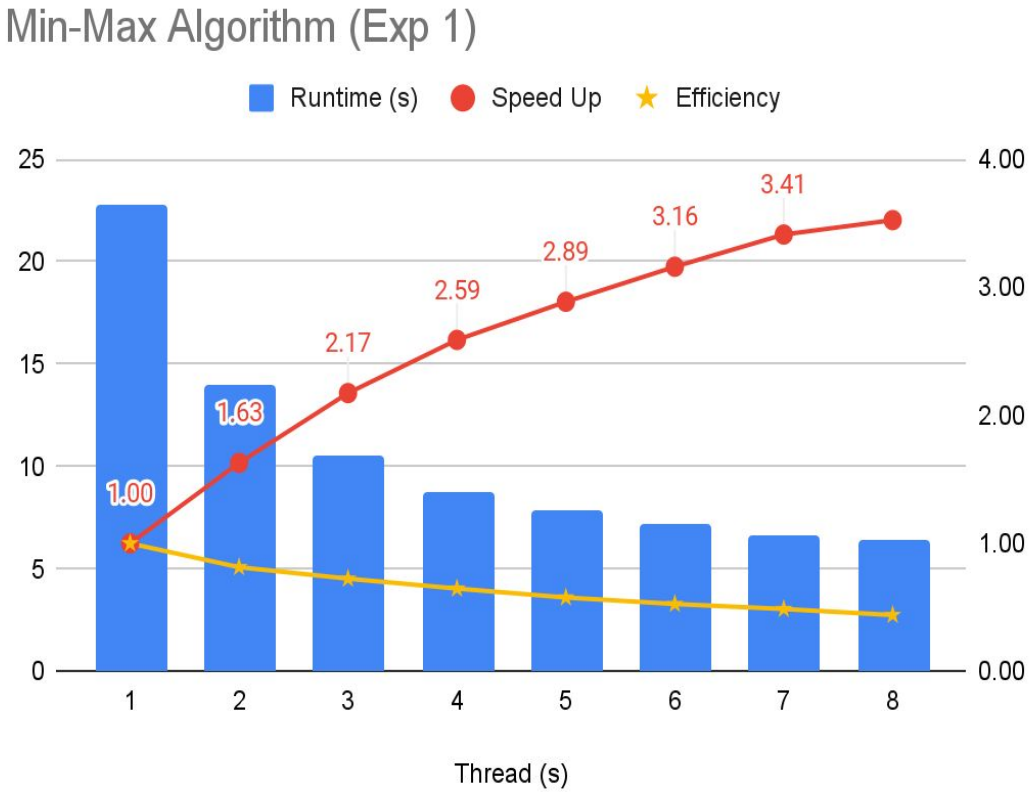
- Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-43-generic x86_64)
- Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz
- CPU MHz: 1200.000
- CPU max MHz: 3500.0000
- CPU min MHz: 1200.0000



Experiment 1

- Min-Max Algorithm
- No Alpha-Beta Pruning
- No Branch Factor
- Maximum depth = 4

