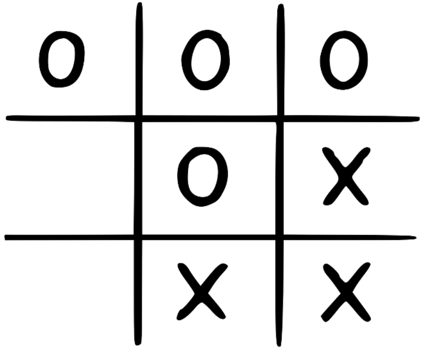
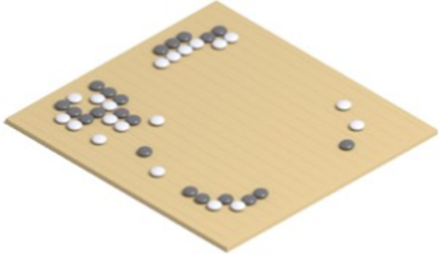
Stochastic Planning in Games

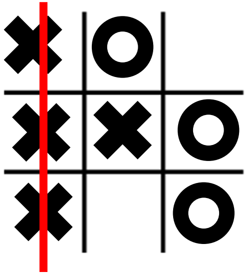
Peter Müller

Why Games?

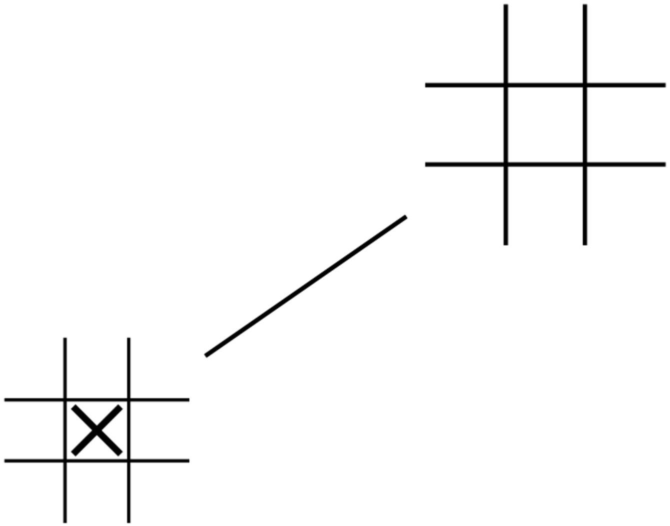




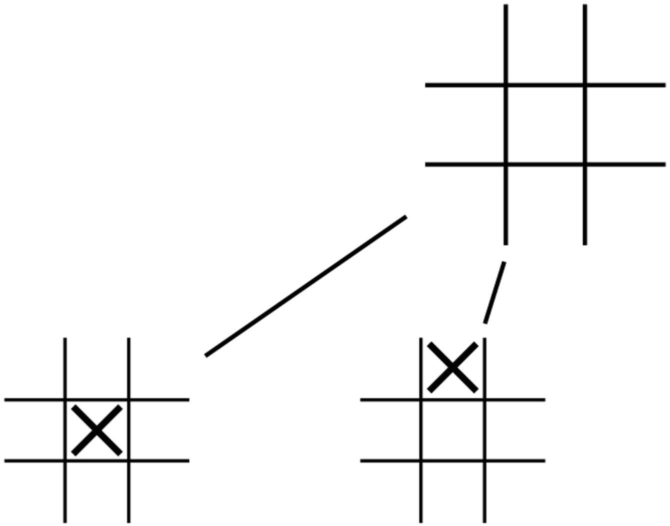
Tic-Tac-Toe



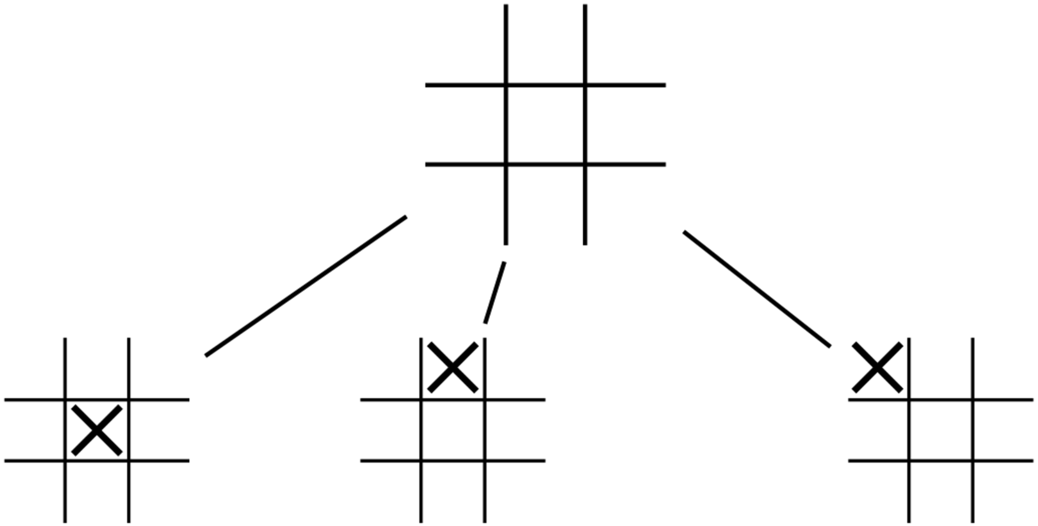
Tic-Tac-Toe



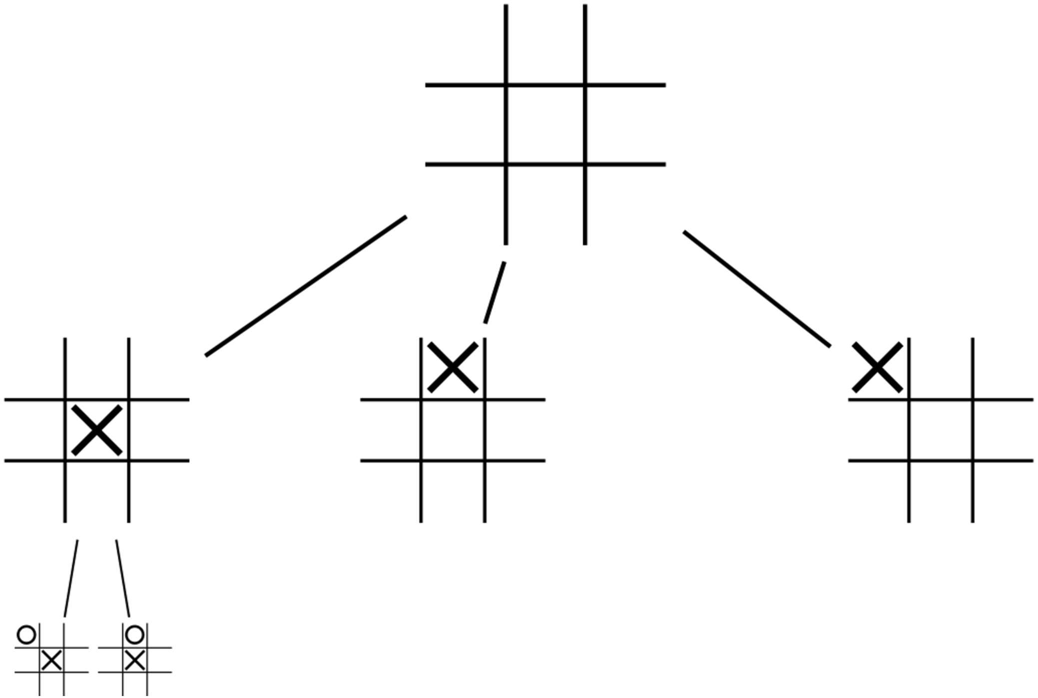
Tic-Tac-Toe



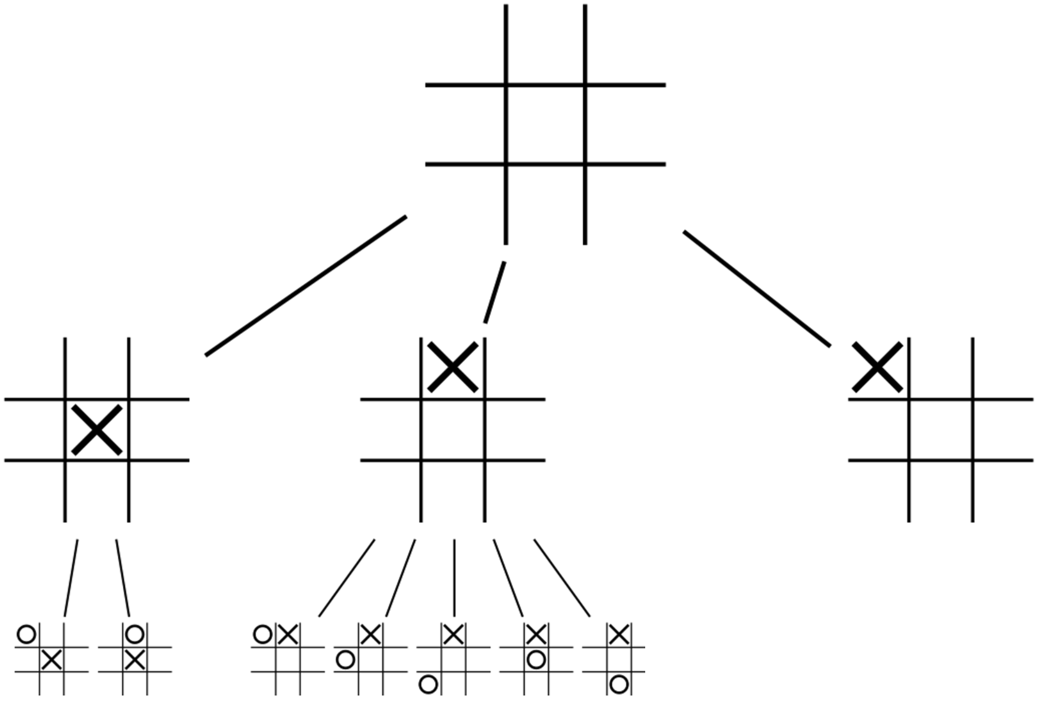
Tic-Tac-Toe



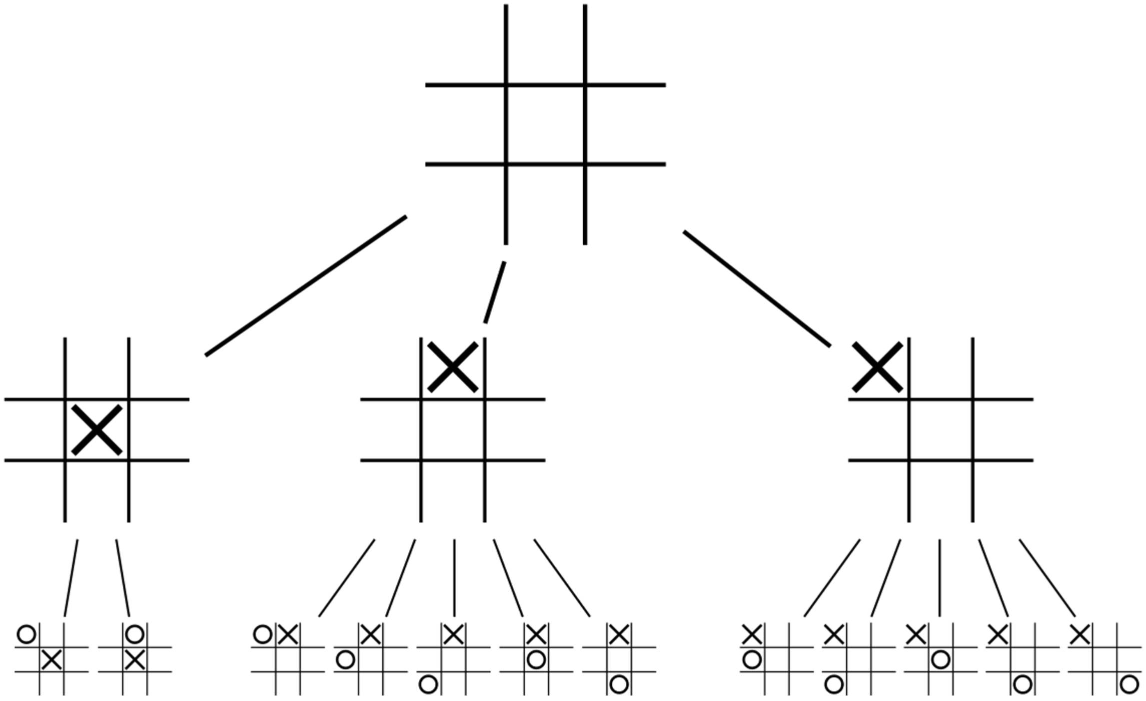
Tic-Tac-Toe



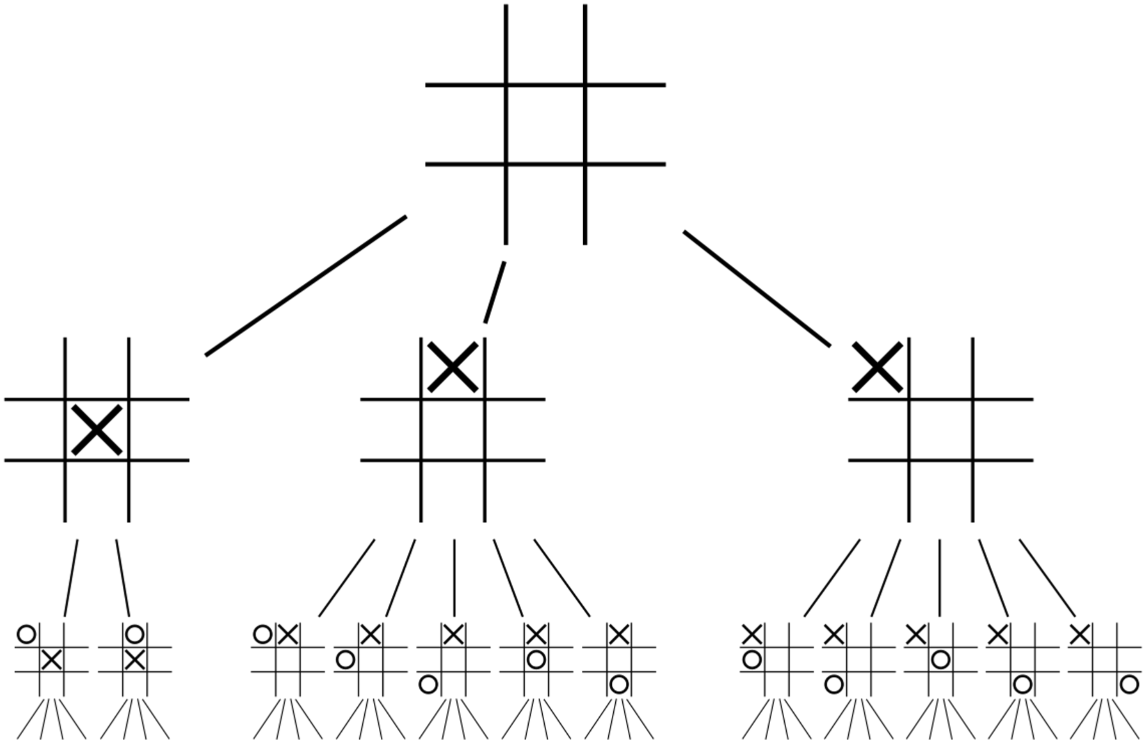
Tic-Tac-Toe



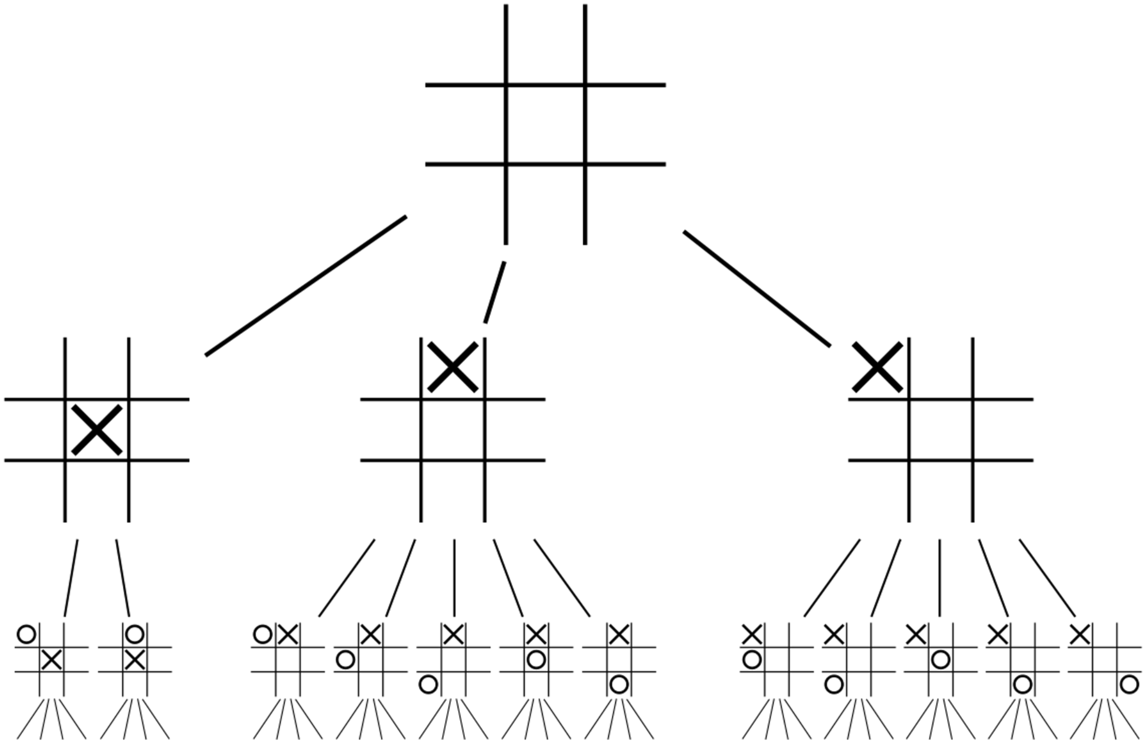
Tic-Tac-Toe



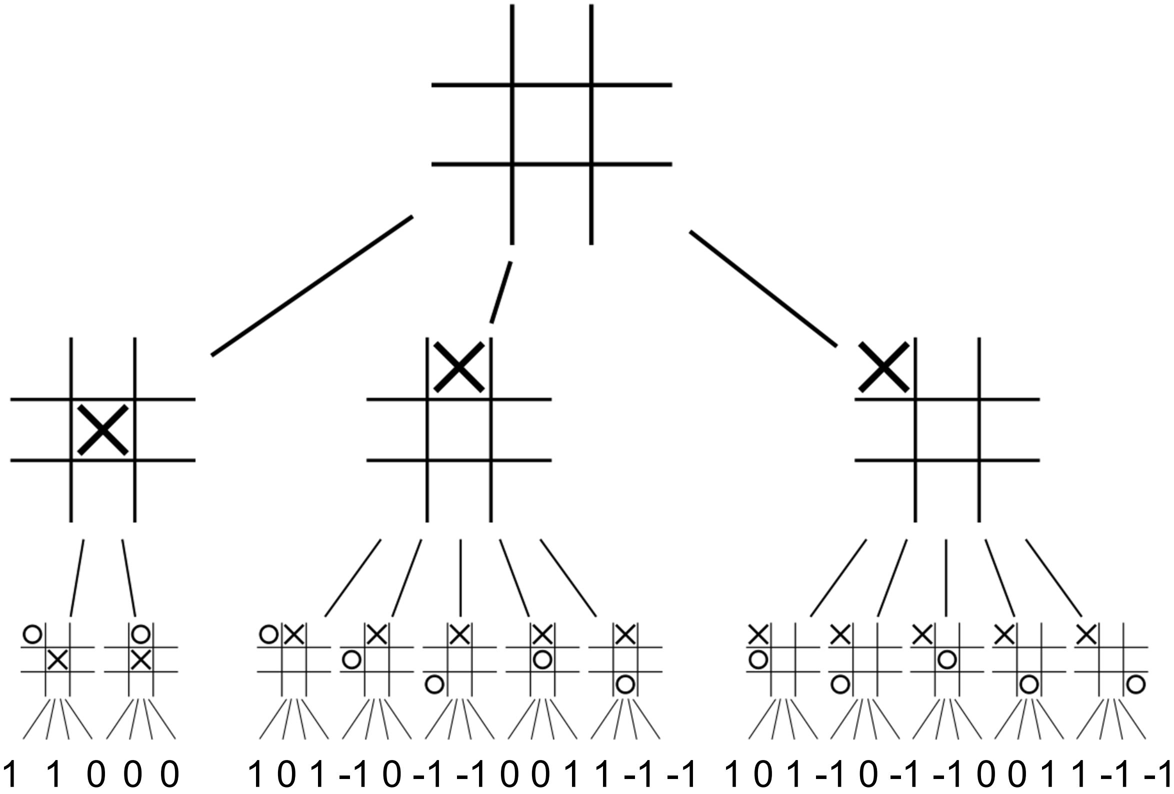
Tic-Tac-Toe



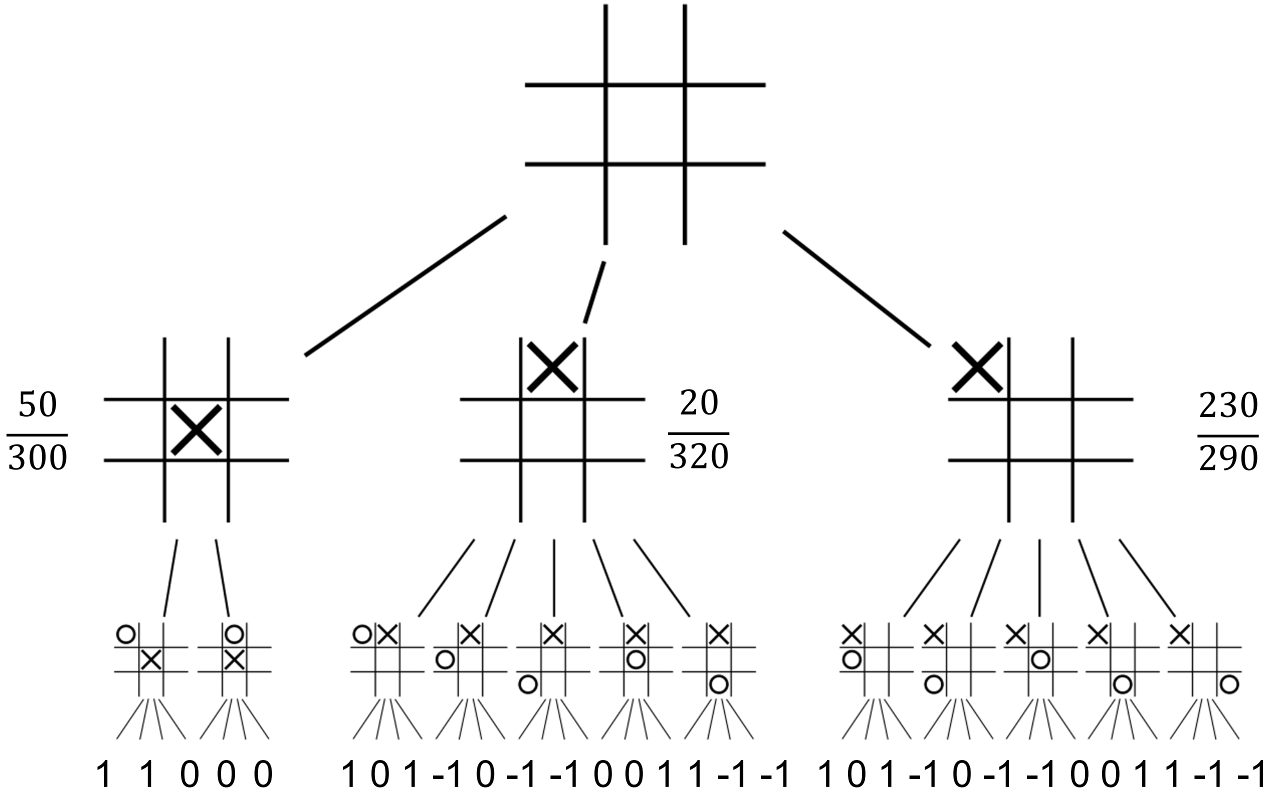
Tic-Tac-Toe



Tic-Tac-Toe



Tic-Tac-Toe

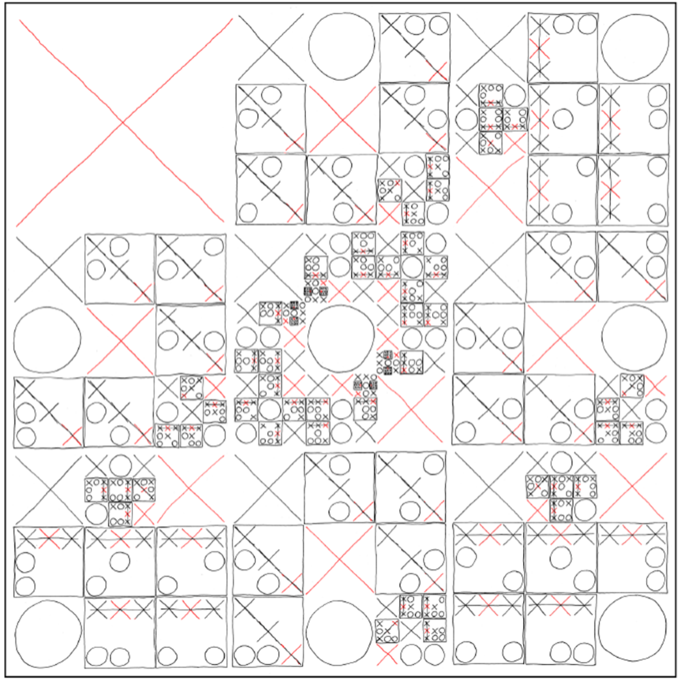


$\frac{50}{300}$
$\frac{20}{320}$
$\frac{230}{290}$
1 1 0 0 0
1 0 1 -1 0 -1 -1 0 0 1 1 -1 -1
1 0 1 -1 0 -1 -1 0 0 1 1 -1 -1

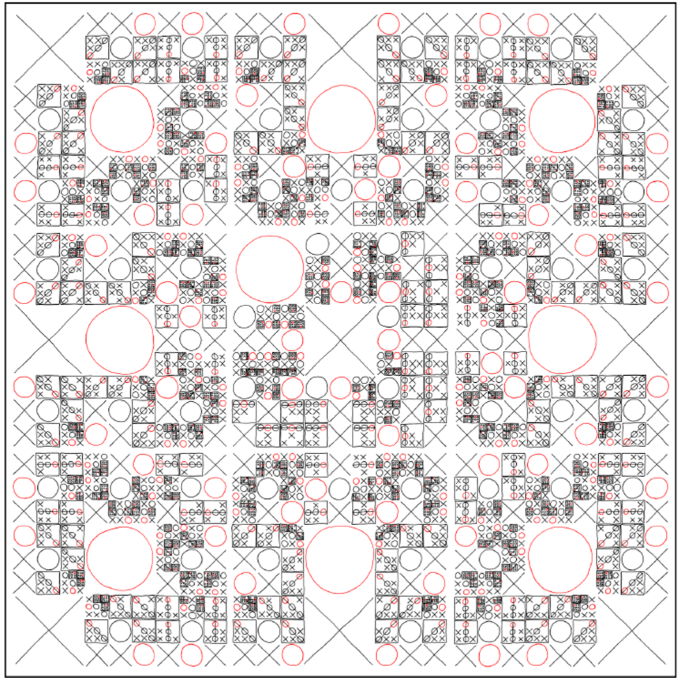
COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES

