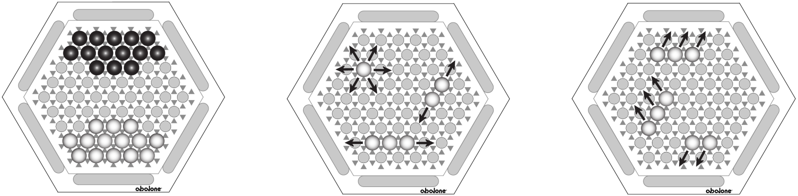
Exploration of Abalone game-playing agents

Ture Claussen

2021-06-14

Rules



(a) Starting position (b) "In-line" moves (c) "Side-step" moves

Agent design: PEAS

Performance measure: Win/loss, number of moves, time to deliberate

Environment: Digital playing board

Actuators: Move marbles, display text to CLI

Sensors: Position of marbles

Agent design: Environment

- fully observable
- two-agent
- competitive
- sequential
- static and discrete

State space complexity

$$\sum_{k=8}^{14} \sum_{m=9}^{14} \frac{61!}{k!(61-k)!} \times \frac{(61-k)!}{m!((61-k)-m)!}$$

Game tree complexity

- Average branching factor b of 60
- Average length of game d of 87 [**?**]

$$b^d = 60^{87}$$

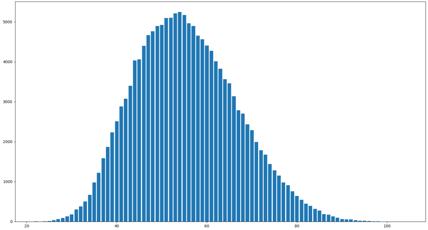
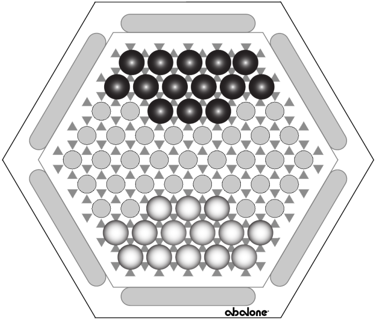


Figure: Counts of moves available for random for random player in 5 games

Complexity Comparision

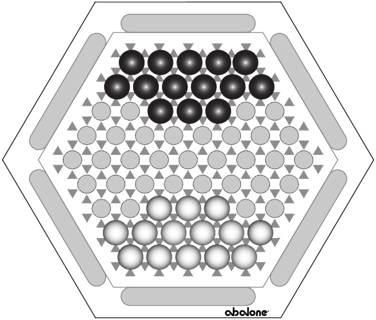
Game	state-space complexity (log)	game-tree complexity (log)
Tic-tac-toe	3	5
Reversi	28	58
Chess	46	123
Abalone	24	154
Go	172	360

Heuristics: Ajdacency



$$\text{adjacency} = n_{\text{self}} - n_{\text{opponent}}$$

Heuristics: Distance



$$\text{distance} = n_{\text{self}} - n_{\text{opponent}}$$

Heuristics: Marble ratio

$$\text{marbleRatio} = n_{\text{won}} - n_{\text{lost}}$$

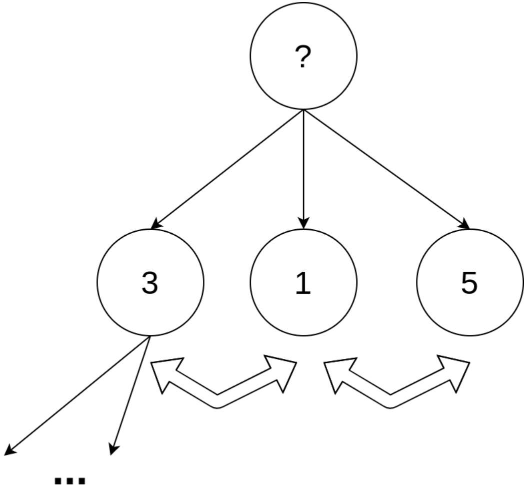
Heuristics: Win and loss

$$\text{winLoss} = \begin{cases} 1 \text{ if game won} \\ -1 \text{ otherwise} \end{cases}$$