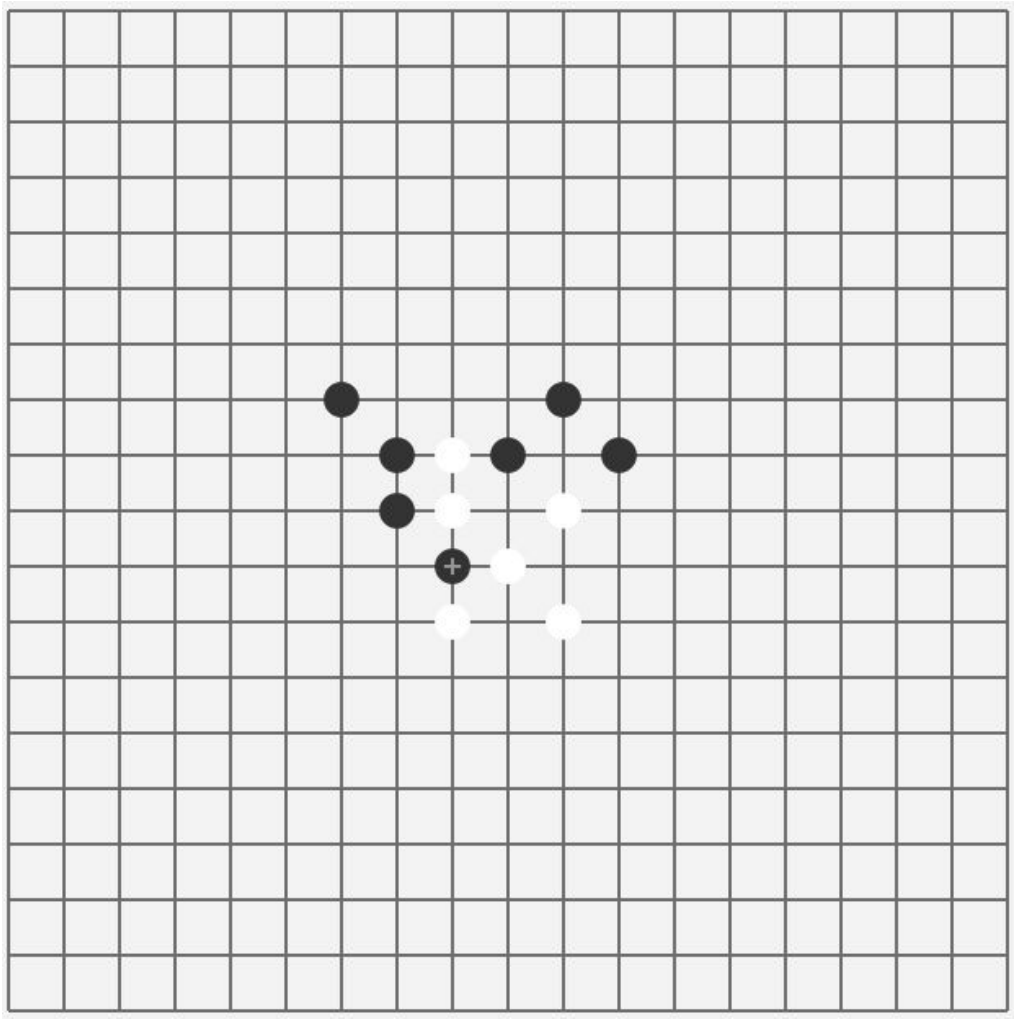
Experiment 1



- Min-Max Algorithm
- No Alpha-Beta Pruning
- No Branch Factor
- Maximum depth = 4
- Nodes = 308394
- Eval = 47258320

Experiment 1

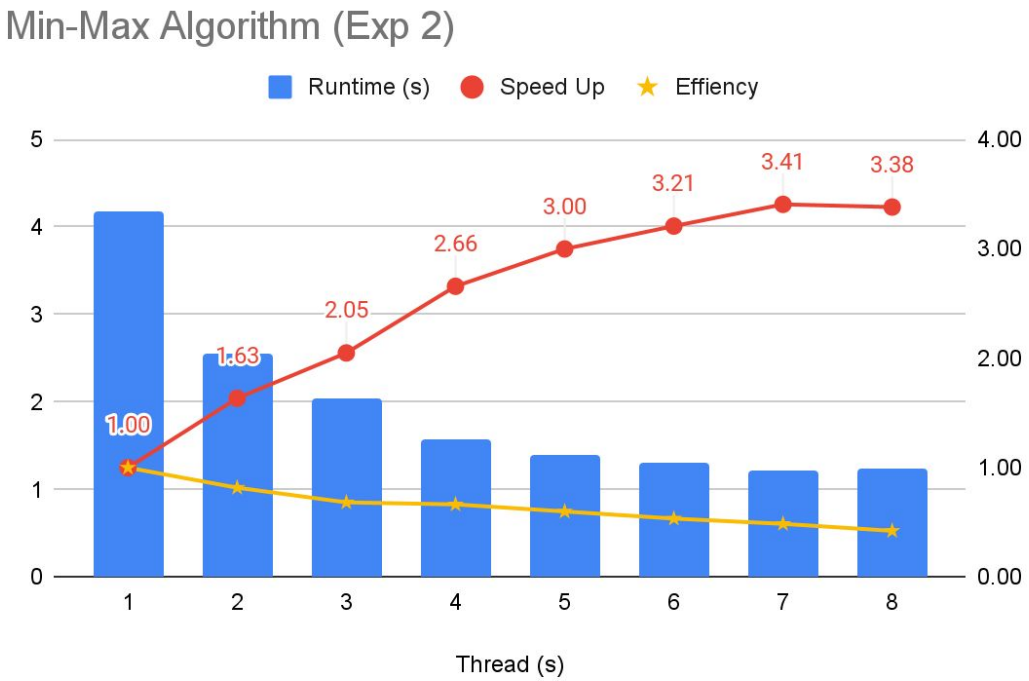
Thread(s)	Runtime (s)	Speed Up	Efficiency	Node-count	eval-count
1	22.8	1.00	1.00	308394	47258320
2	14	1.63	0.81		
3	10.489	2.17	0.72		
4	8.806	2.59	0.65		
5	7.898	2.89	0.58		
6	7.214	3.16	0.53		
7	6.682	3.41	0.49		
8	6.469	3.52	0.44		