YOUR MOVE IS GIVEN BY THE POSITION OF THE LARGEST RED SYMBOL ON THE GRID. WHEN YOUR OPPONENT PICKS A MOVE, ZOOM IN ON THE REGION OF THE GRID WHERE THEY WENT. REPEAT.

MAP FOR X:

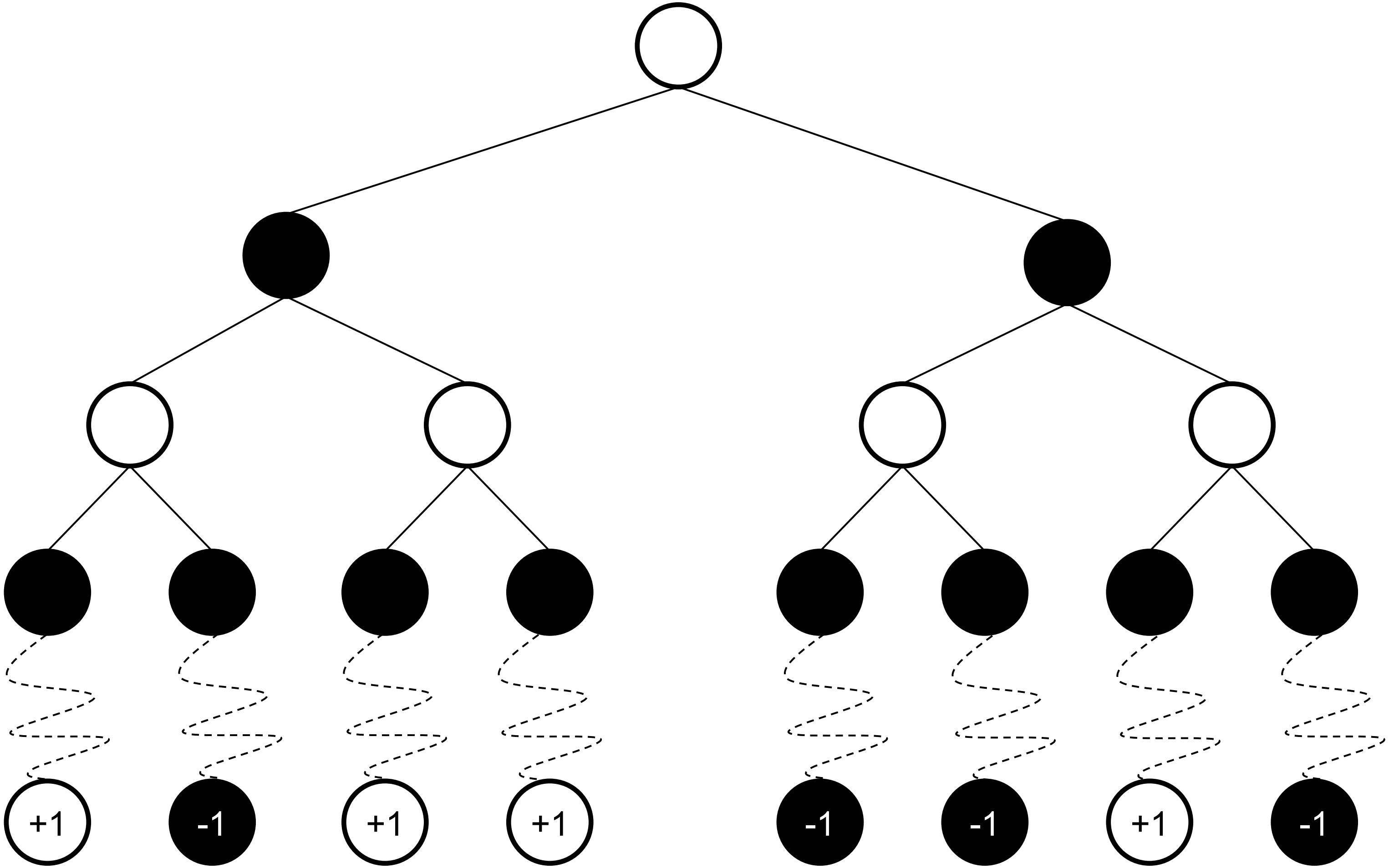


MAP FOR O:

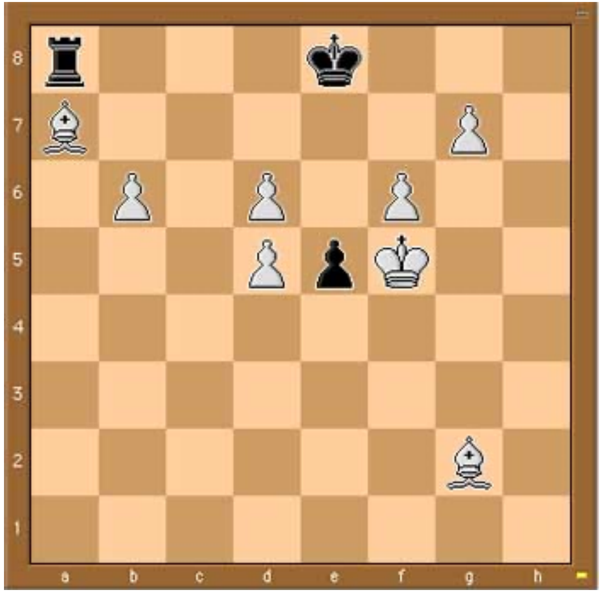


Chess

- Branching Factor: 35
- Game Length: 80

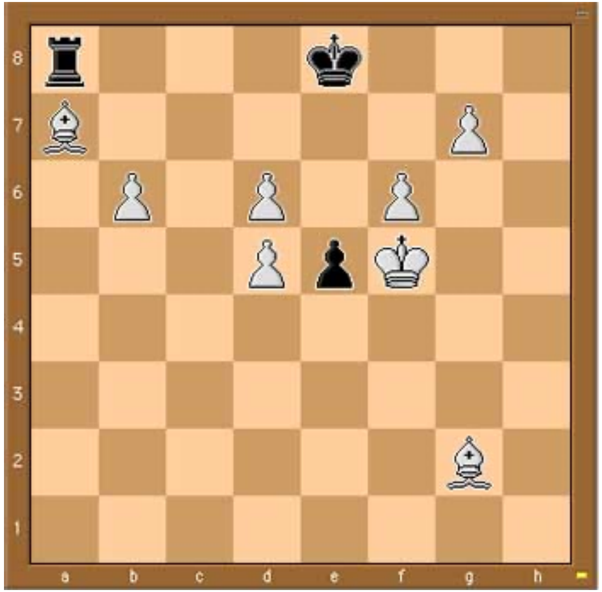

+1
-1
+1
+1
-1
-1
+1
-1

Evaluation Function



White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

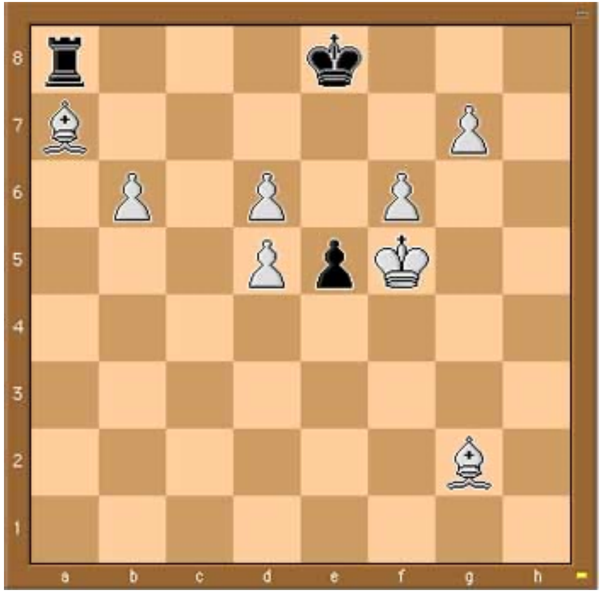
Evaluation Function



White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

Black: 1x Rook (5 points) + 1x Pawn (1 point) = 6 points

Evaluation Function

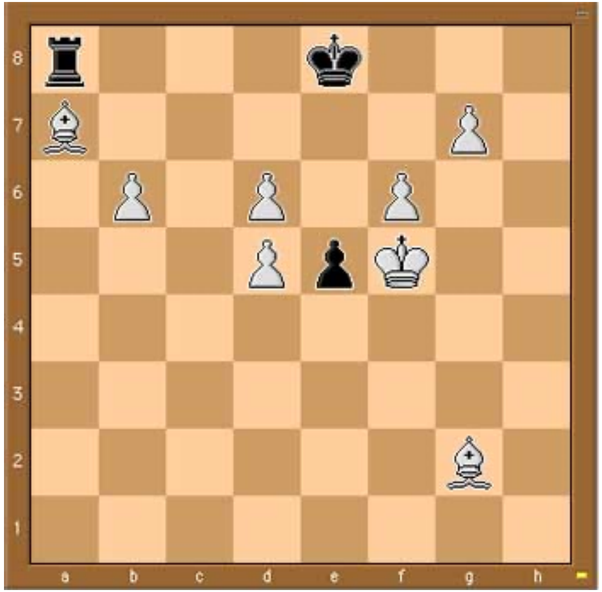


White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

Black: 1x Rook (5 points) + 1x Pawn (1 point) = 6 points

Eval: 11 - 6 = 5 points

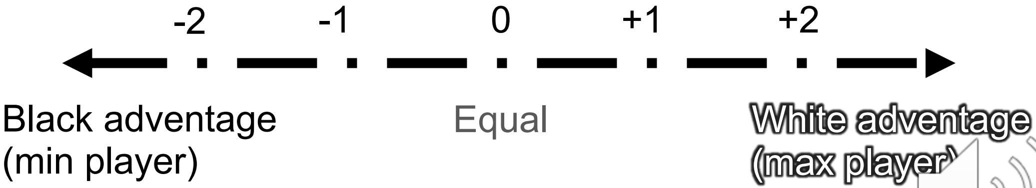
Evaluation Function



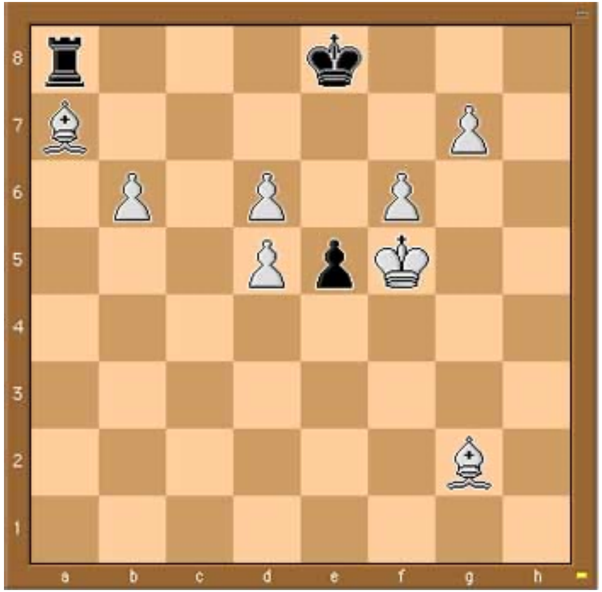
White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