Heuristic	weight
adjacency	1
distance	-1.5
marbleRatio	100
winLoss	100000

Table: Weights for the linear combination

Alpha-beta pruning agent: Move ordering



Alpha-beta pruning agent: Move ordering

- Move capturing marble: $+3$
- Move pushing marble: $+1$
- Move involving $2/3$ marbles: $+1/+2$

Alpha-beta pruning agent: Move ordering

Depth	Without ordering	Evaluation 1	Evaluation 2	$\sqrt{b^d}$
1	45	45	45	8
2	1594	304	132	60
3	9755	4971	2423	464
4	457309	94650	6918	3600

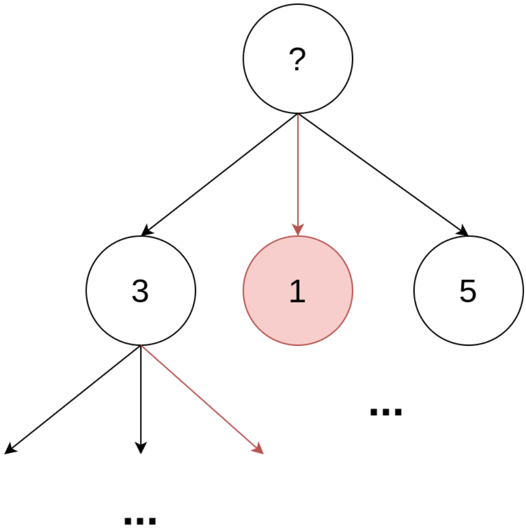
Table: Nodes visited with/without move ordering and the optimal case

Alpha-beta pruning agent: Transposition table

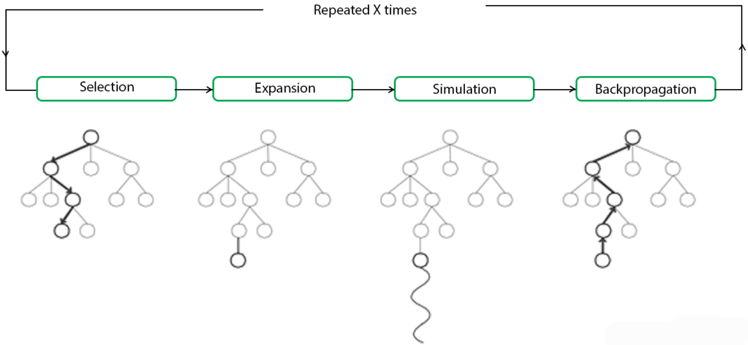
	1	2	...
1	129310293812	929310293912	...
2	889394293012	426317297917	...
...	...	...	...

Table: Zobrist hash table

Alpha-beta pruning agent: Branch cutting



Monte carlo search agent



Monte carlo search agent: UCB

$$UCB(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(Parent(n))}{N(n)}}$$

Monte carlo search agent: Playout policy

- Random moves bad as branching factor is high
- Choose moves based on evaluation function

Face-off

Black player	White player	Marbles lost b	Marbles lost w	time p. move b	time p. move w	total moves (avg)	n
AlphaBeta (d=3)	Random	0.2	6.0	11.11	0.0	57.6	5
AlphaBeta (d=4)	Random	0.0	6.0	142.8	0.0	52.4	5
AlphaBeta (d=3)	AlphaBetaFast (d=3)	6.0	5.0	10.02	4.16	92	1
MonteCarloPure (t=20s)	RandomPlayer	5.0	0.0	20.33	0.0	1008.0	1
MonteCarloImproved (t=20s)	RandomPlayer	0.0	6.0	20.05	0.0	306.0	1
MonteCarloImproved (t=20s)	AlphaBetaFast (d=3)	0.0	6.0	69.0	20.06	6.24	1

Conclusion

- Implementation simple, improvement requires a lot of engineering
- MCTS performed badly but with more time still promising