Experiment 2

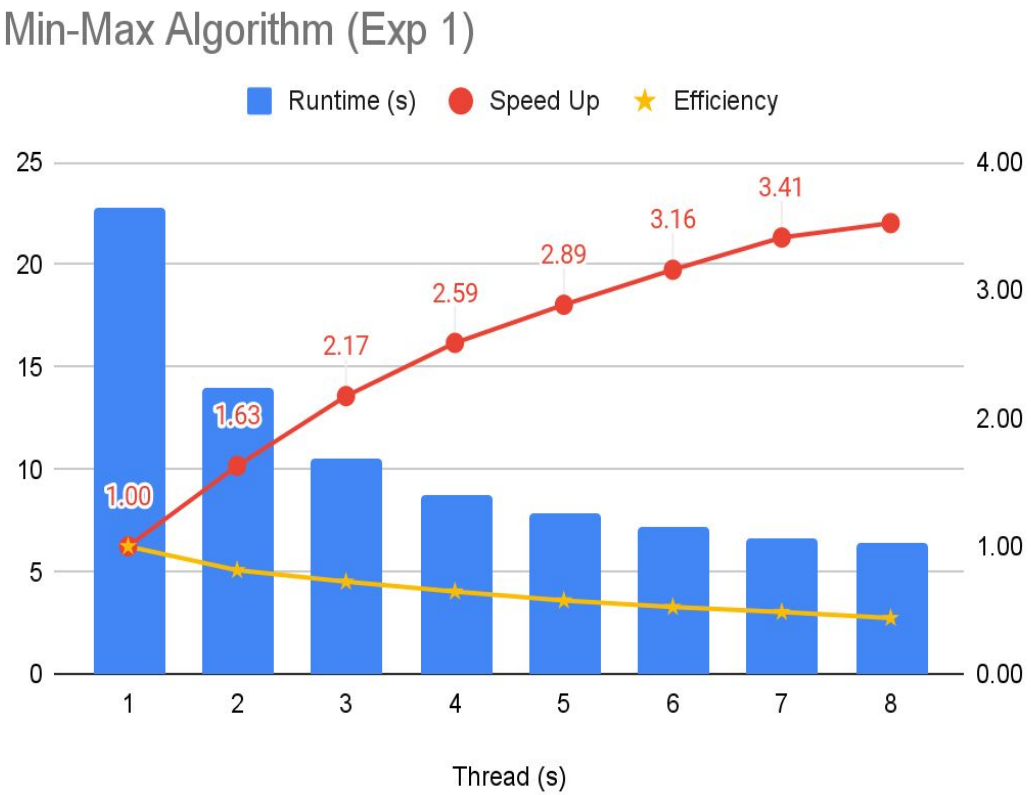
- Min-Max Algorithm
- Alpha-Beta Pruning
- No Branch Factor
- Queue Apply
- Maximum depth = 4

Experiment 2



- Min-Max Algorithm
- Alpha-Beta Pruning
- No Branch Factor
- Queue Apply
- Maximum depth = 4
- Nodes = 75211
- Eval = 12479778