Black: 1x Rook (5 points) + 1x Pawn (1 point) = 6 points

Eval: 11 - 6 = 5 points



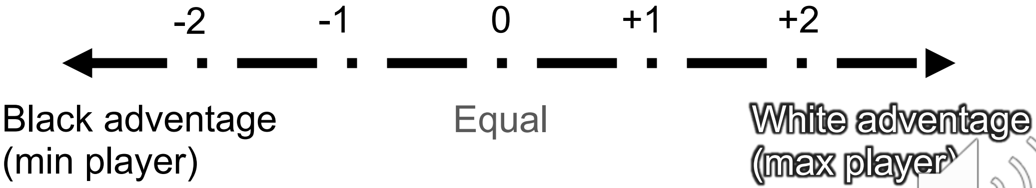
Evaluation Function

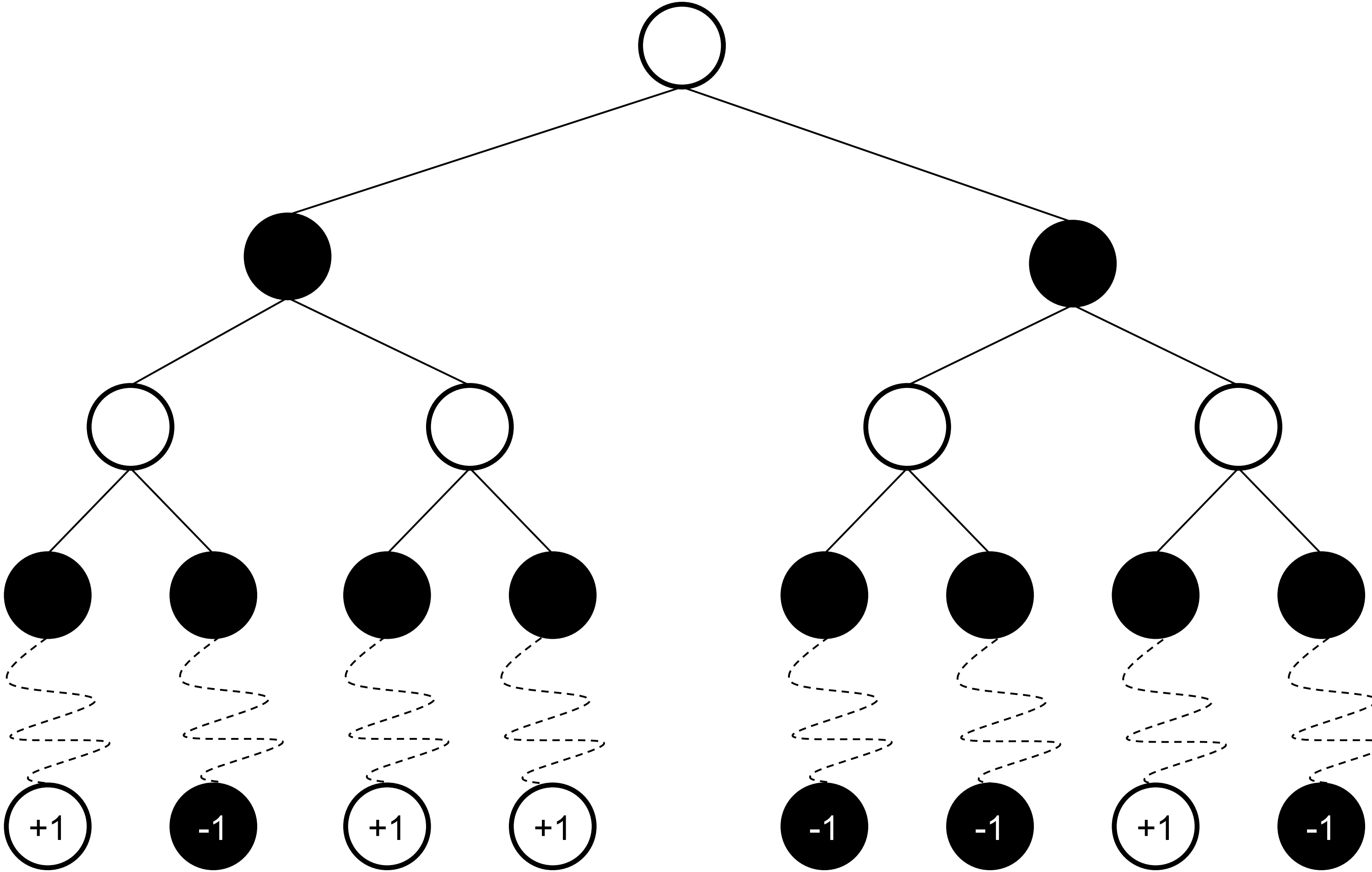


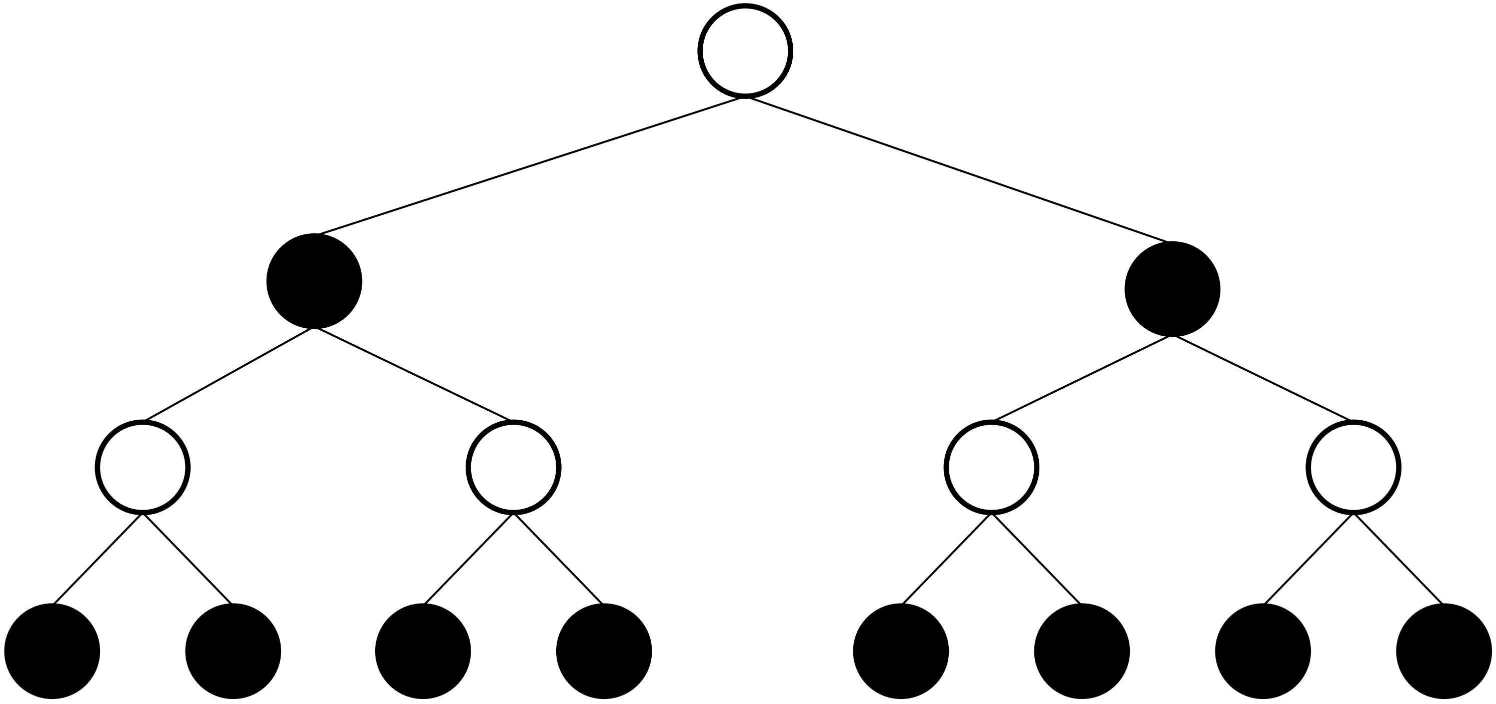
White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

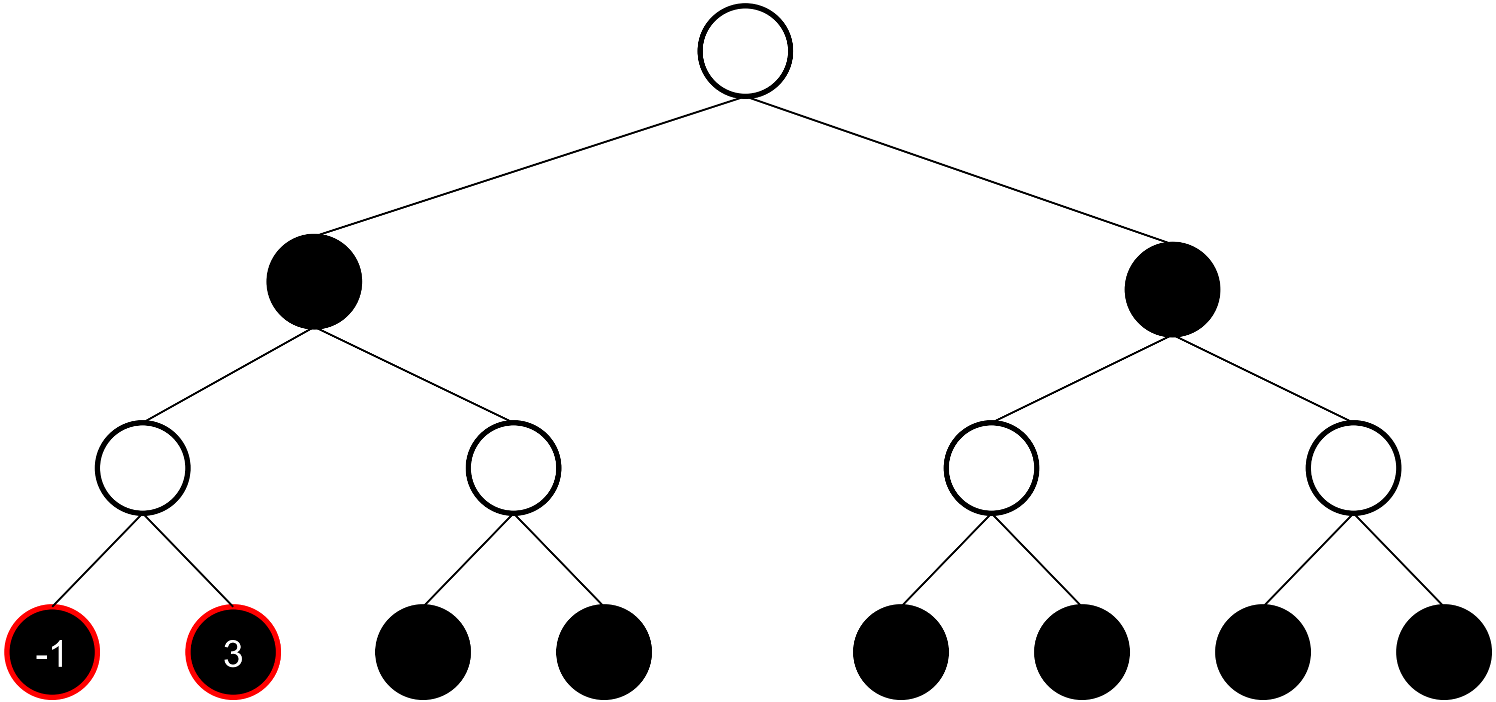
Black: 1x Rook (5 points) + 1x Pawn (1 point) = 6 points

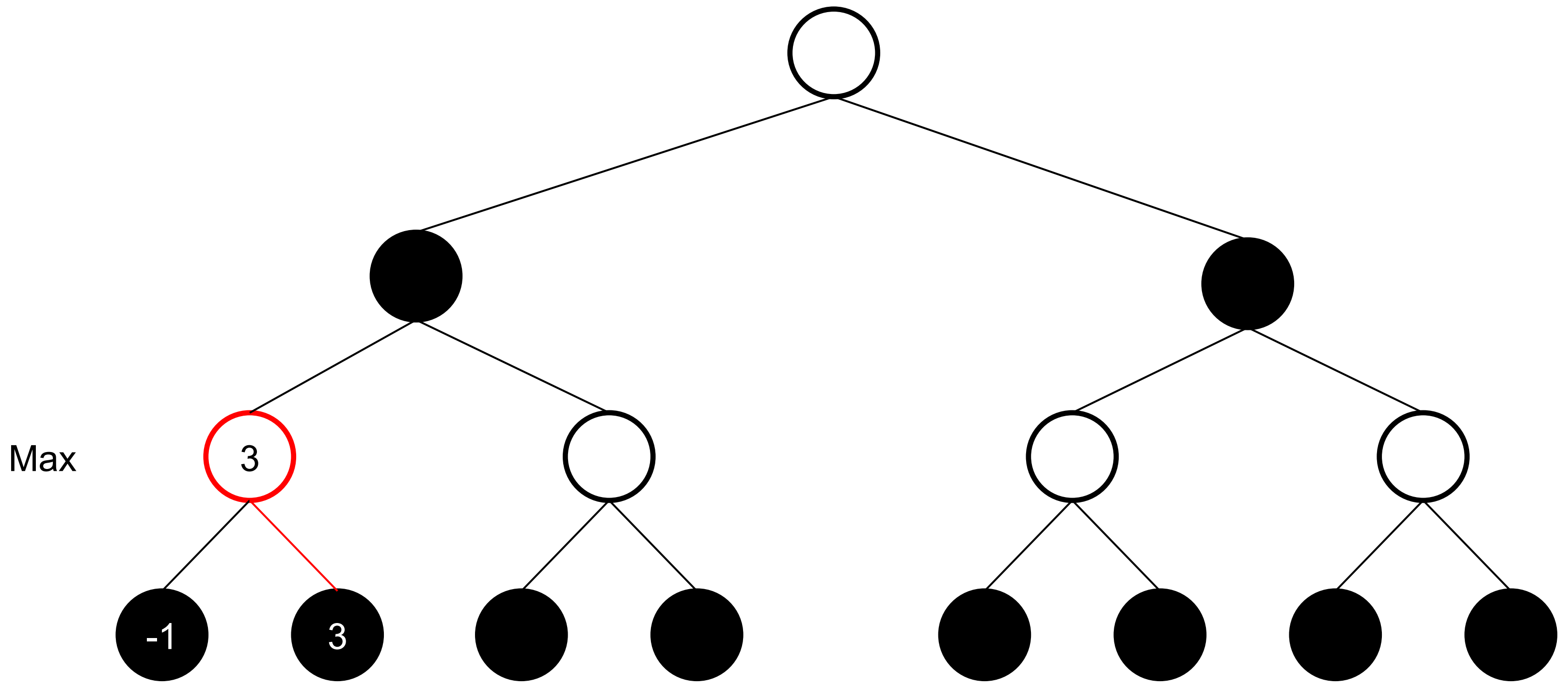
Eval: 11 - 6 = 5 points

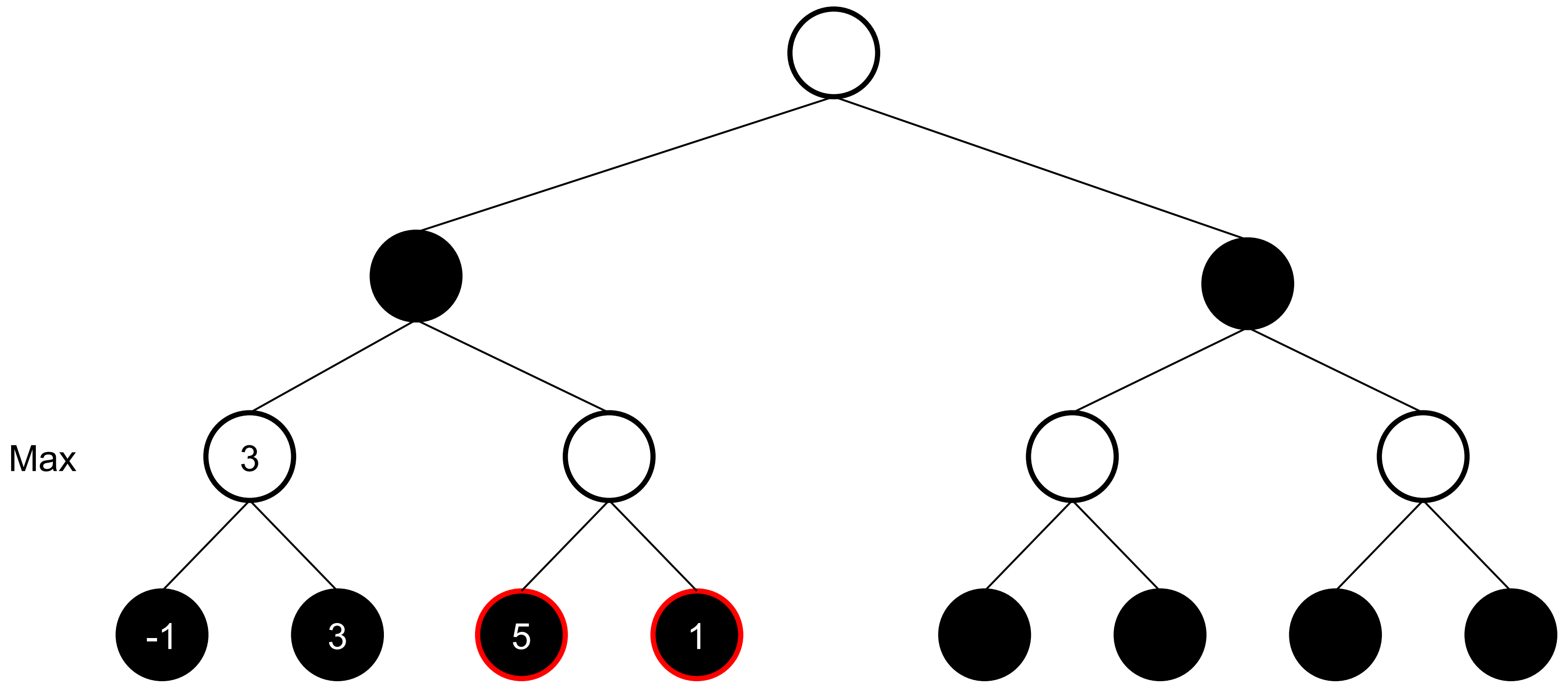



+1
-1
+1
+1
-1
-1
+1
-1

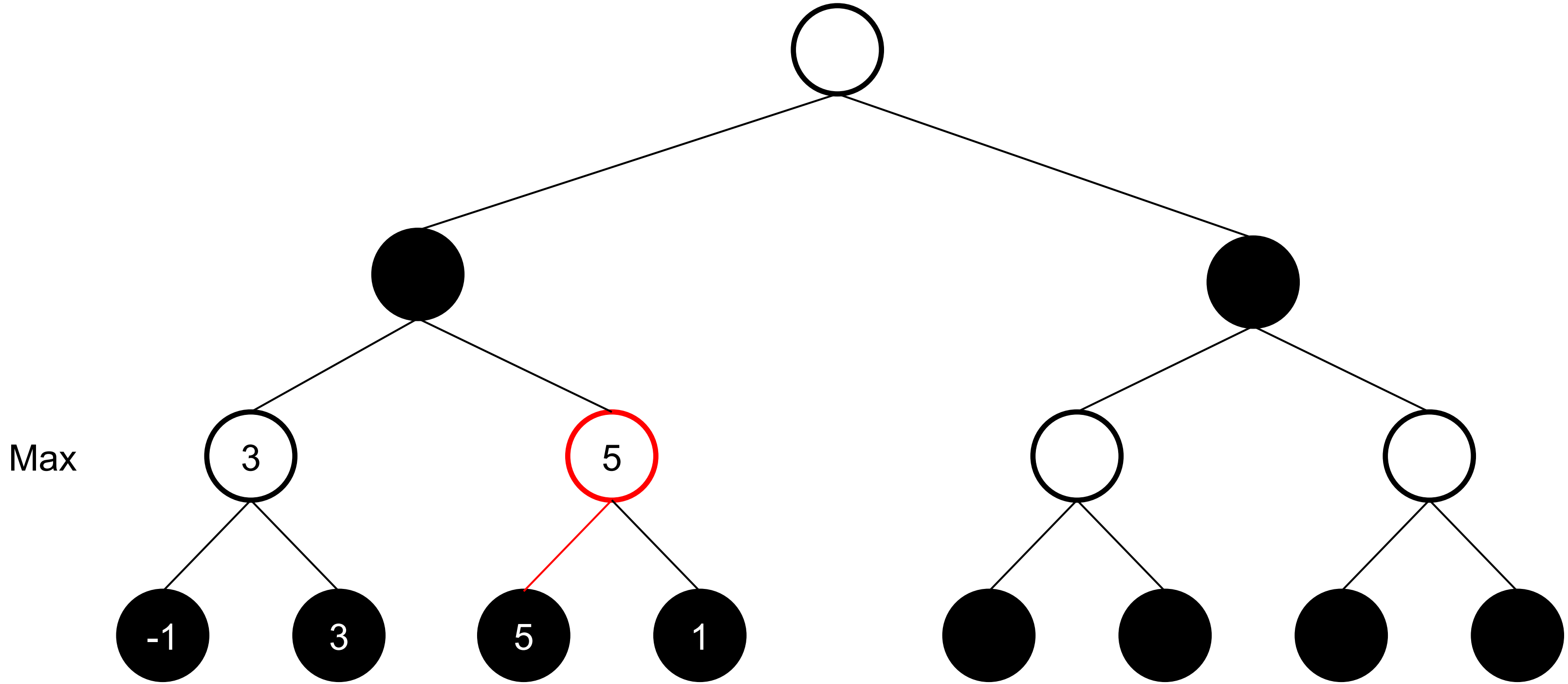


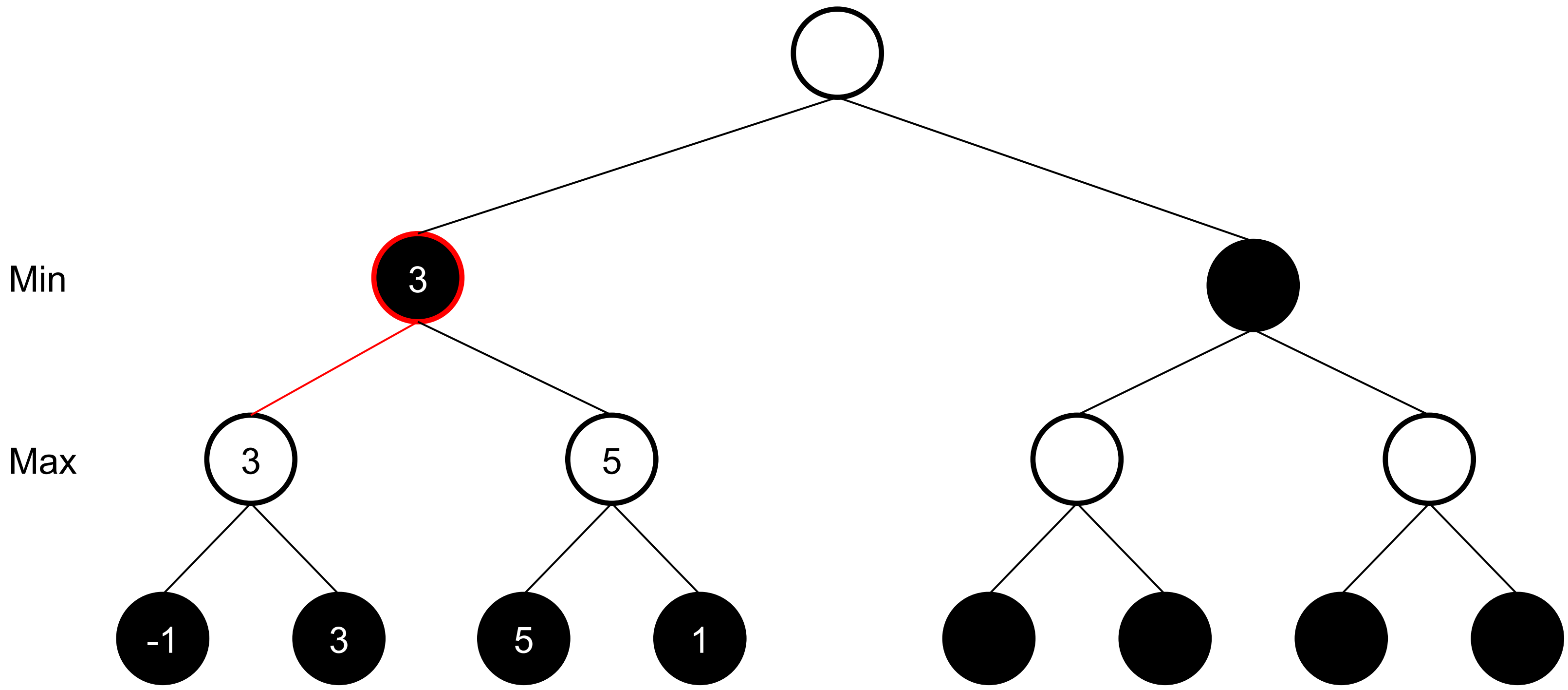

-1
3


Max
3
-1
3

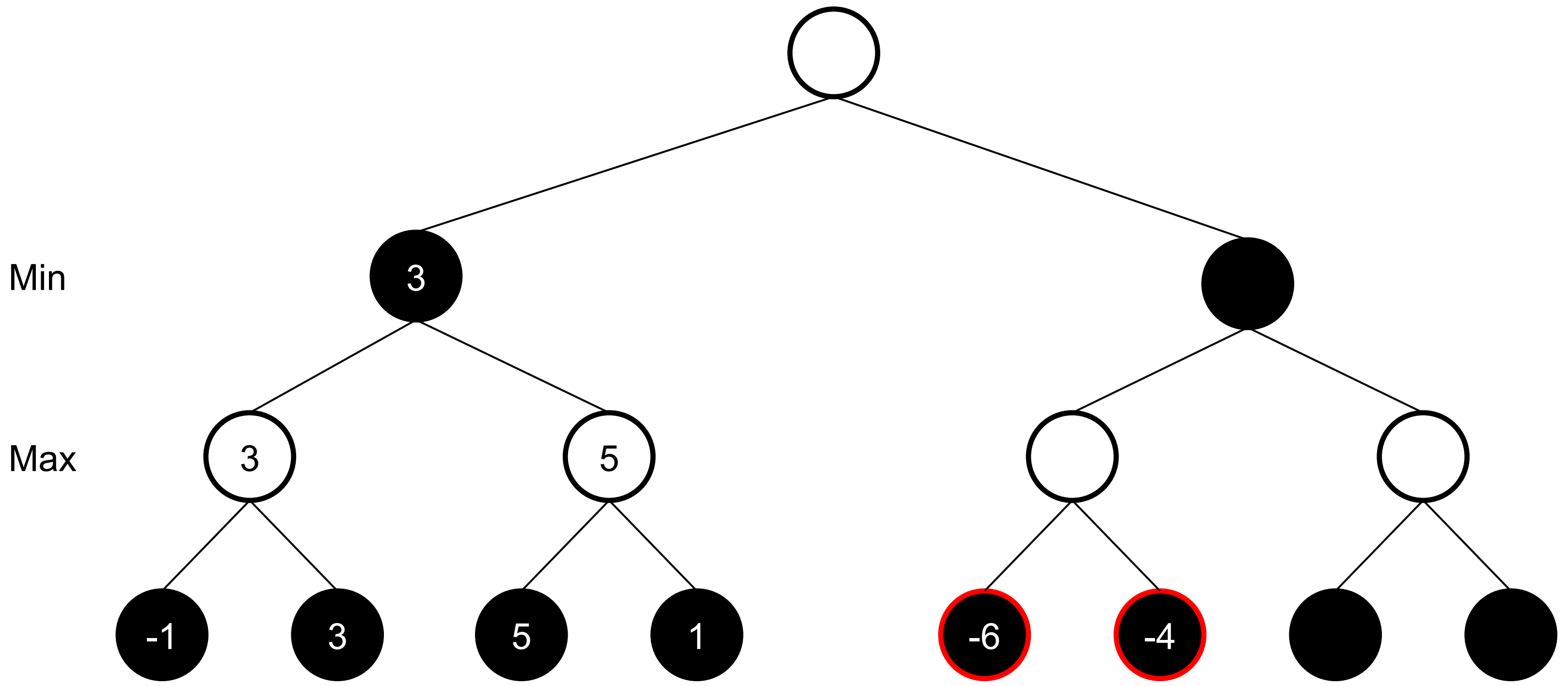


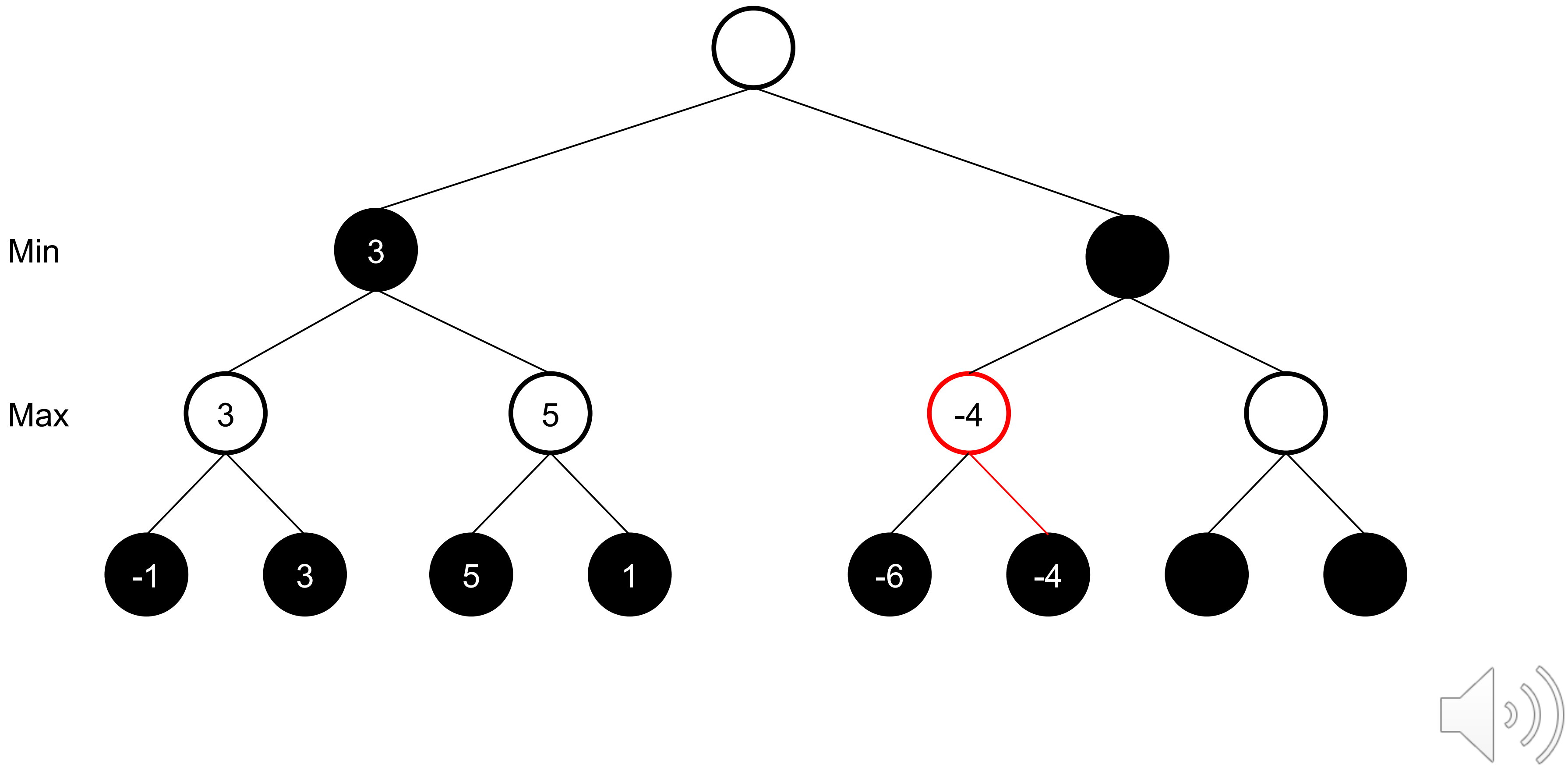
Max
3
-1
3
5
1


Max
3
5
-1
3
5
1

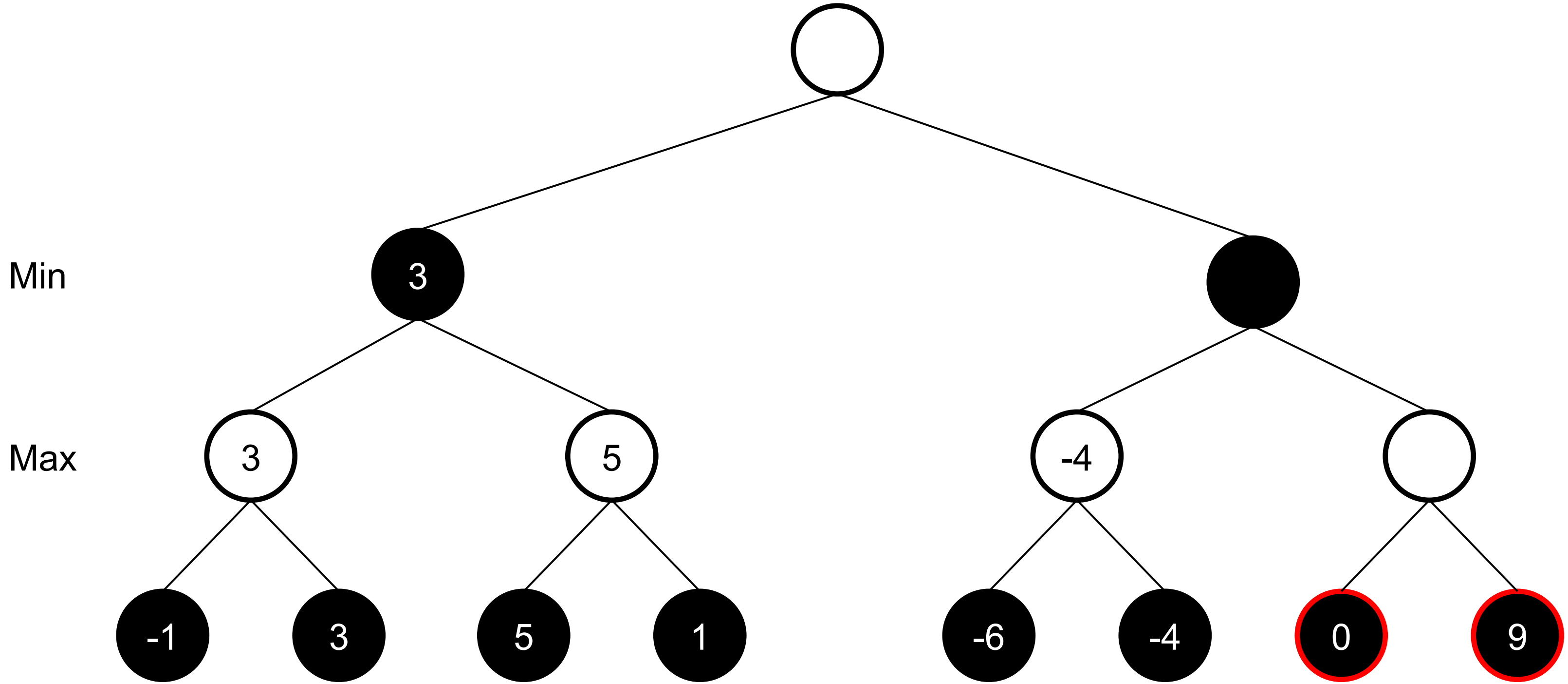


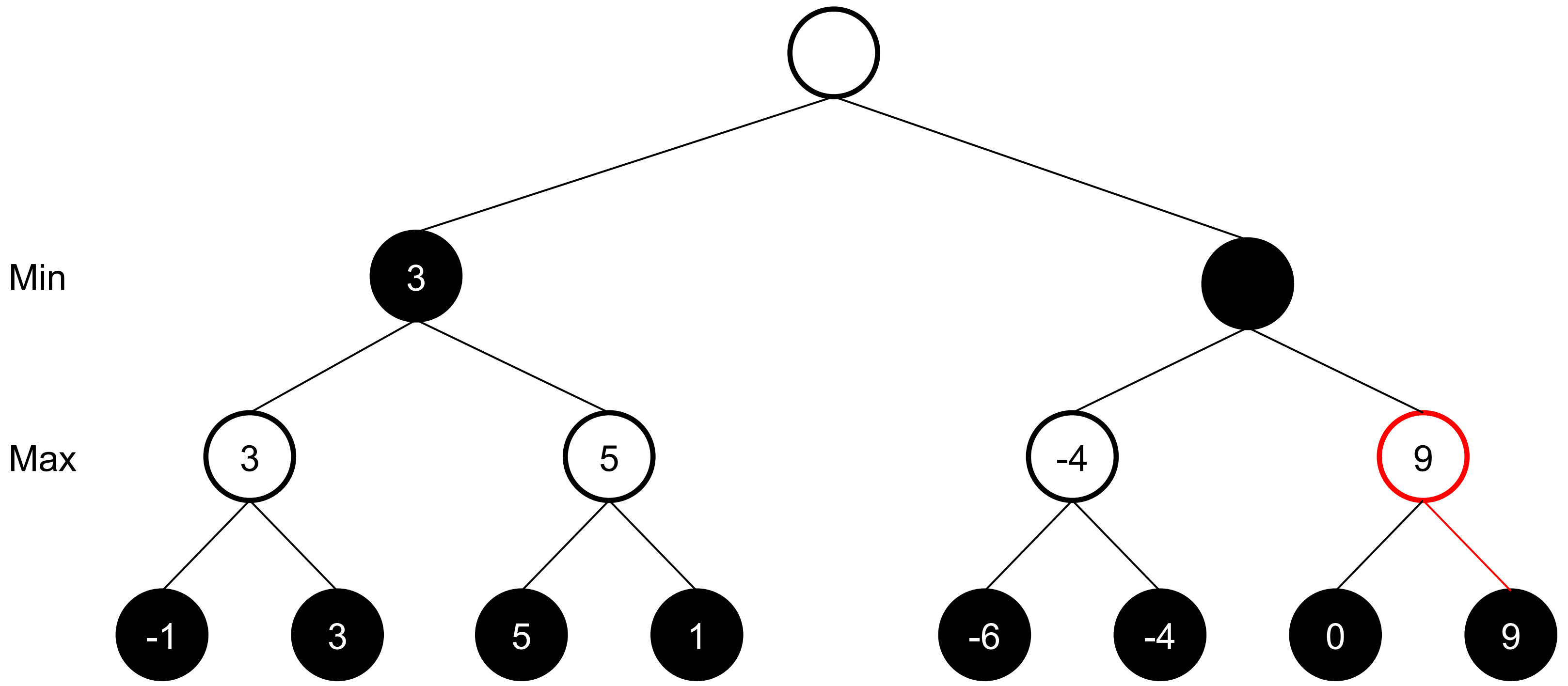
Min
Max
3
3
5
-1
3
5
1

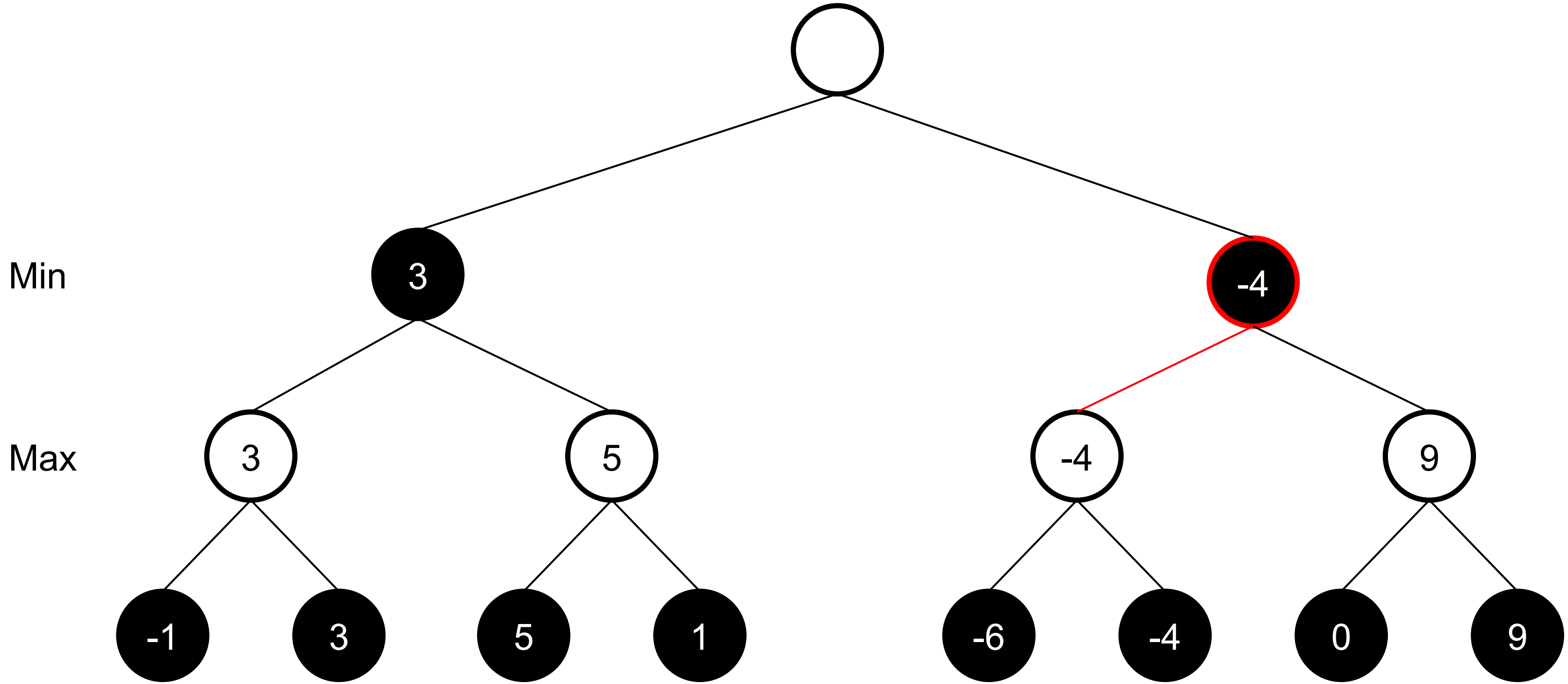

Min
Max
3
3
5
-1
3
5
1
-6
-4