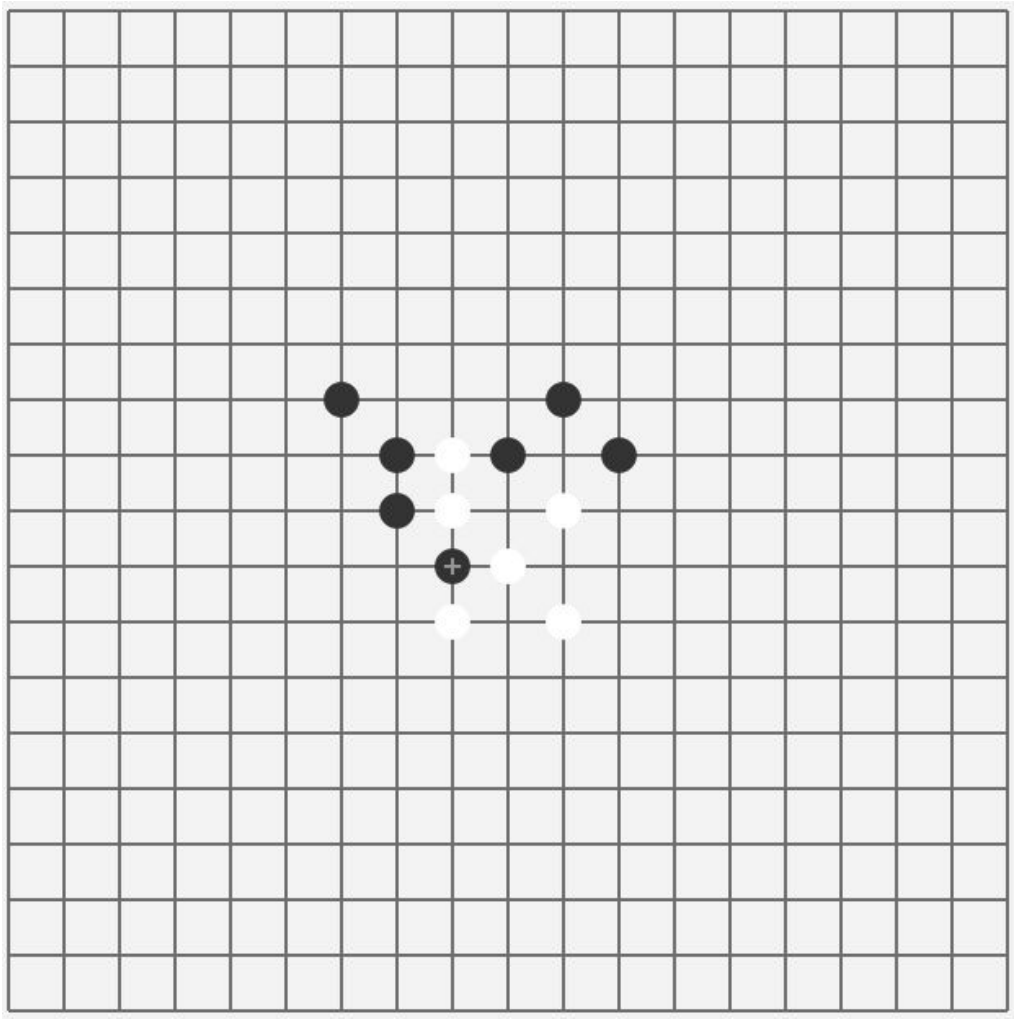
Compare to Experiment 1



- Min-Max Algorithm
- No Alpha-Beta Pruning
- No Branch Factor
- Maximum depth = 4
- Nodes = 308394
- Eval = 47258320

Experiment 2

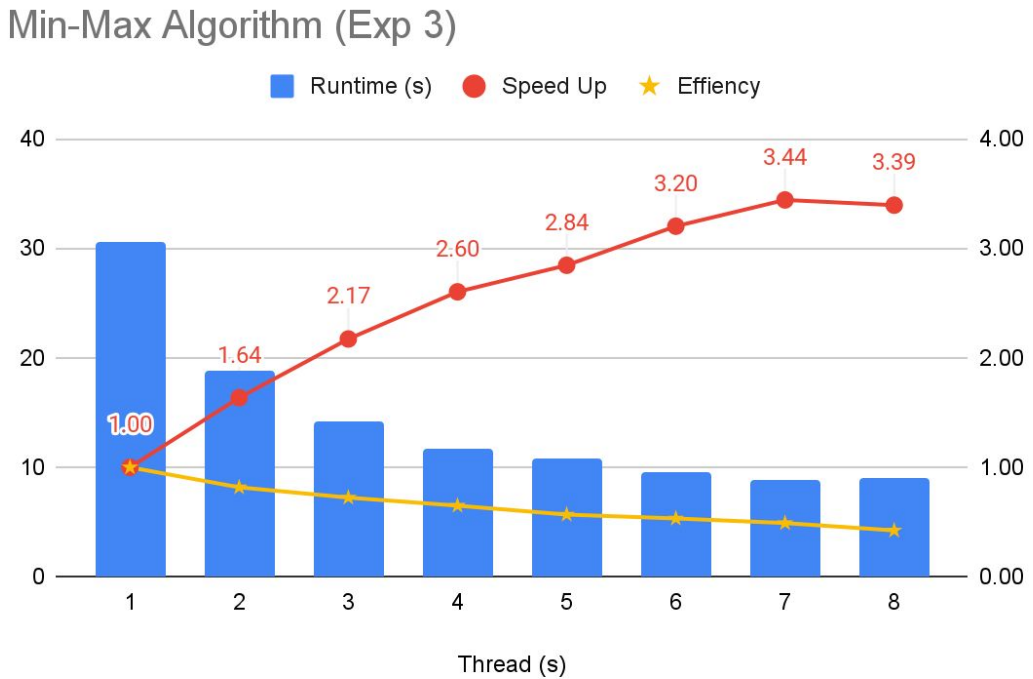
Thread(s)	Runtime (s)	Speed Up	Effiency	Node-count	eval-count
1	4.183	1.00	1.00	75211	12479778
2	2.56	1.63	0.82		
3	2.042	2.05	0.68		
4	1.574	2.66	0.66		
5	1.395	3.00	0.60		
6	1.304	3.21	0.53		
7	1.228	3.41	0.49		
8	1.237	3.38	0.42		



Experiment 3

- Min-Max Algorithm
- Alpha-Beta Pruning
- No Branch Factor
- Queue Apply
- Maximum depth = 5