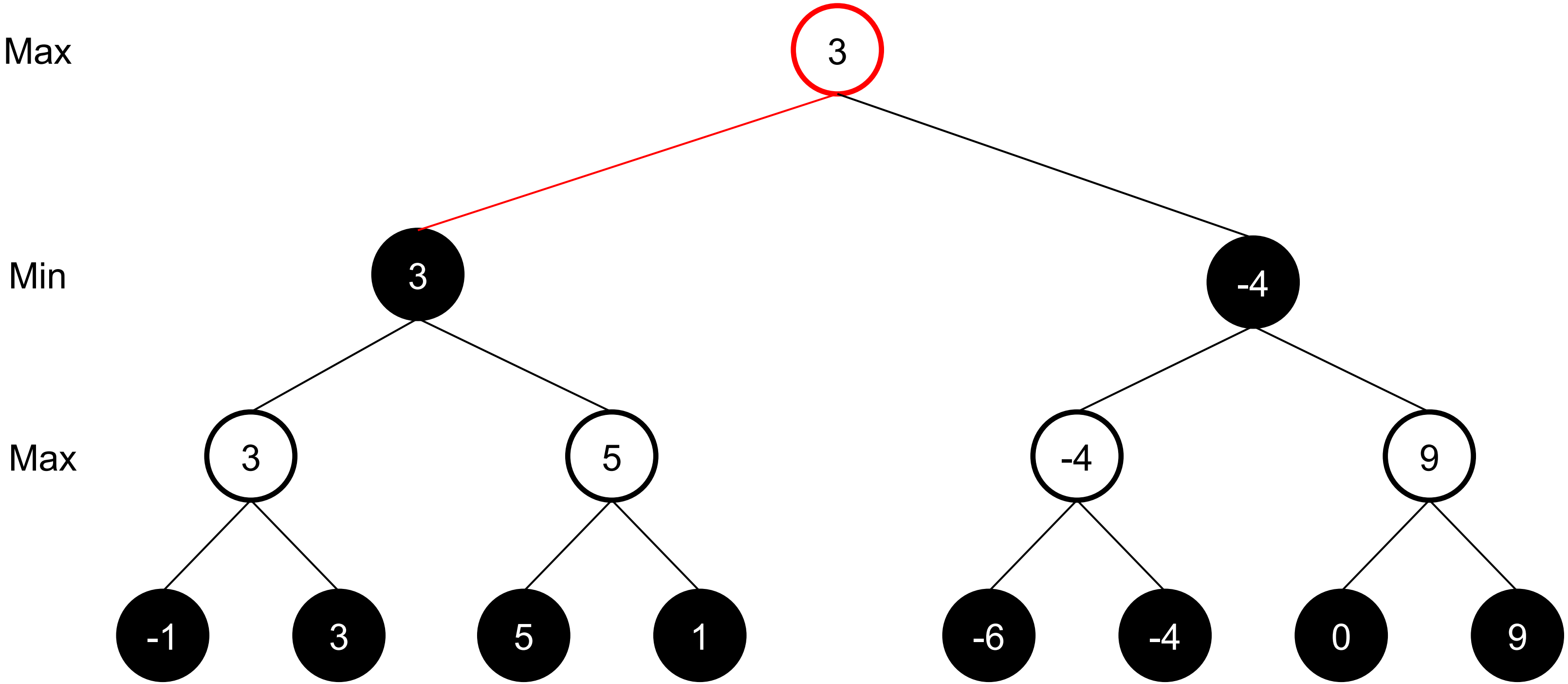
Min
Max
3
3
5
-4
-1
3
5
1
-6
-4

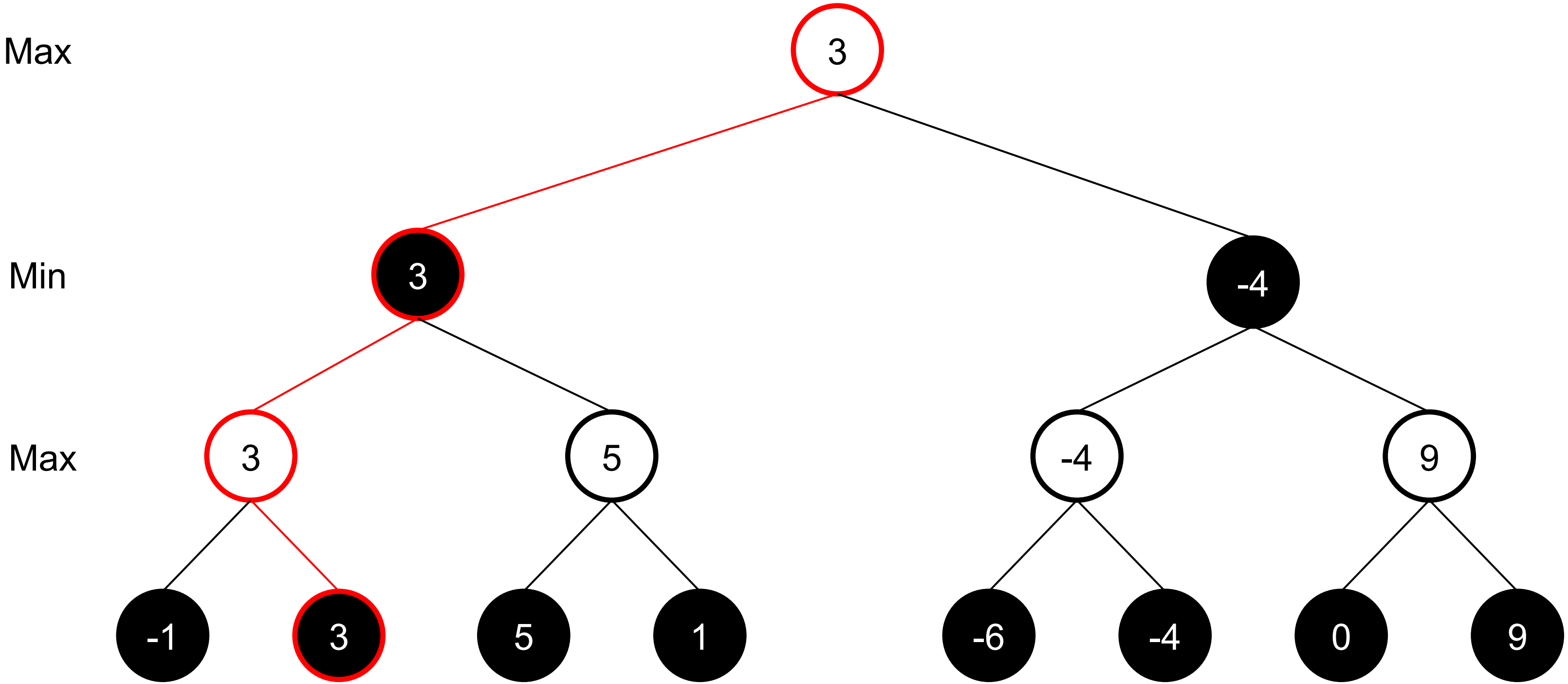

Min
Max
3
3
5
-4
-1
3
5
1
-6
-4
0
9


Min
Max
3
3
5
-4
9
-1
3
5
1
-6
-4
0
9

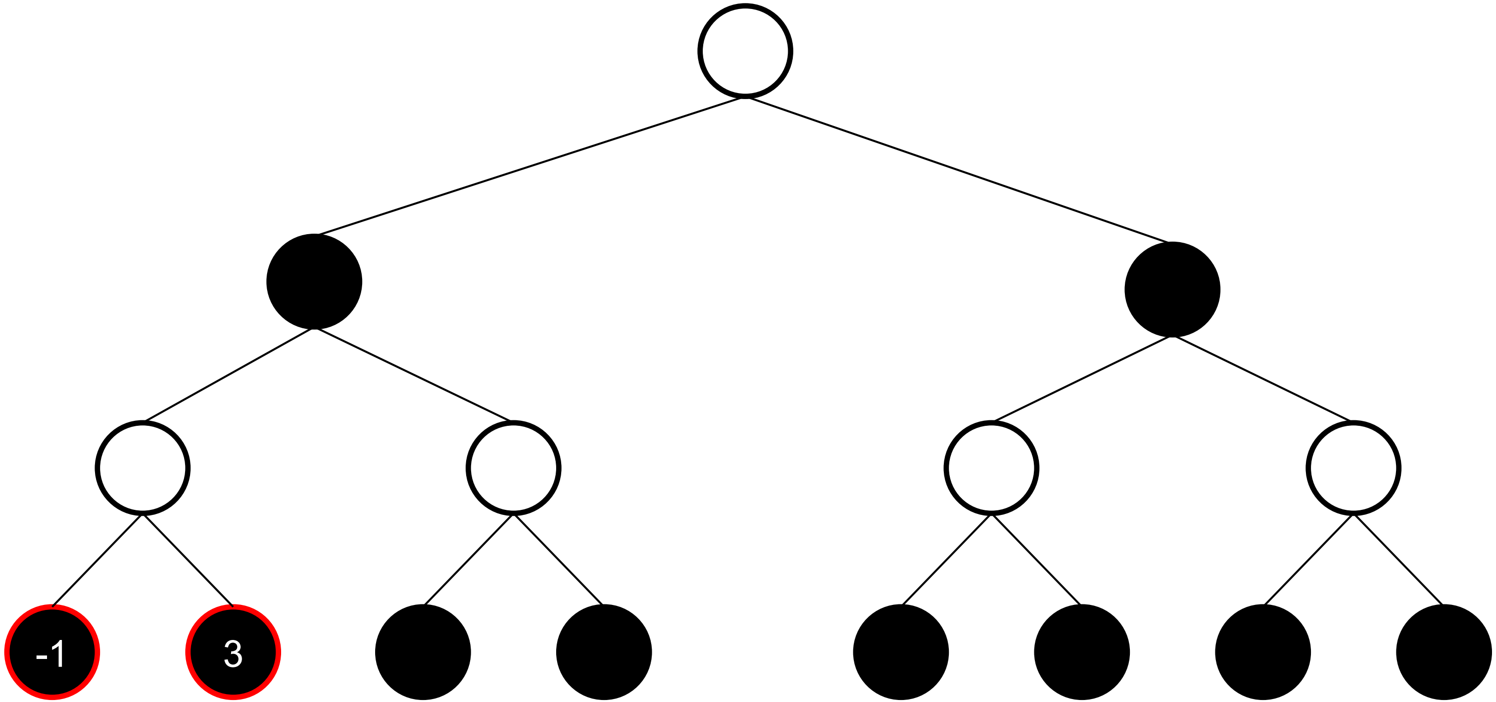


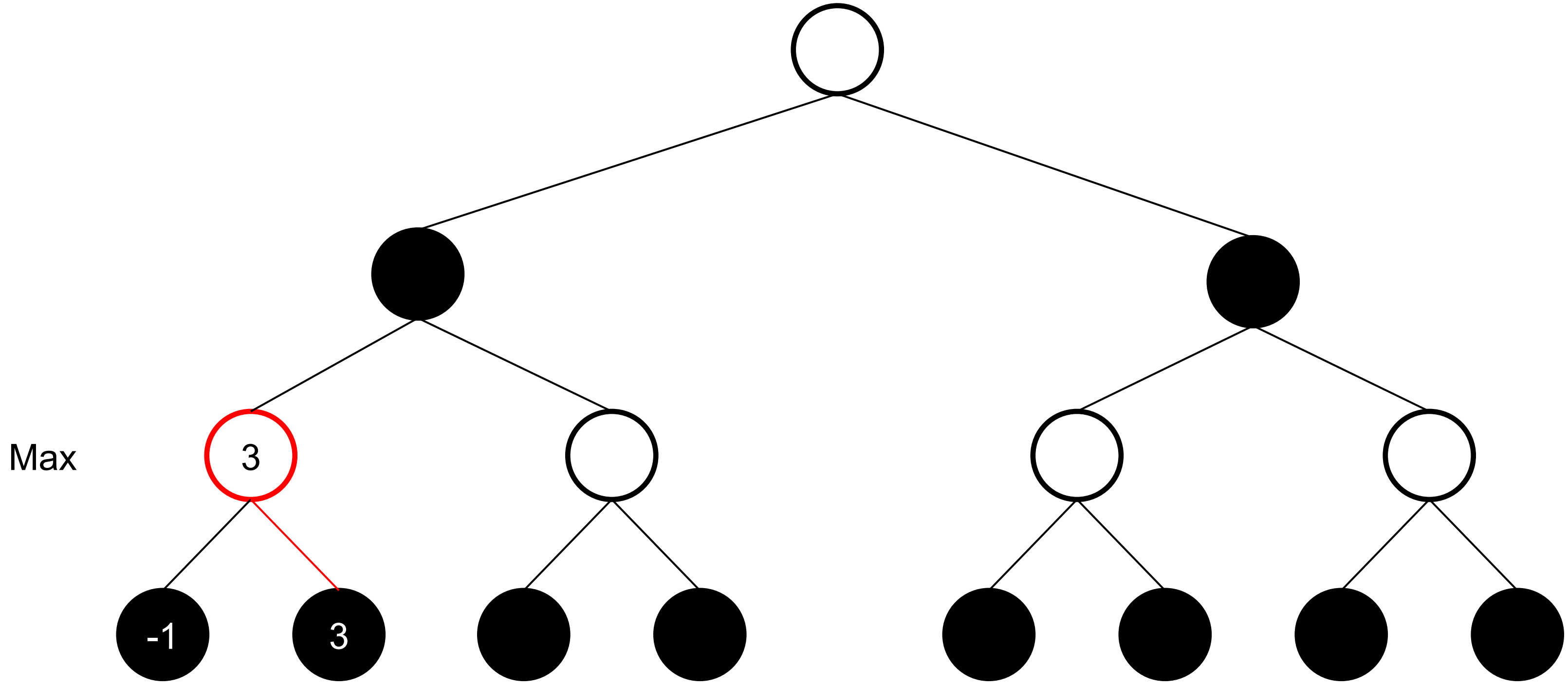
Min
Max
3
-4
3
5
-4
9
-1
3
5
1
-6
-4
0
9

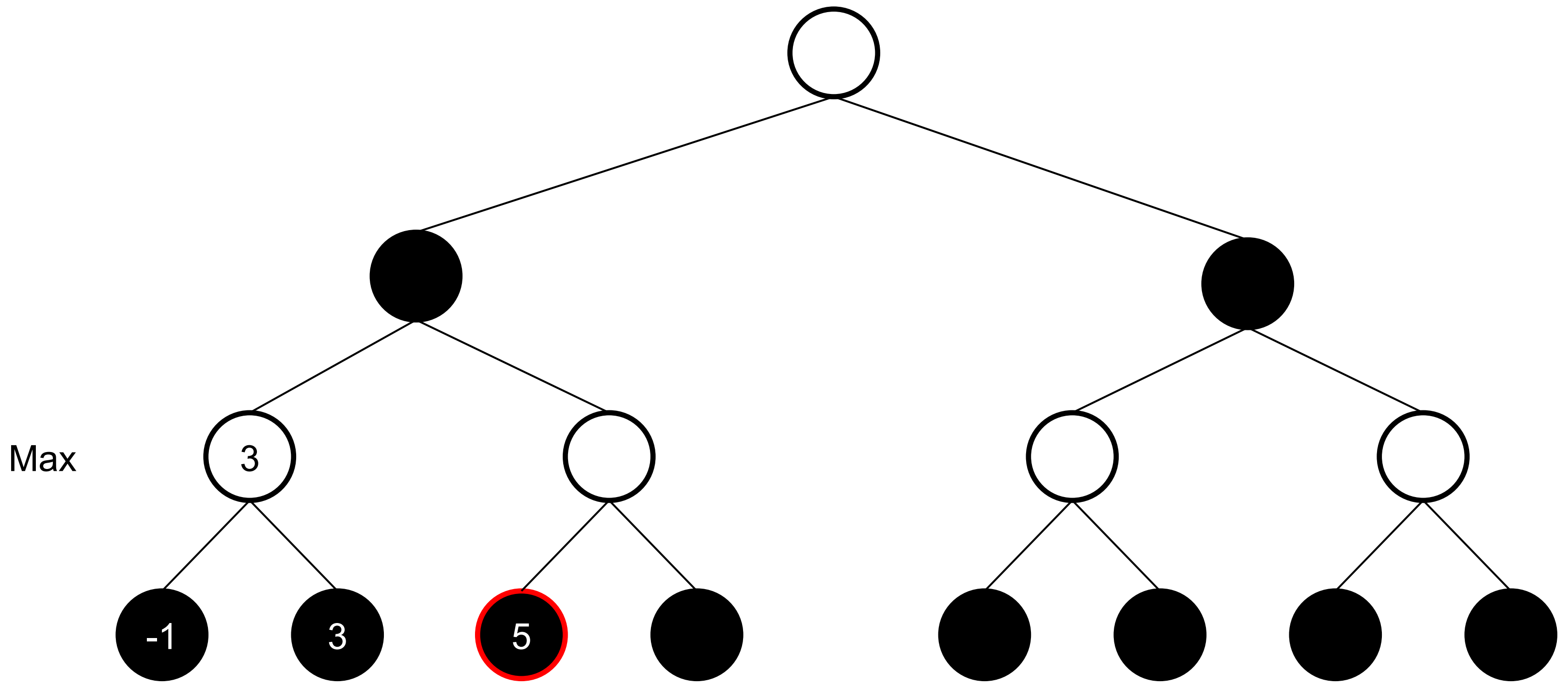

Max
Min
Max
3
3
-4
3
5
-4
9
-1
3
5
1
-6
-4
0
9

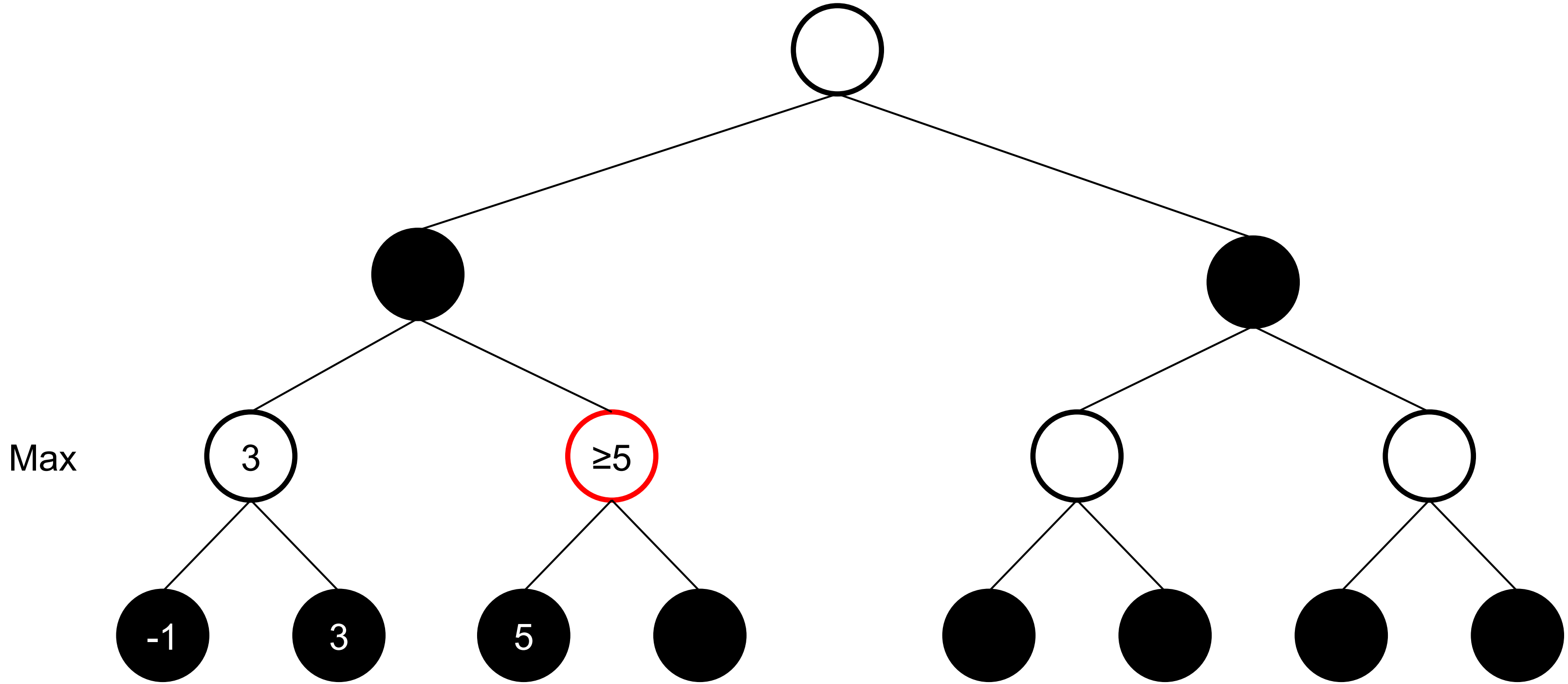


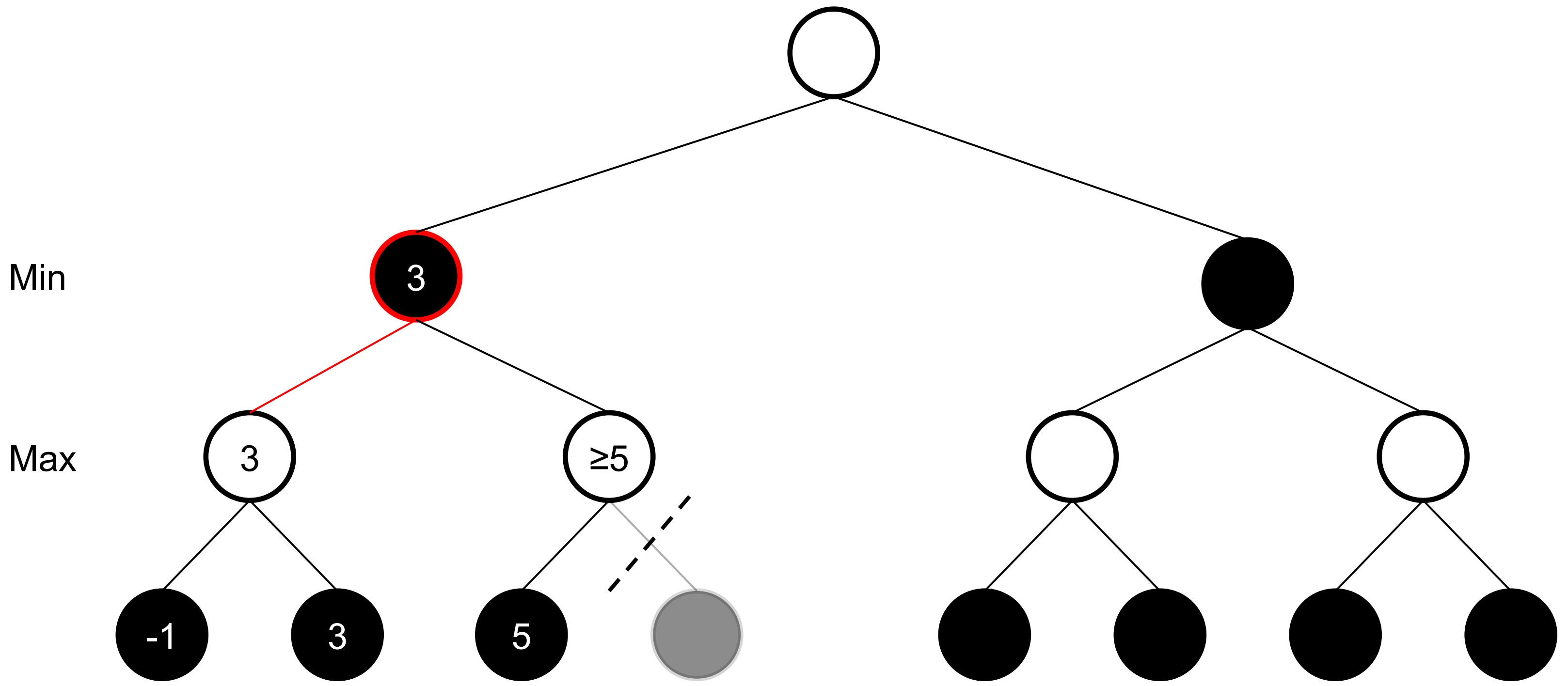
Max
Min
Max
3
3
-4
3
5
-4
9
-1
3
5
1
-6
-4
0
9


-1
3

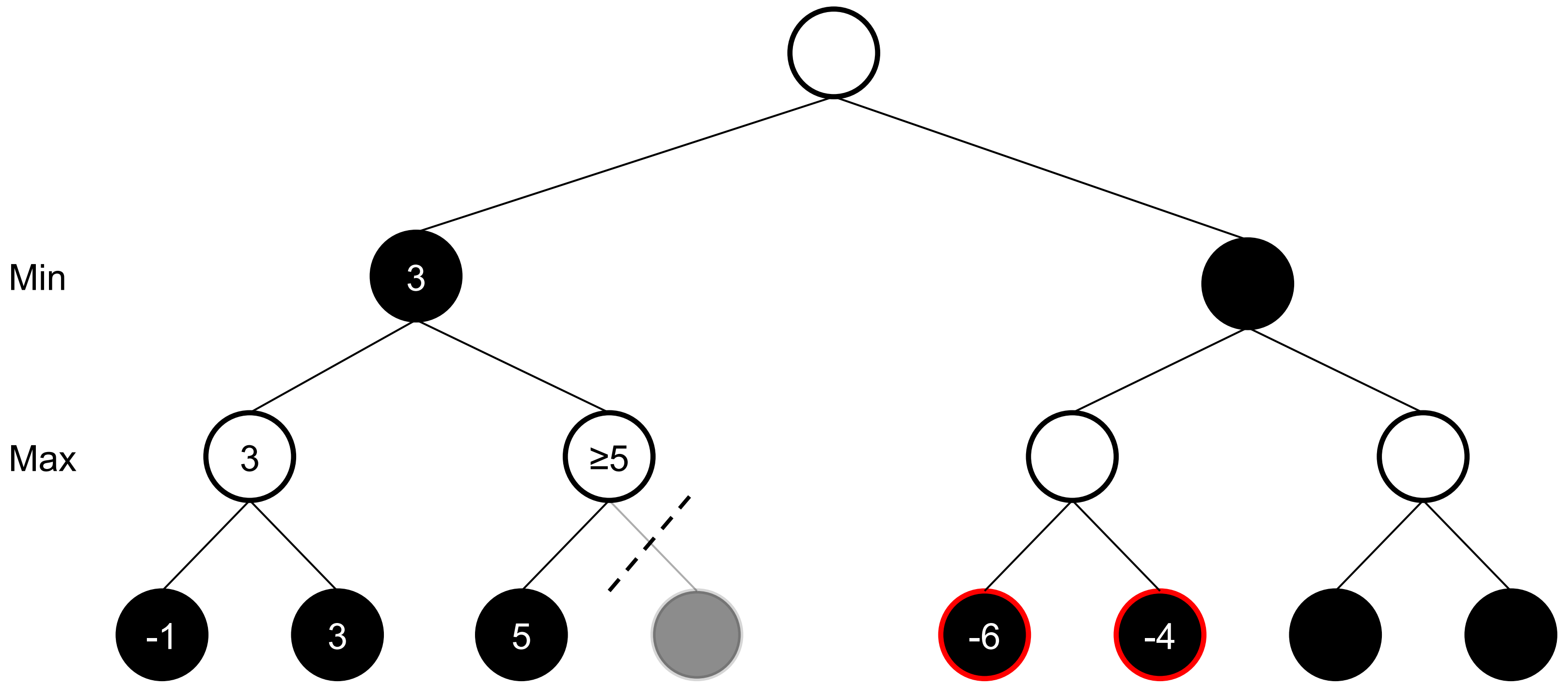

Max
3
-1
3

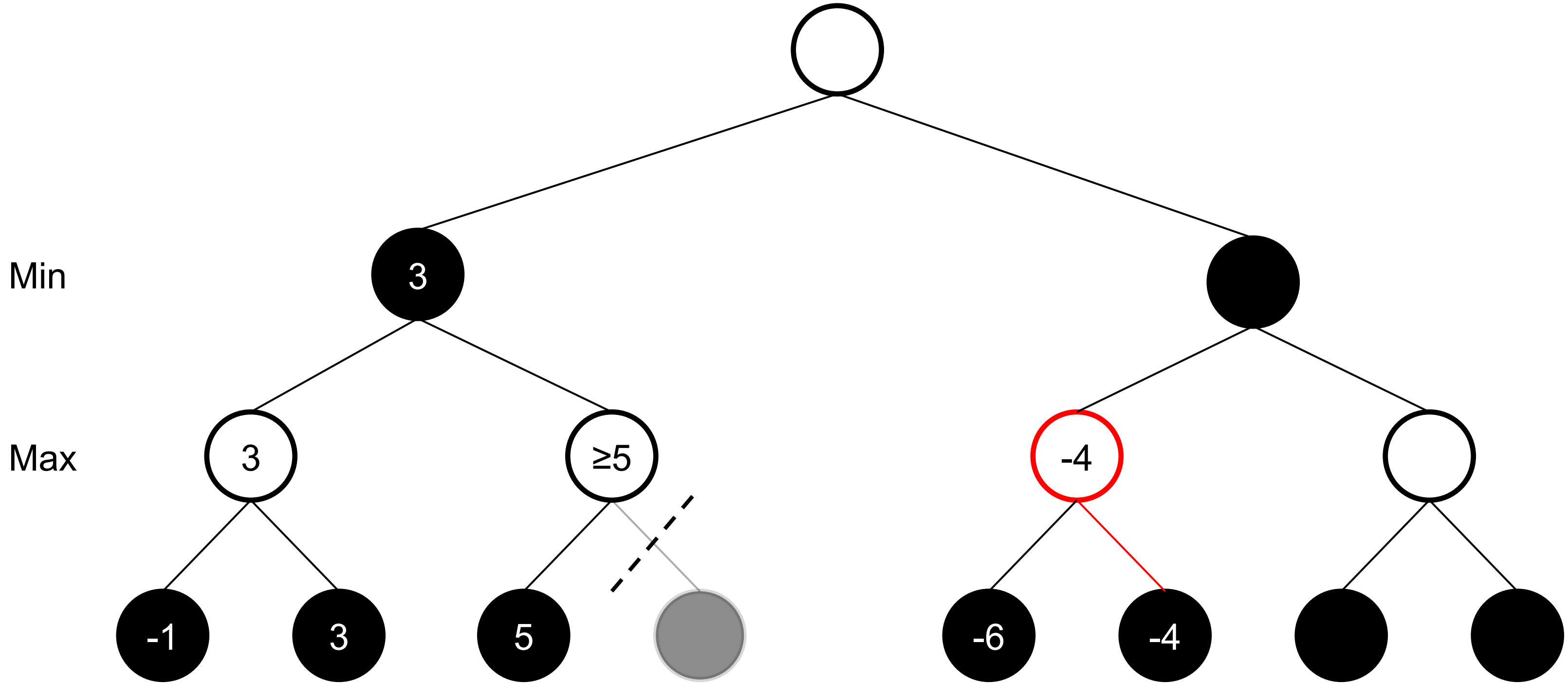

Max
3
-1
3
5

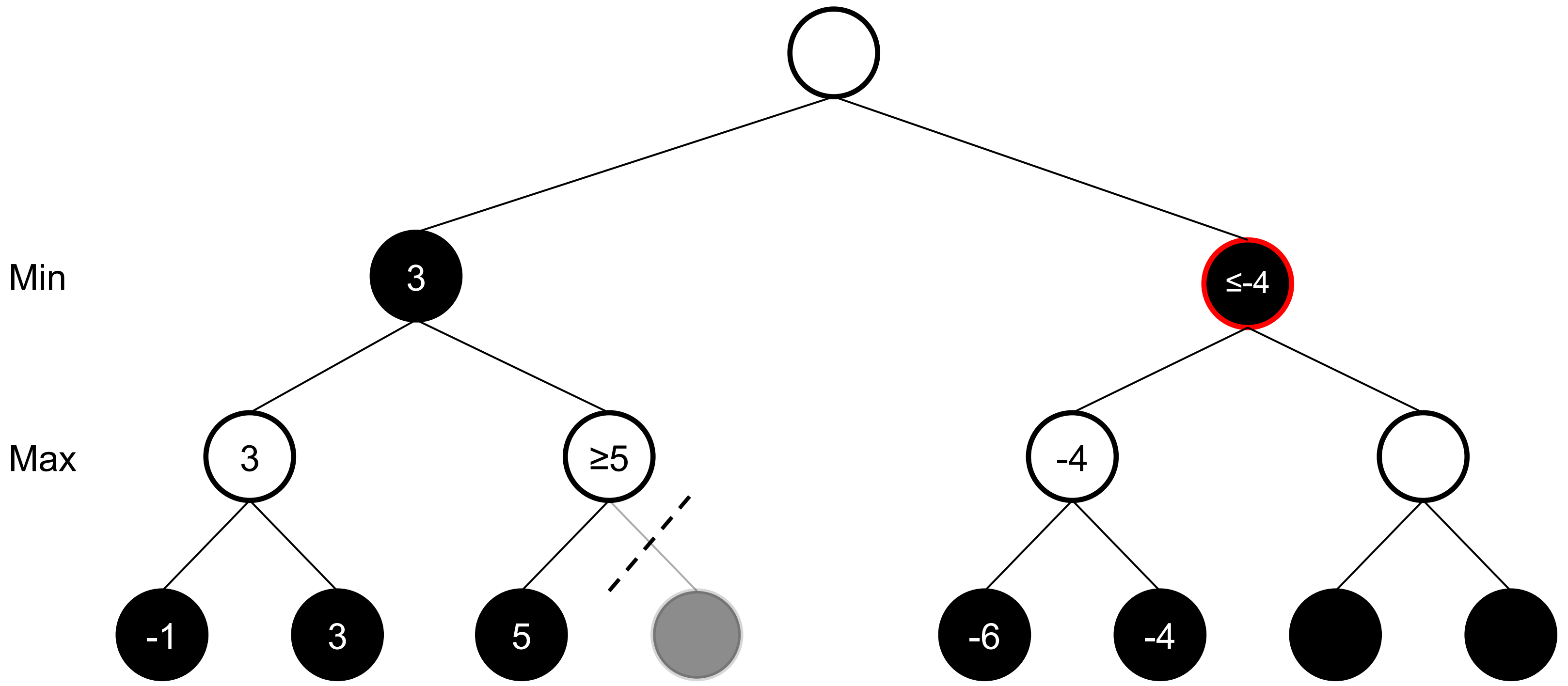

Max
3
≥5
-1
3
5



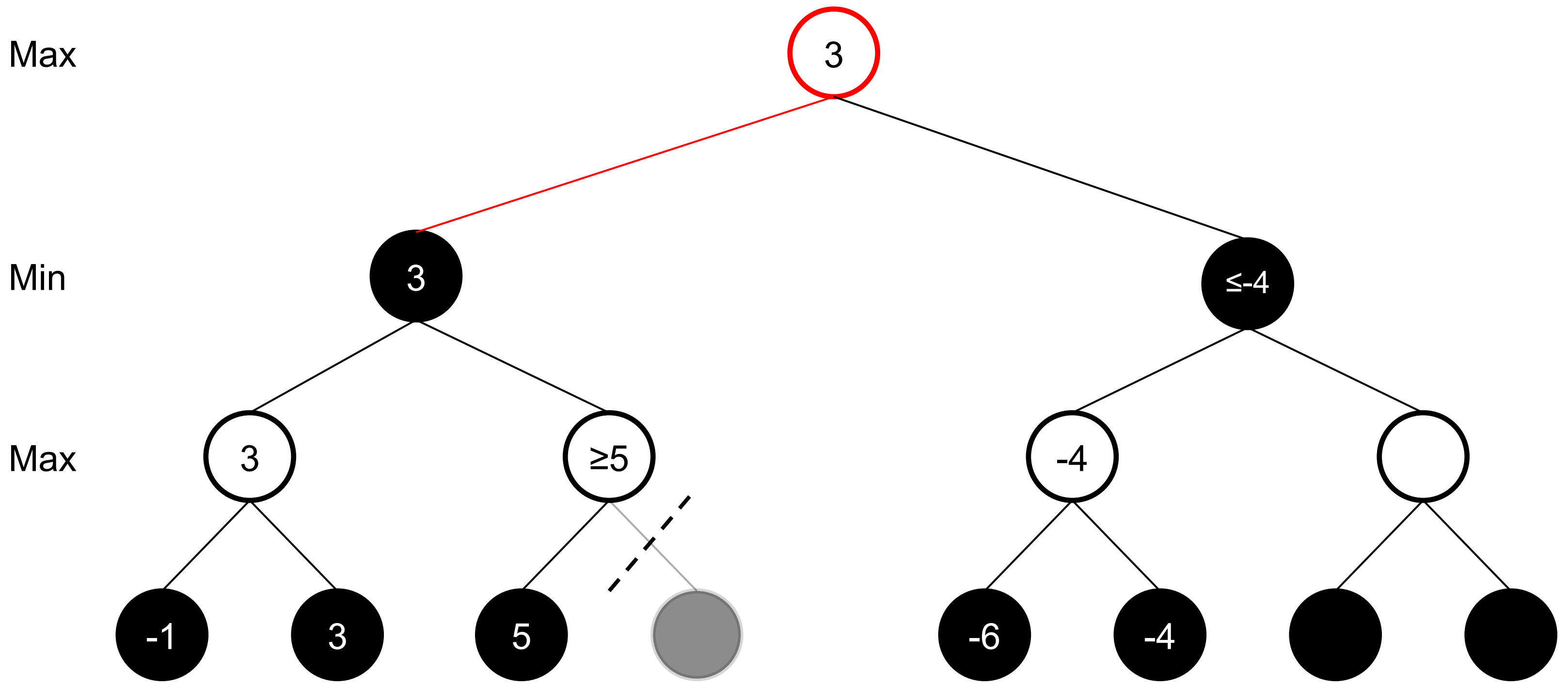
Min
Max
3
3
≥5
-1
3
5

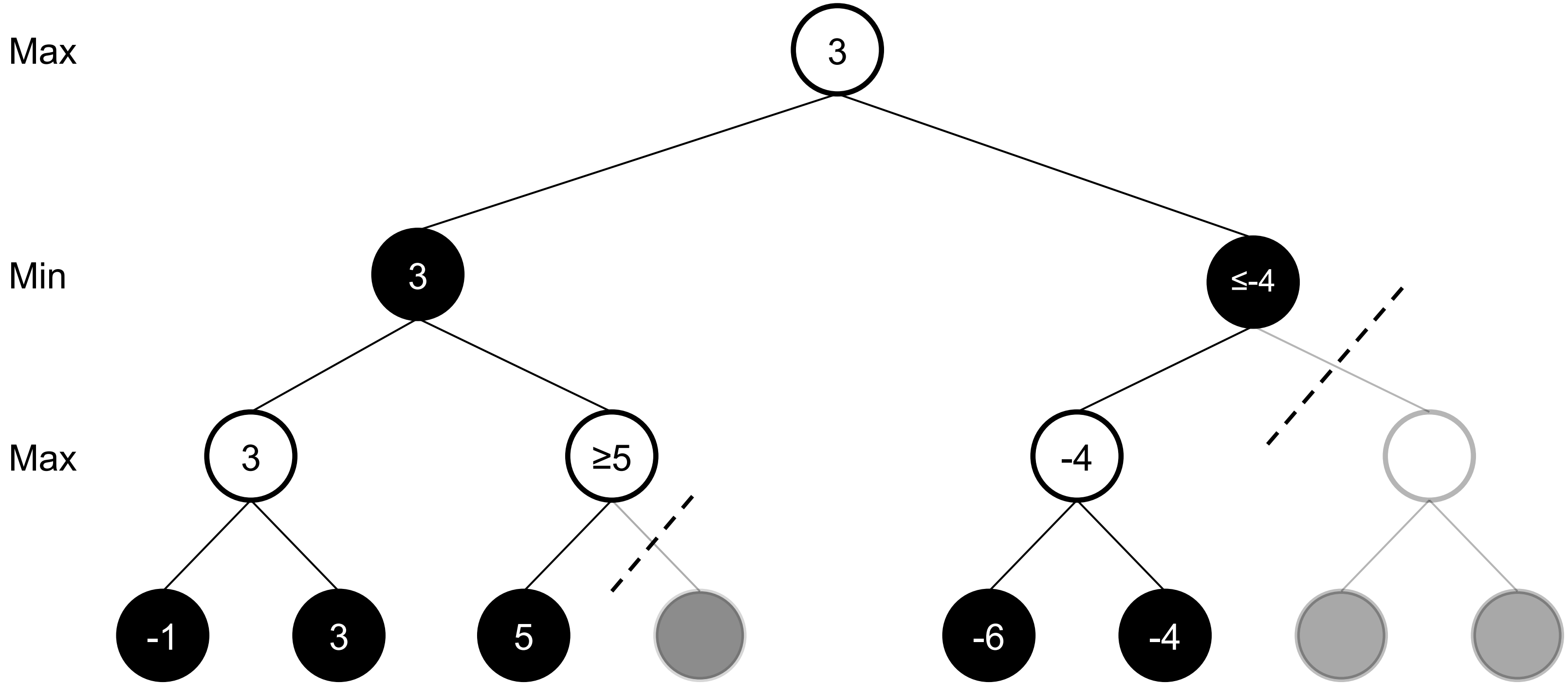

Min
Max
3
3
≥5
-1
3
5
-6
-4


Min
Max
3
3
≥5
-4
-1
3
5
-6
-4

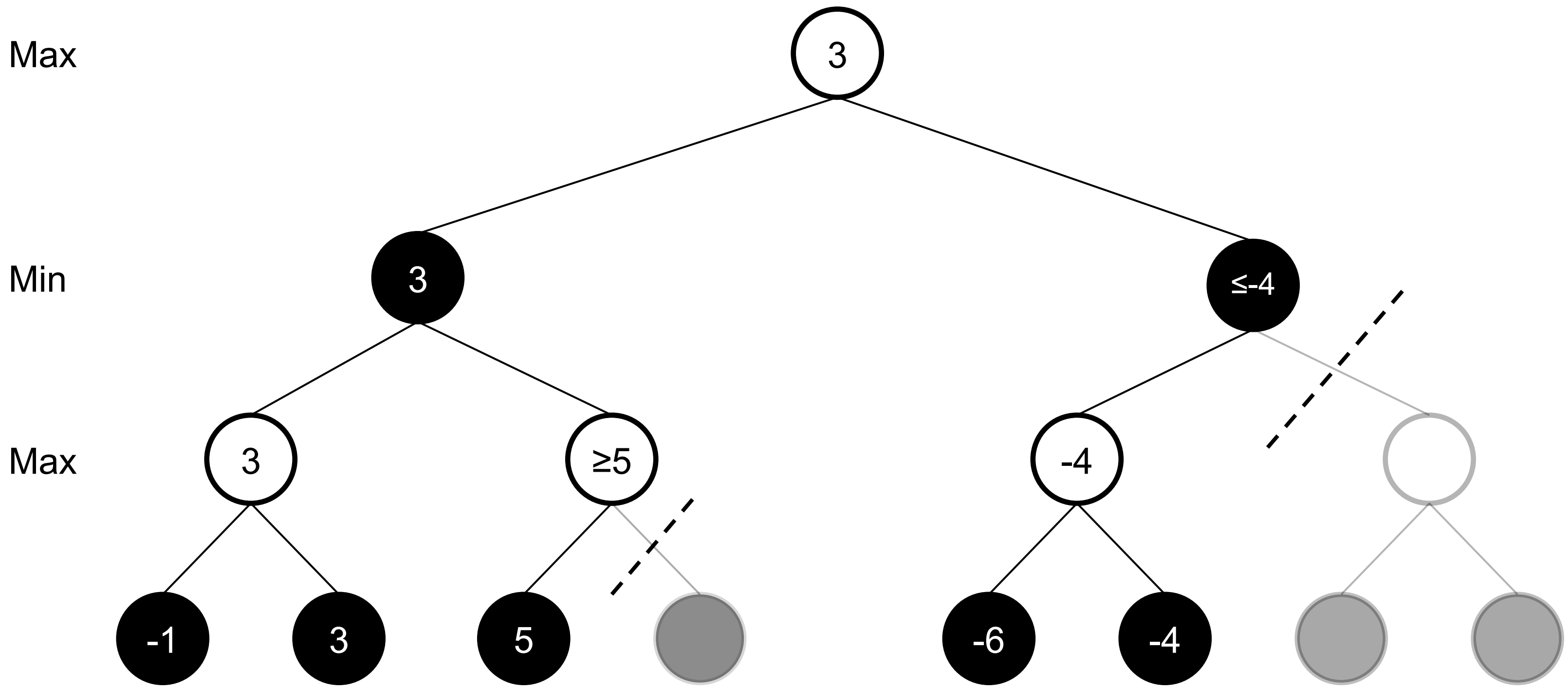


Min
Max
3
≤-4
3
≥5
-4
-1
3
5
-6
-4

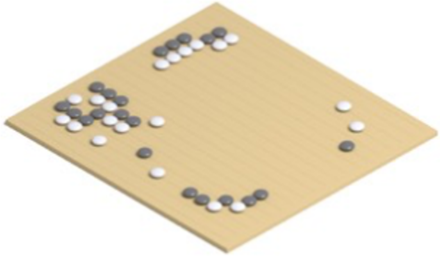

Max
Min
Max
3
3
≤-4
3
≥5
-4
-1
3
5
-6
-4



Max
Min
Max
3
3
≤-4
3
≥5
-4
-1
3
5
-6
-4

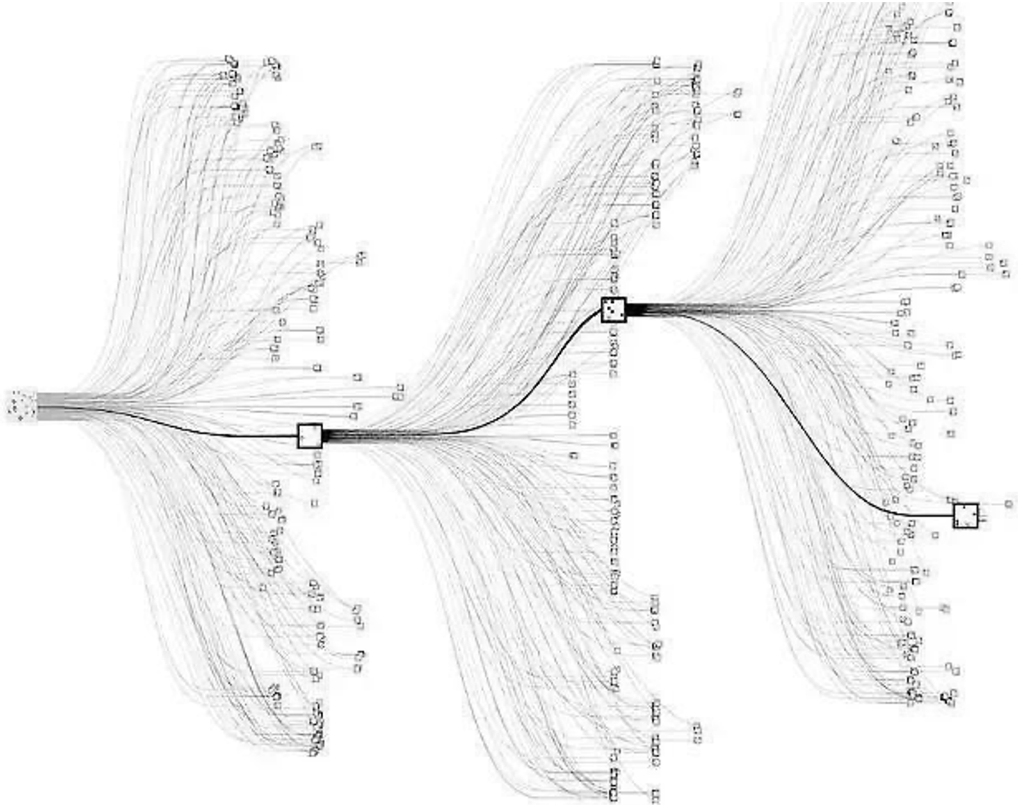

Max
Min
Max
3
3
≤-4
3
≥5
-4
-1
3
5
-6
-4

Go



- Branching Factor: 250
- Game Length: 150

Monte Carlo Tree Search



Monte Carlo Tree Search

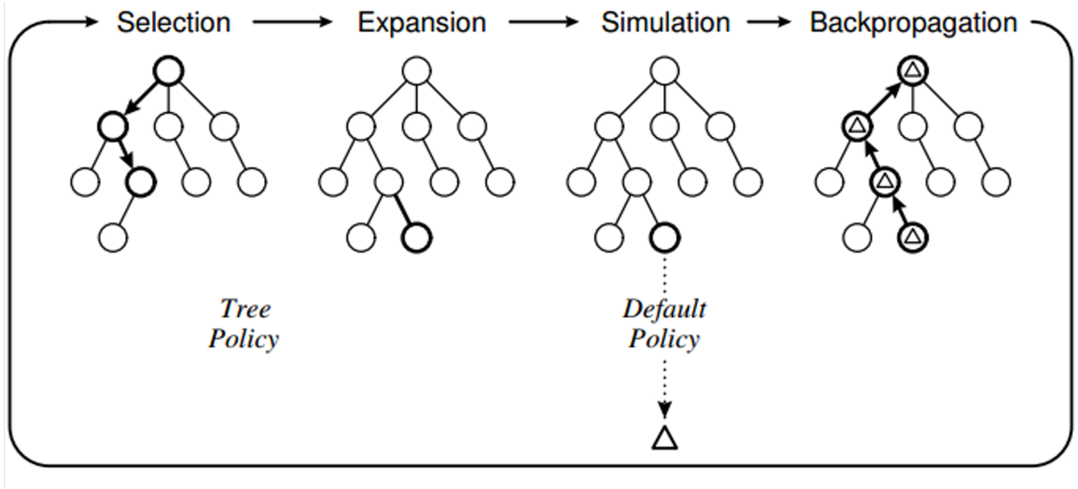
1. Selection (Tree Traversal)