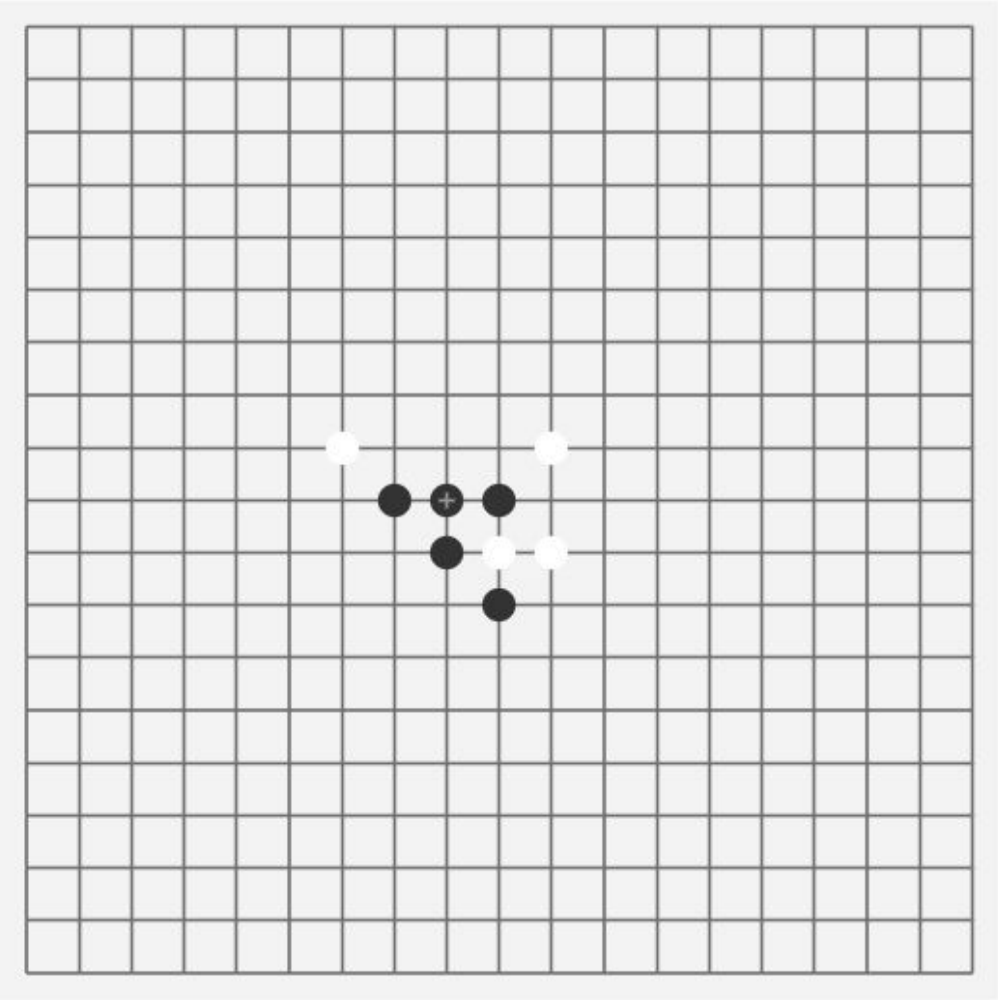
Experiment 3



- Min-Max Algorithm
- Alpha-Beta Pruning
- No Branch Factor
- Queue Apply
- Maximum depth = 5
- Nodes = 520681
- Eval = 90618342

Experiment 3

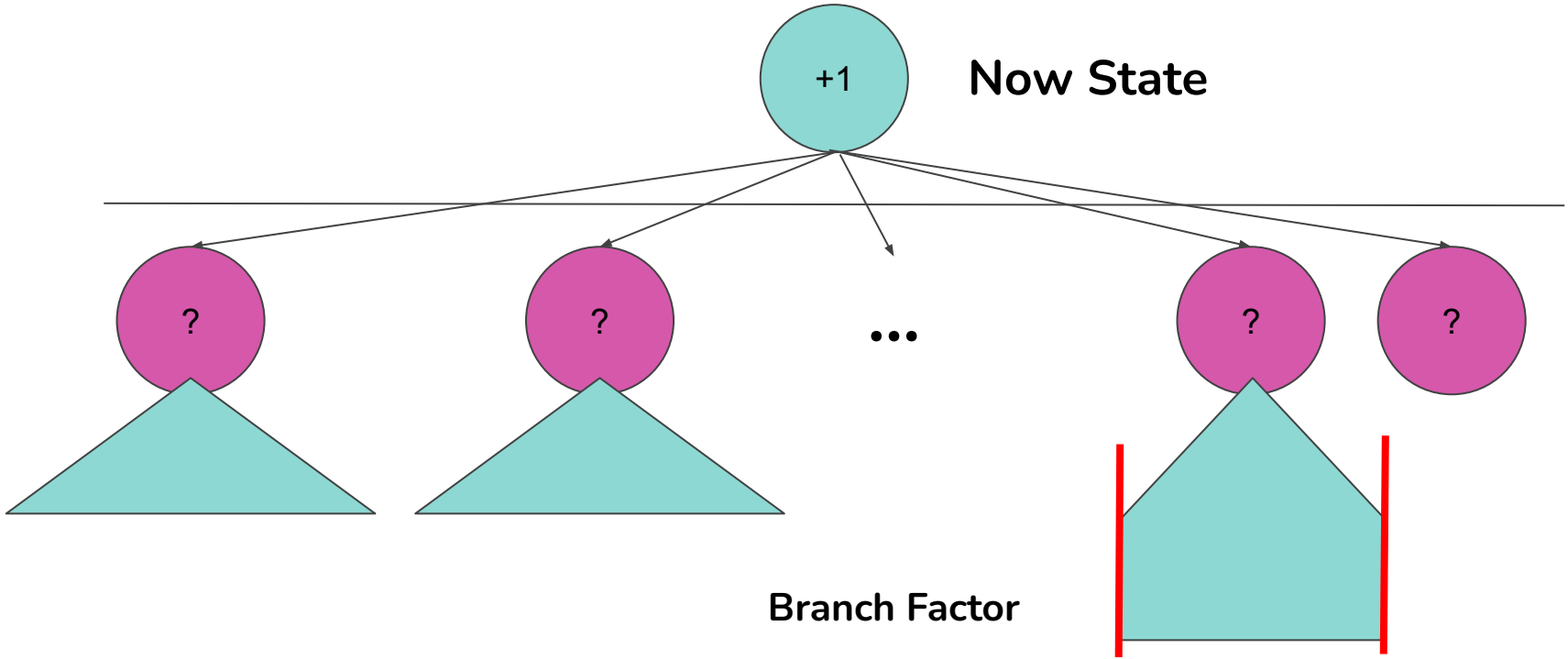
Thread(s)	Runtime (s)	Speed Up	Effiency	Node-count	eval-count
1	30.644	1.00	1.00	520861	90618342
2	18.738	1.64	0.82		
3	14.112	2.17	0.72		
4	11.778	2.60	0.65		
5	10.774	2.84	0.57		
6	9.574	3.20	0.53		
7	8.907	3.44	0.49		
8	9.032	3.39	0.42		