$$UCB1(s_i) = \frac{w_i}{n_i} + C\sqrt{\frac{\ln N}{n_i}}$$

2. Expansion

3. Simulation (Rollout)

4. Backpropagation



Monte Carlo Tree Search

Exploitation vs. Exploration

- Exploit promising actions
- Explore little known actions

Monte Carlo Tree Search

Exploitation vs. Exploration

$$UCB1(s_i) = \frac{w_i}{n_i} + C\sqrt{\frac{\ln N}{n_i}}$$

- Exploit promising actions
- Explore little known actions

Monte Carlo Tree Search

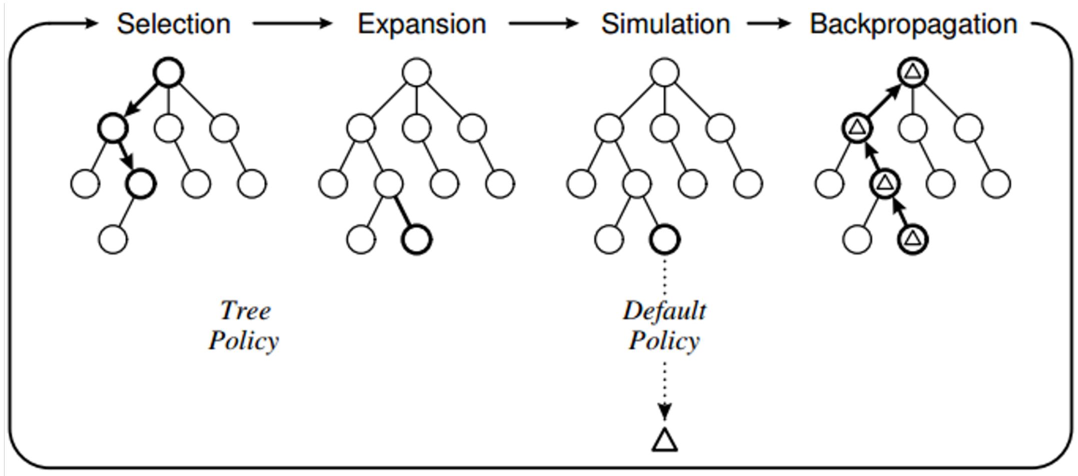
1. Selection (Tree Traversal)

$$UCB1(s_i) = \frac{w_i}{n_i} + C\sqrt{\frac{\ln N}{n_i}}$$

2. Expansion

3. Simulation (Rollout)

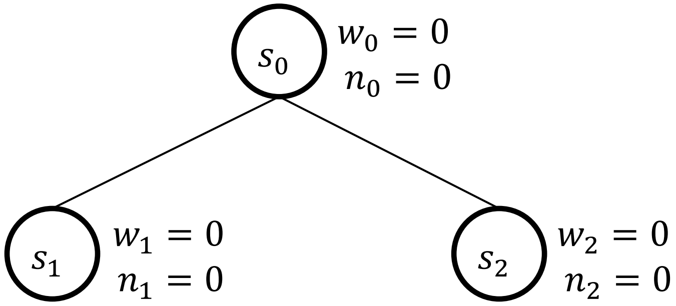
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

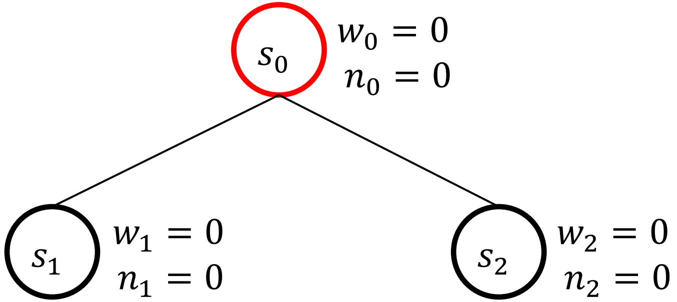
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

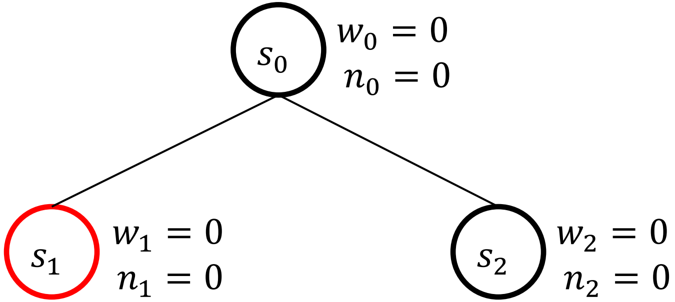
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

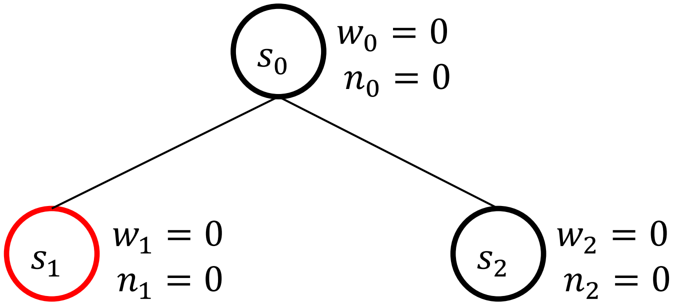
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

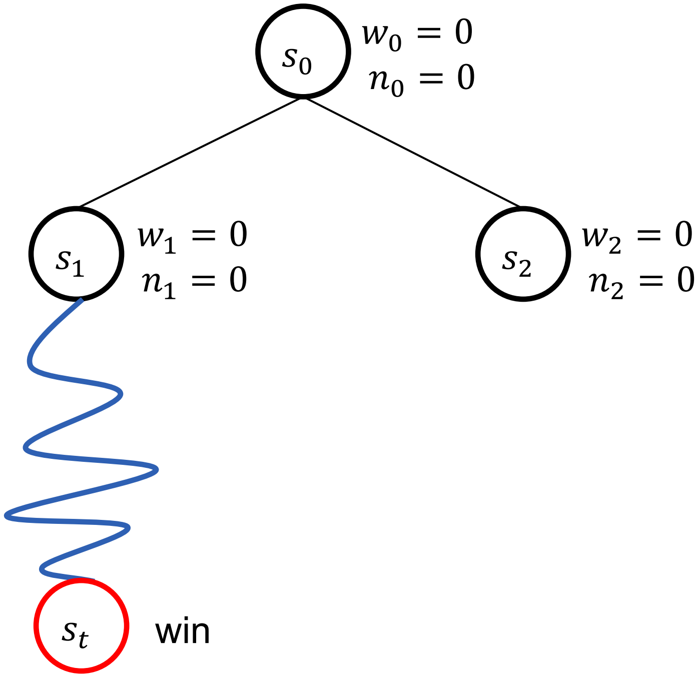
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

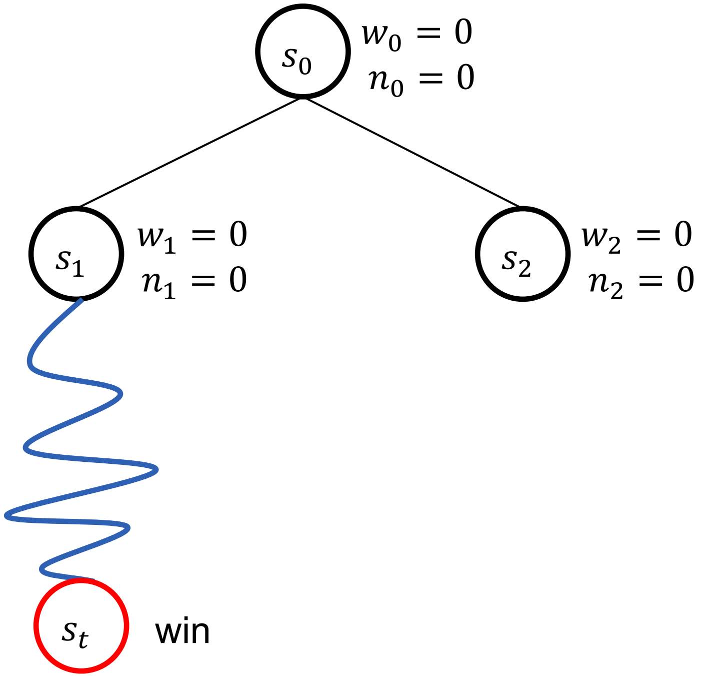
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

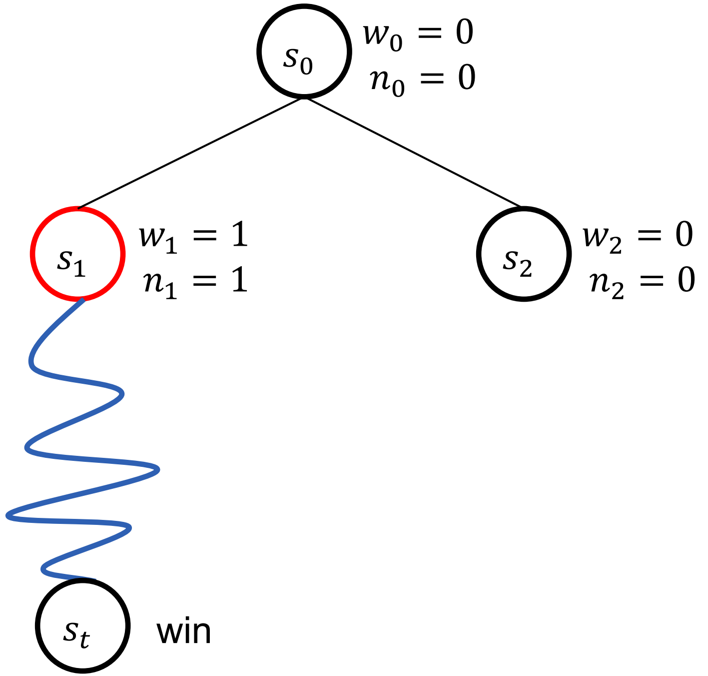
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

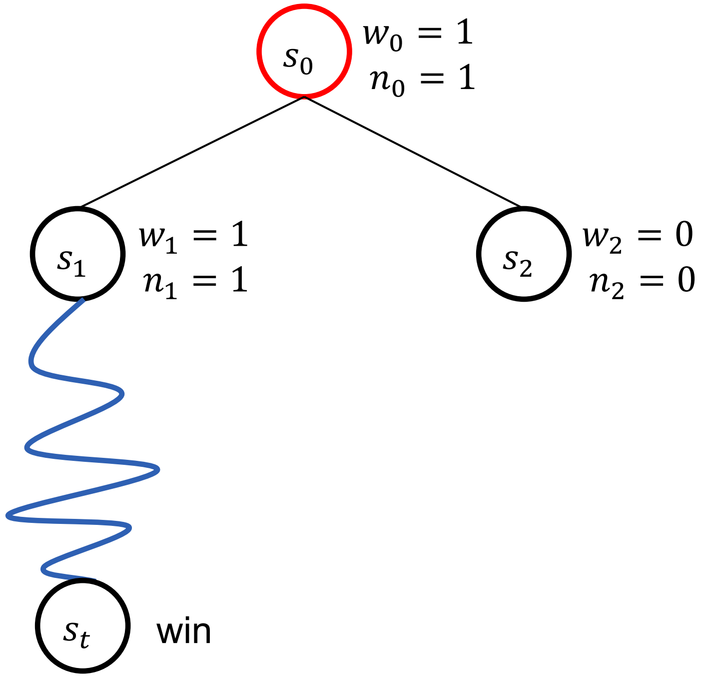
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

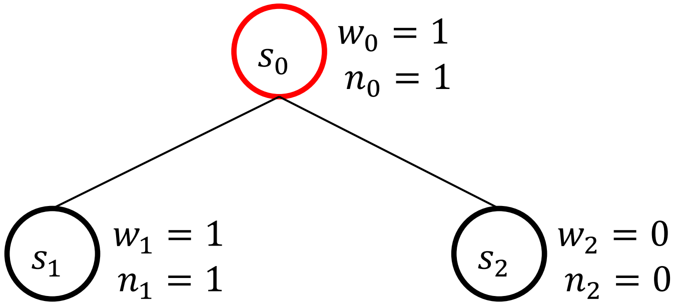
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

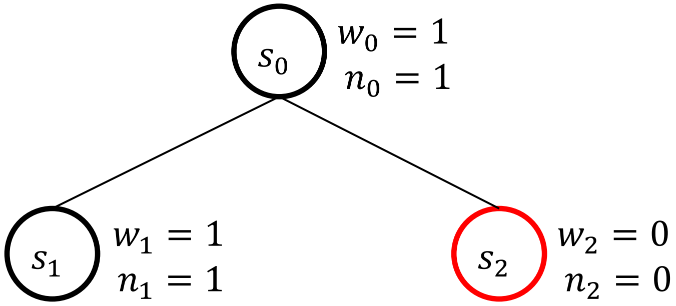
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

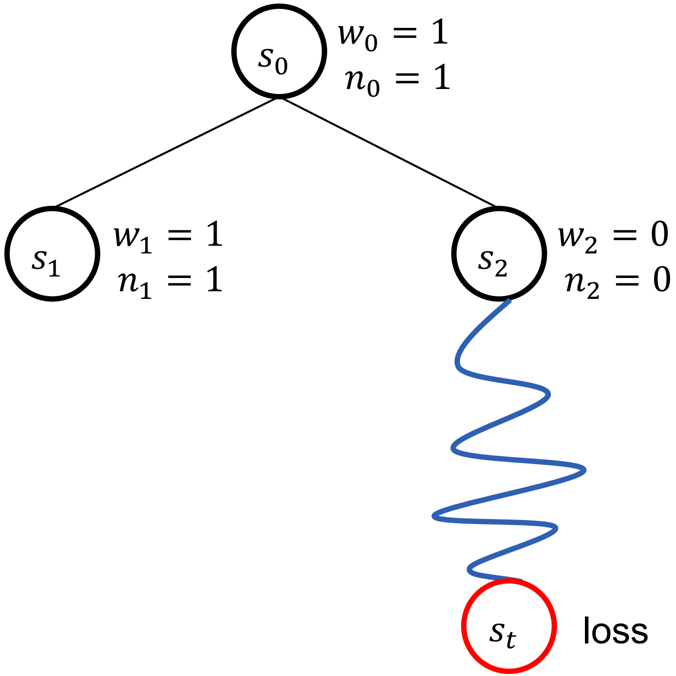
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

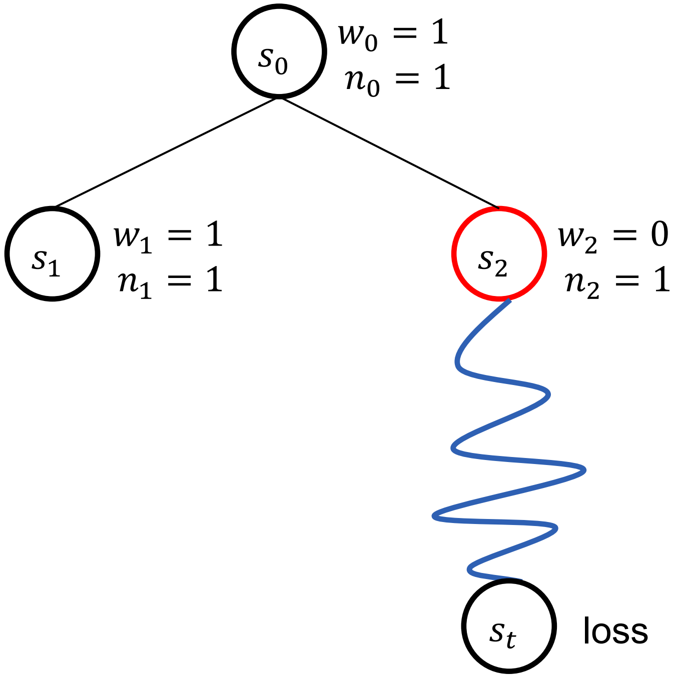
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

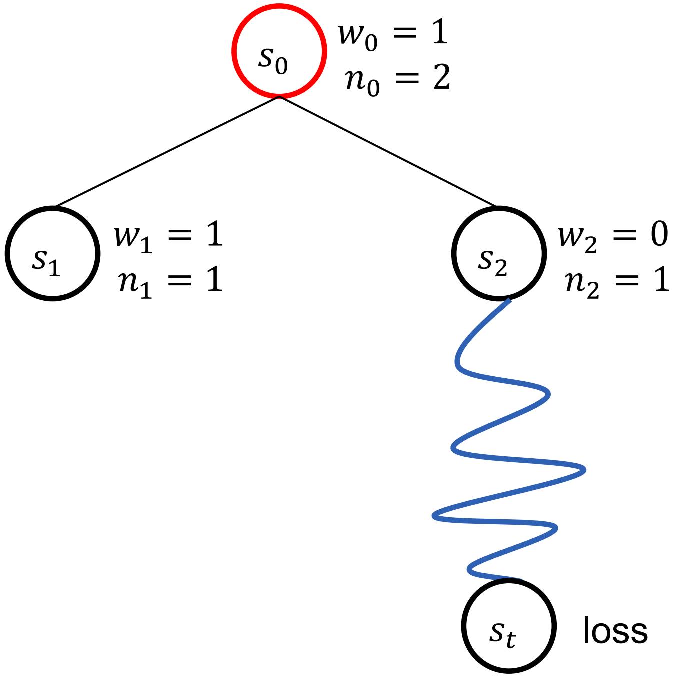
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

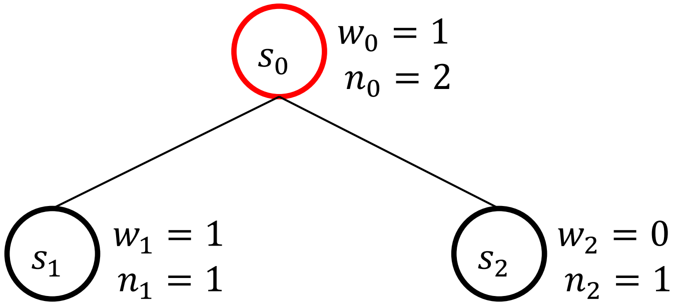
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

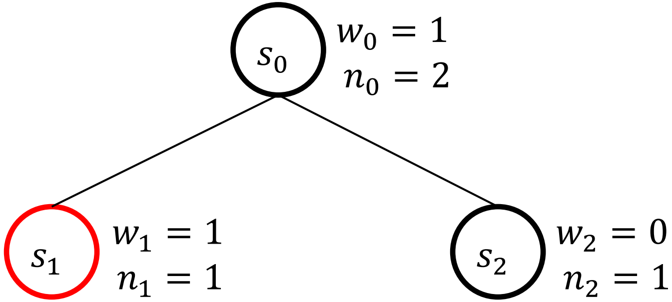
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

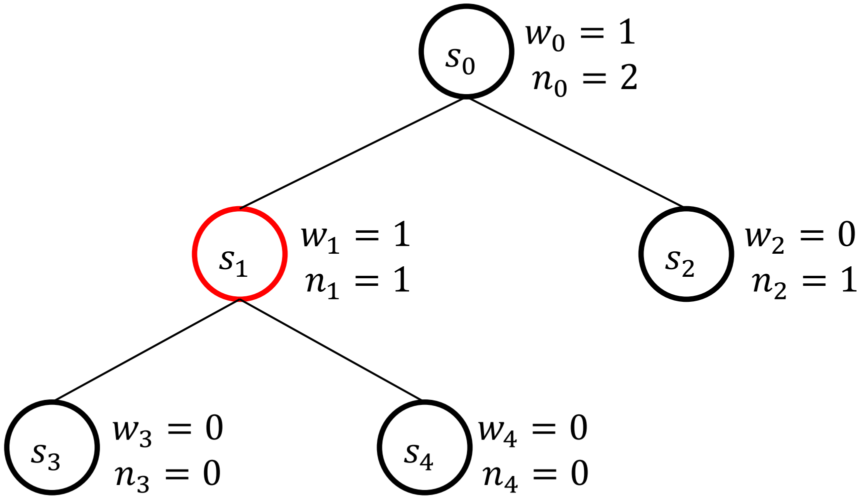
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