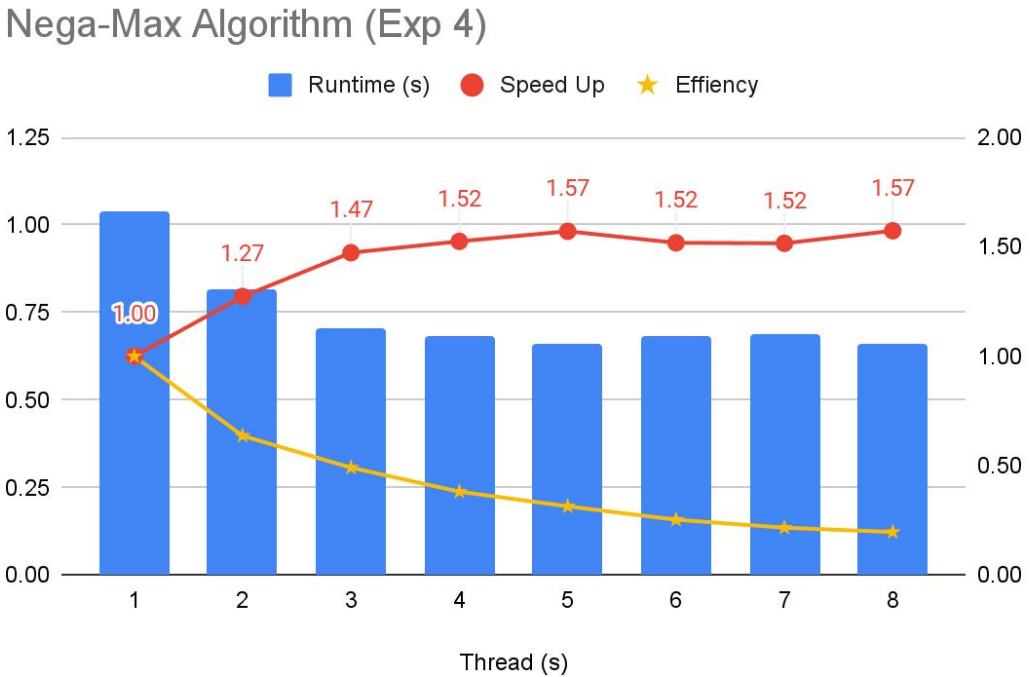
Experiment 4

- NegaMax Algorithm
- Alpha-Beta Pruning
- No queue
- Branch Factor
- Maximum depth = 8

Min-Max Tree Search



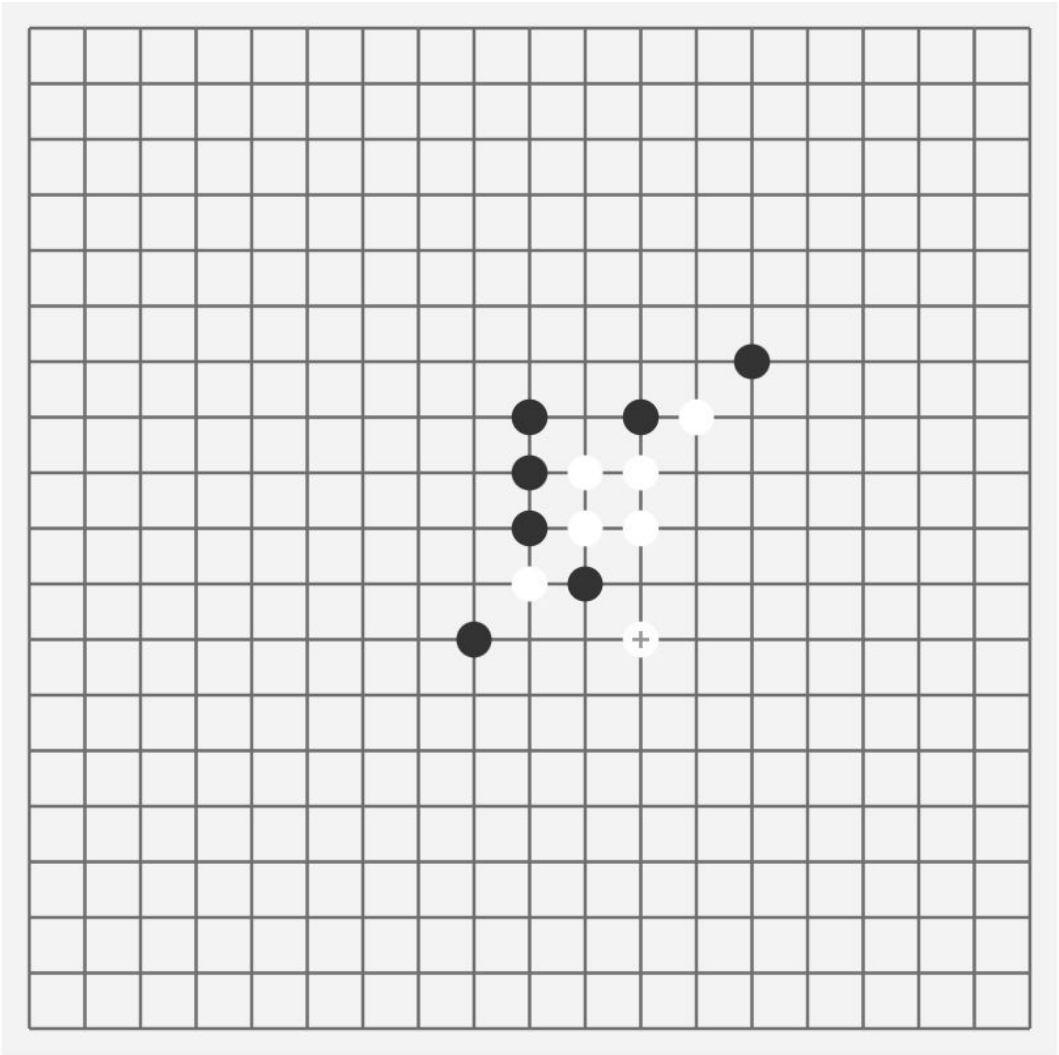
Experiment 4



- NegaMax Algorithm
- Alpha-Beta Pruning
- No queue
- Branch Factor
- Maximum depth = 8
- Nodes = 20691
- Eval = 280308

Experiment 4

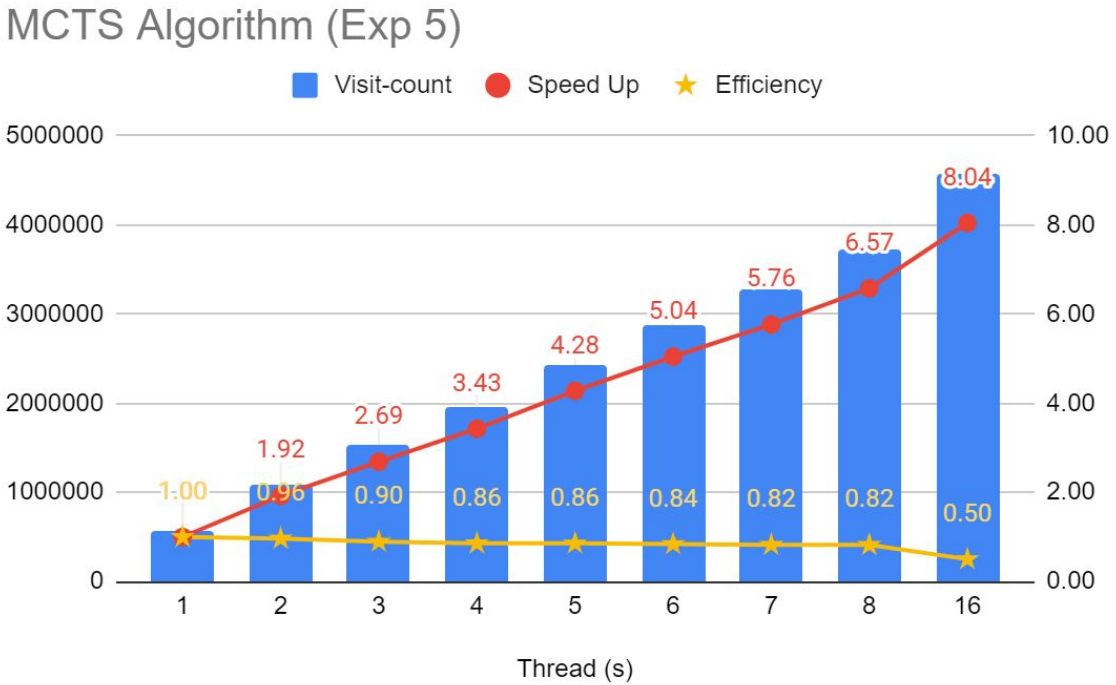
Thread(s)	Runtime (s)	Speed Up	Effiency	Node-count	eval-count
1	1.04	1.00	1.00	20691	2803038
2	0.817	1.27	0.64		
3	0.706	1.47	0.49		
4	0.682	1.52	0.38		
5	0.662	1.57	0.31		
6	0.685	1.52	0.25		
7	0.686	1.52	0.22		
8	0.661	1.57	0.20		



Experiment 5

- MCTS Algorithm
- Leaf parallelization
- Root parallelization
- Maximum time = 5 s

Experiment 5



- MCTS Algorithm
- Leaf parallelization
- Root parallelization
- Maximum time = 5

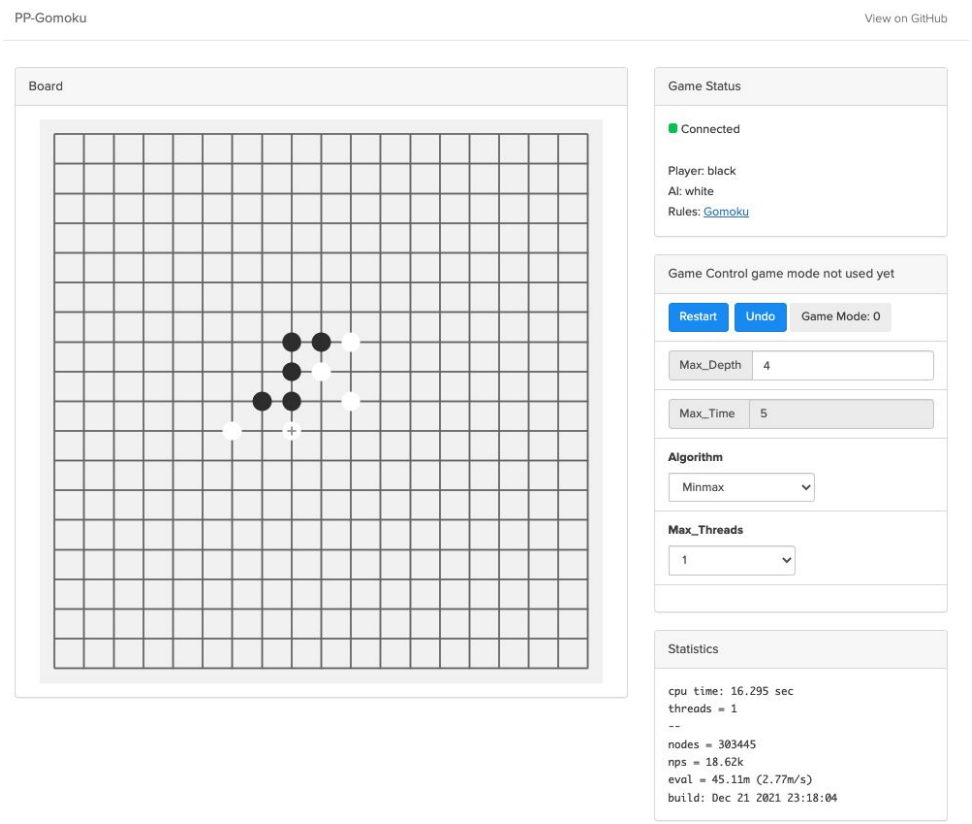
Experiment 5

Thread (s)	Visit count	Speed Up	Efficiency (Speed up/Threads)
1	567736	1.00 x	1.00
2	1092766	1.92 x	0.96
3	1525638	2.69 x	0.90
4	1945536	3.43 x	0.86
5	2428195	4.28 x	0.86
6	2863542	5.04 x	0.84
7	3272241	5.76 x	0.82
8	3732536	6.57 x	0.82
16	4564848	8.04 x	0.50

Demo

GUI:

yunzhu-li / blupig-gomoku Public



DEMO

Conclusion

Contributions of each member

310554039 Tseng, Yang	MCTS GUI 40%
0816153 Tsung Fang, Chen	Min-Max Nega-Max GUI 60%

Conclusion

- Tree Search on Gomoku
  - Min-Max with alpha-beta pruning **30% speed up / per thread**
  - MCTS Root & Leaf Parallelization **70% speed up / per thread**
- Too much serial, low efficiency
- MPI, Cuda