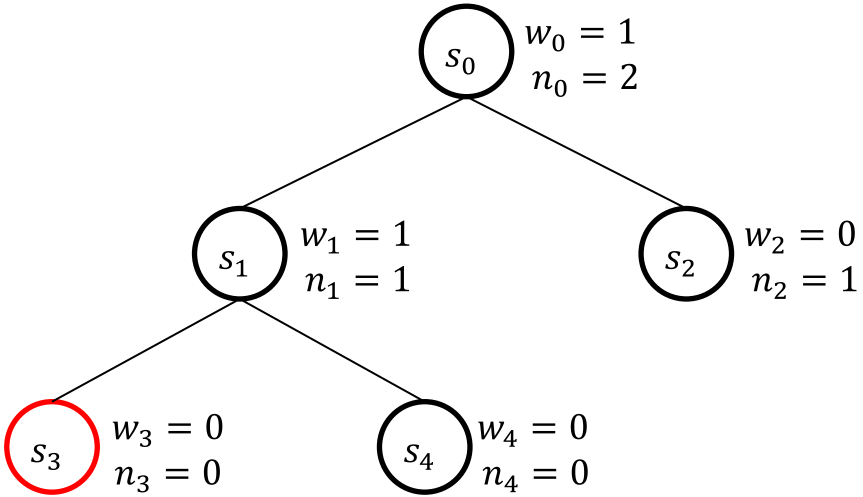
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

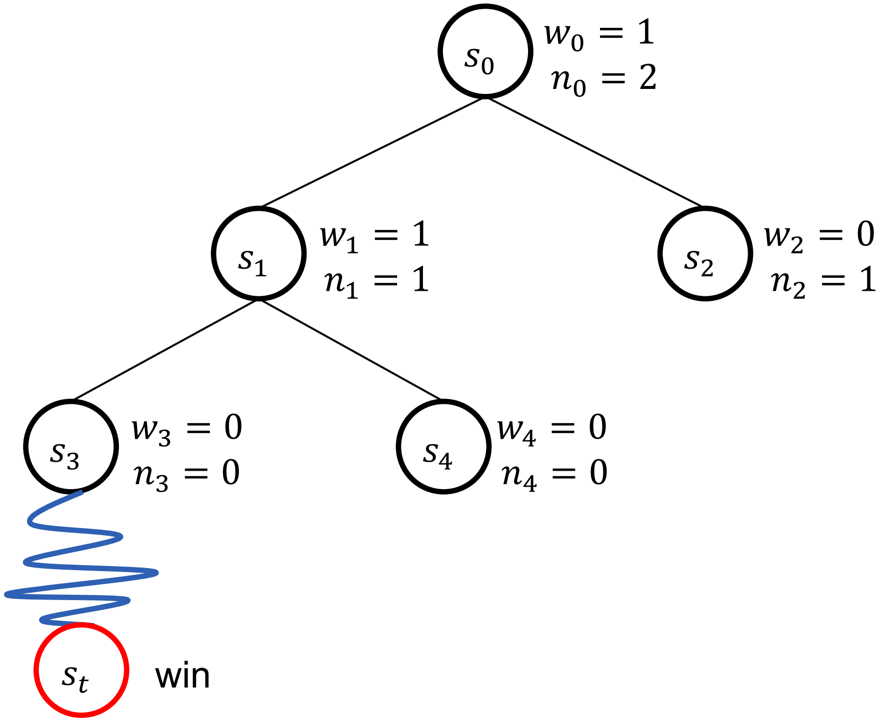
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

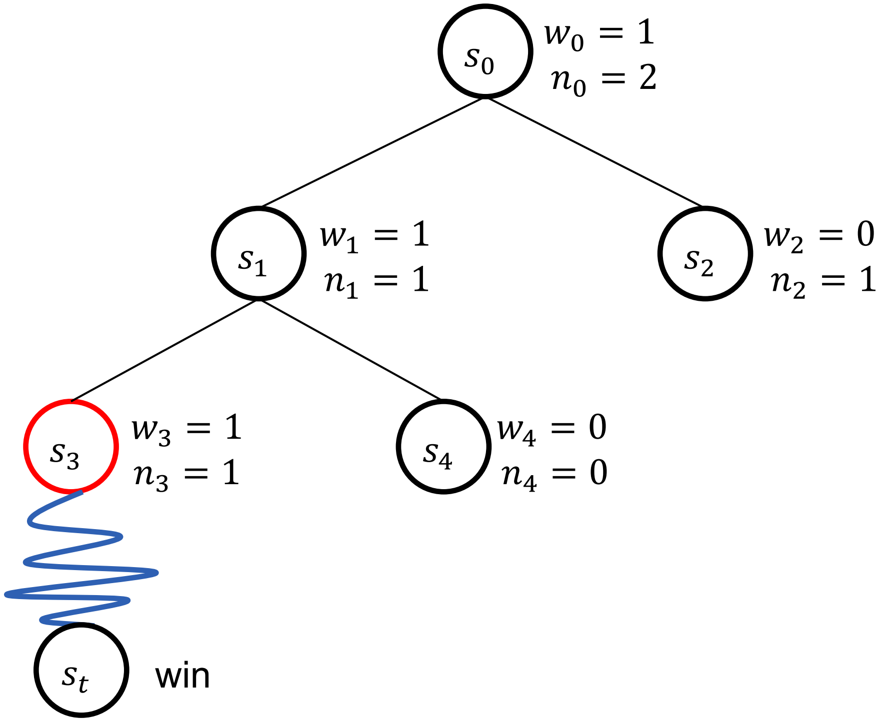
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

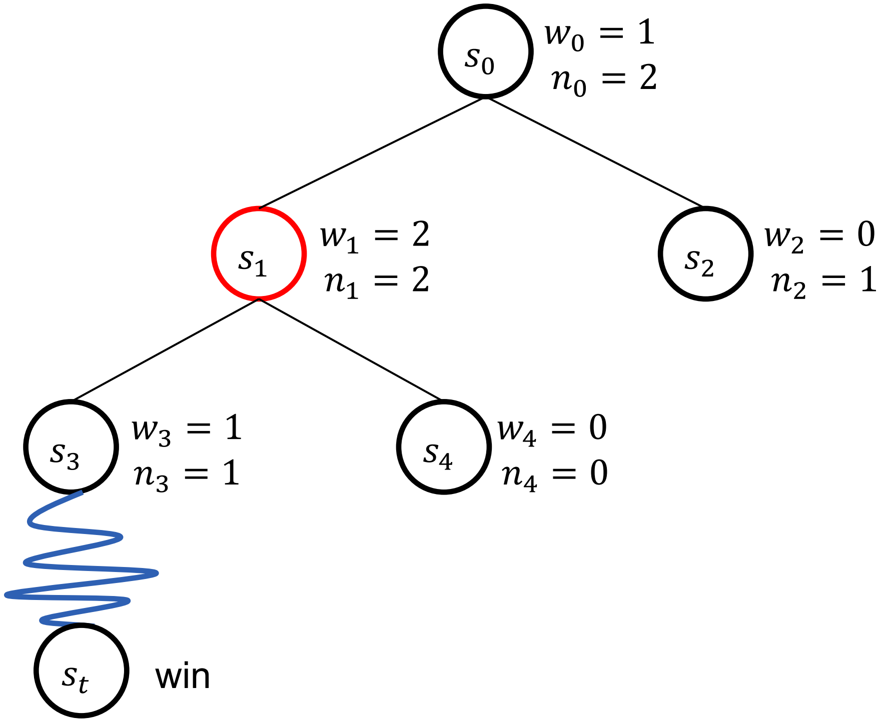
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

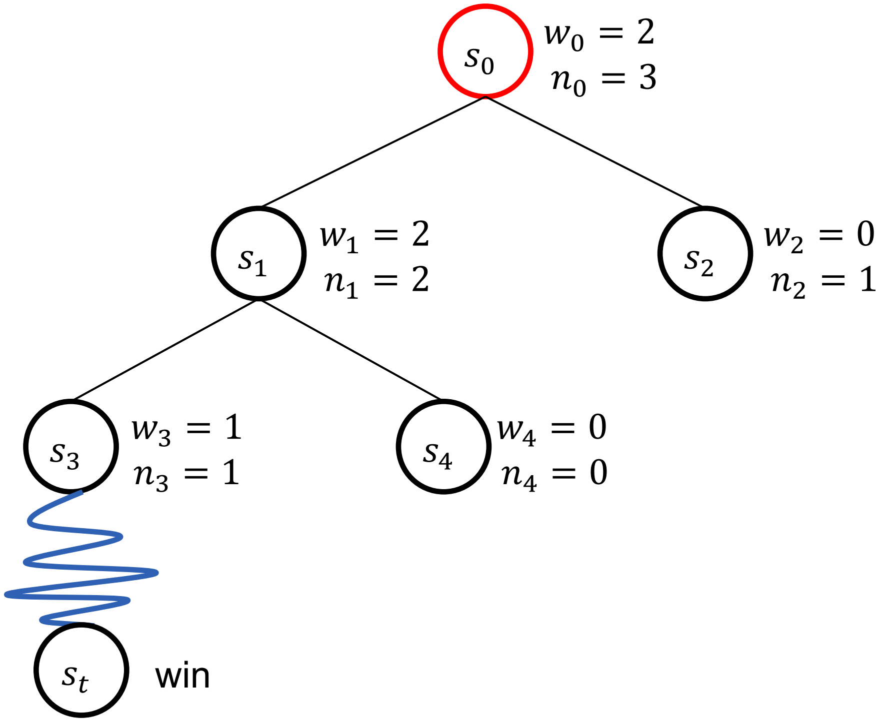
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

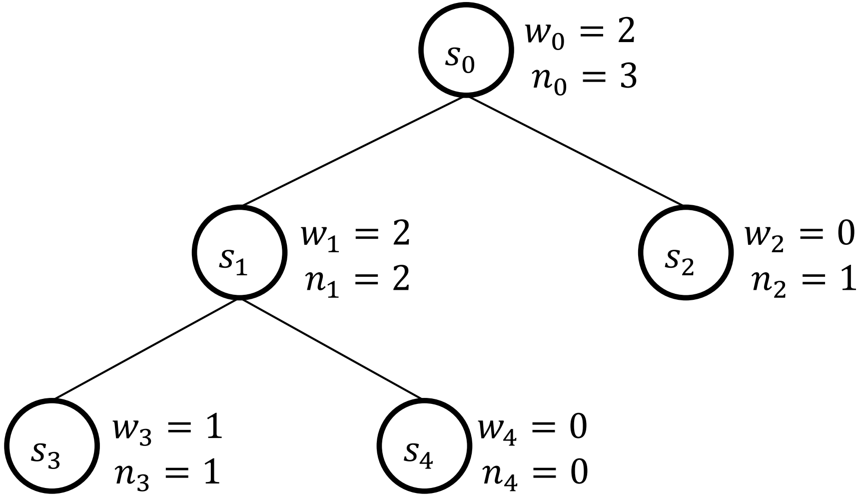
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

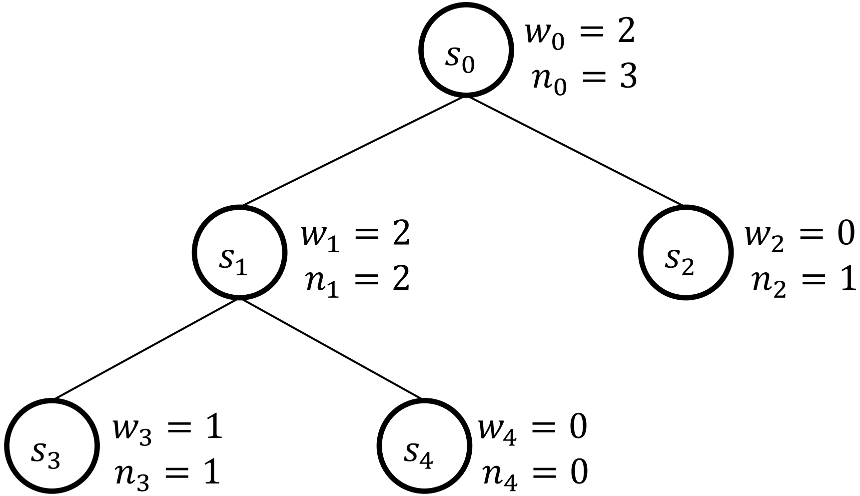
1. Selection
2. Expansion
3. Simulation
4. Backpropagation



MCTS

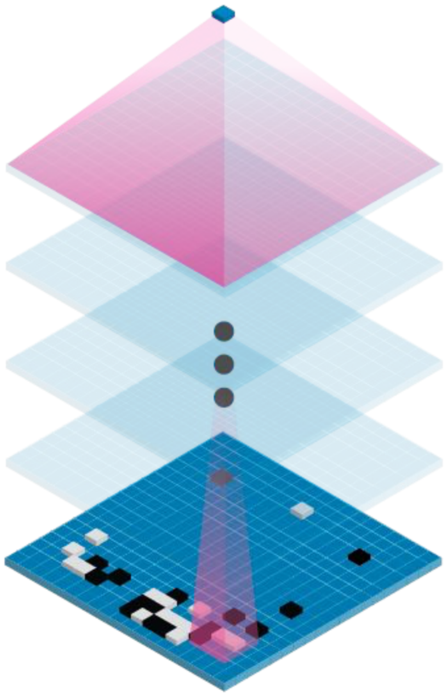
$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$

1. Selection
2. Expansion
3. Simulation
4. Backpropagation



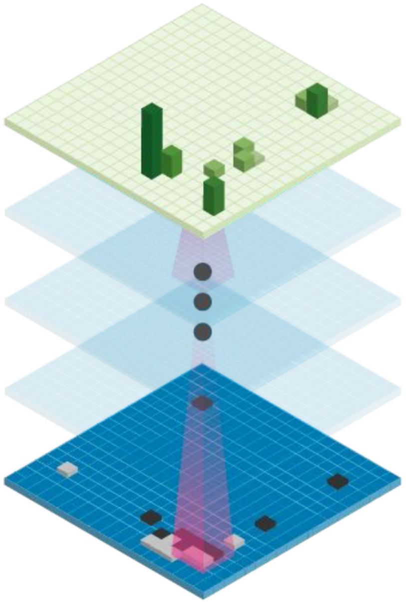
Value Network

How well are we doing?



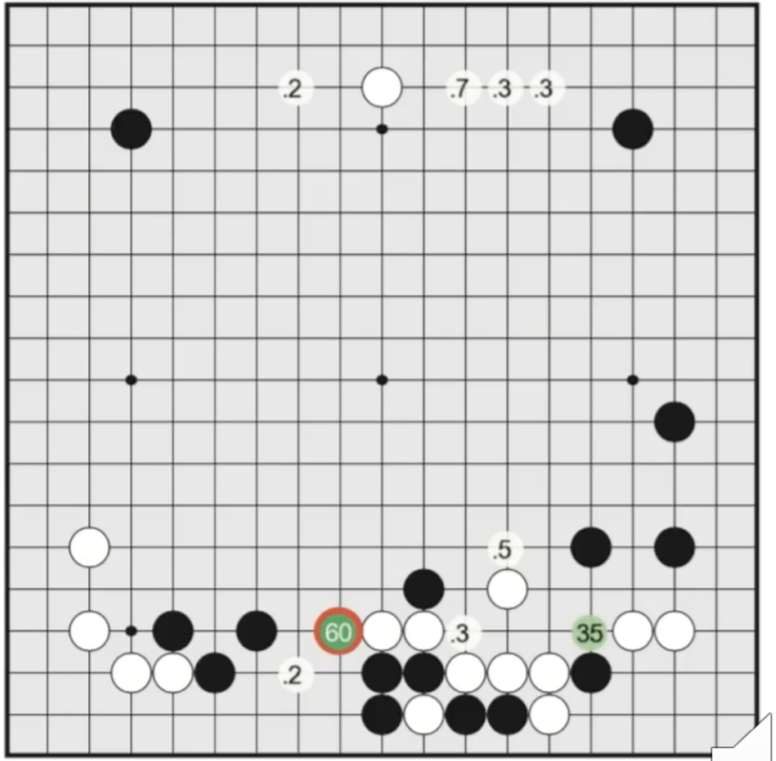
Policy Network

What are the most likely actions?

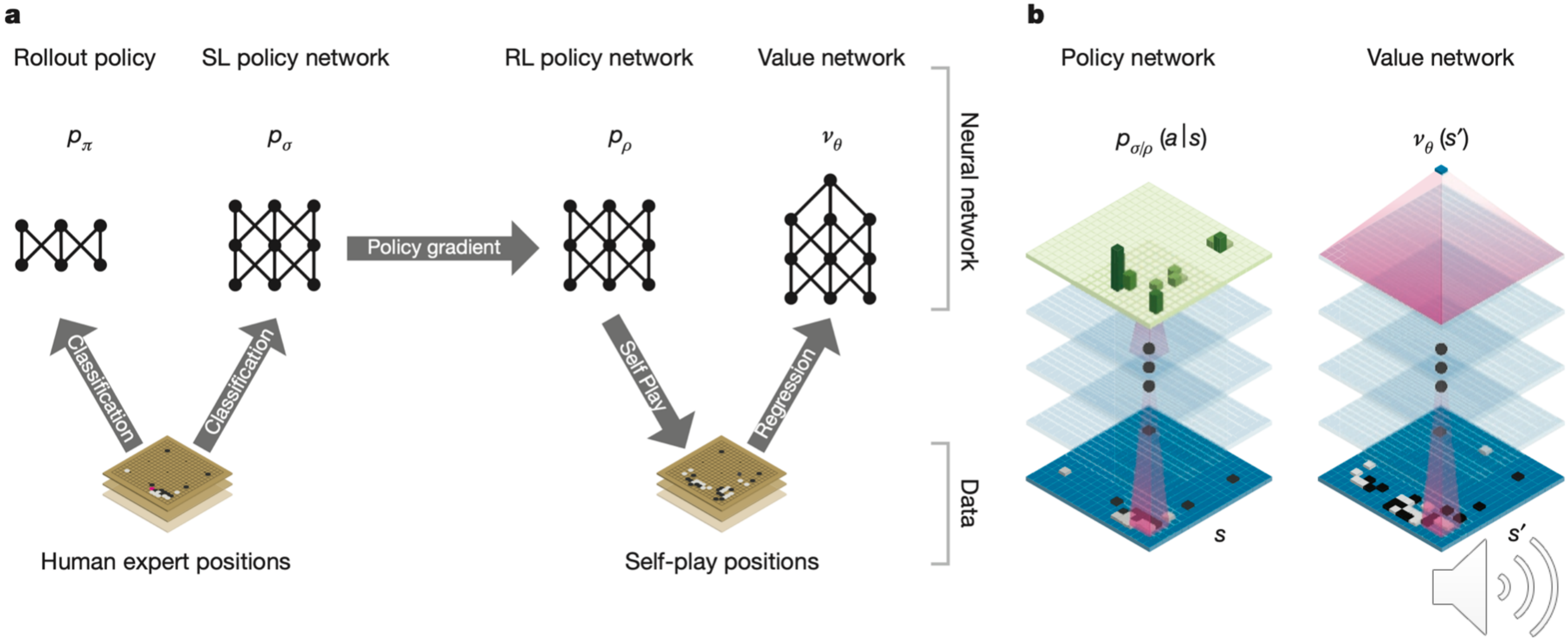


Policy Network

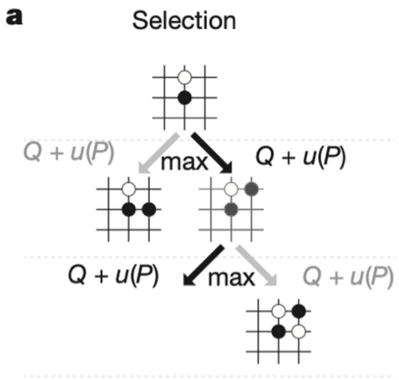
What are the most likely actions?



AlphaGo

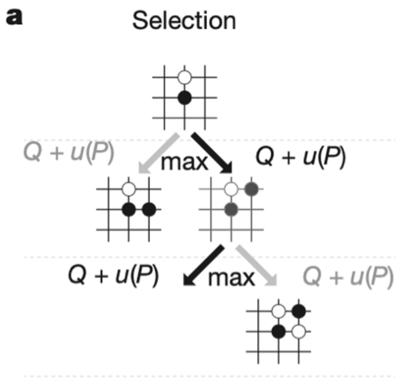

a
Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy gradient
Classification
Classification
Self Play
Regression
Neural network
Data
Human expert positions
Self-play positions
b
Policy network
Value network
$p_{\sigma/\rho}(a|s)$
$v_\theta(s')$
s
s'

AlphaGo MCTS



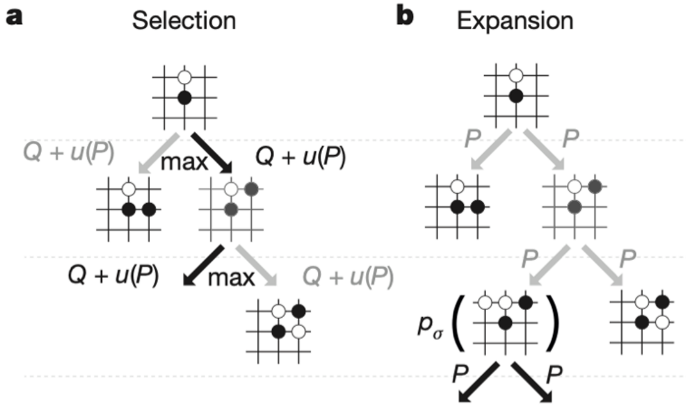
$$u(a) = v(a) + p(a) \cdot pb_c$$

AlphaGo MCTS



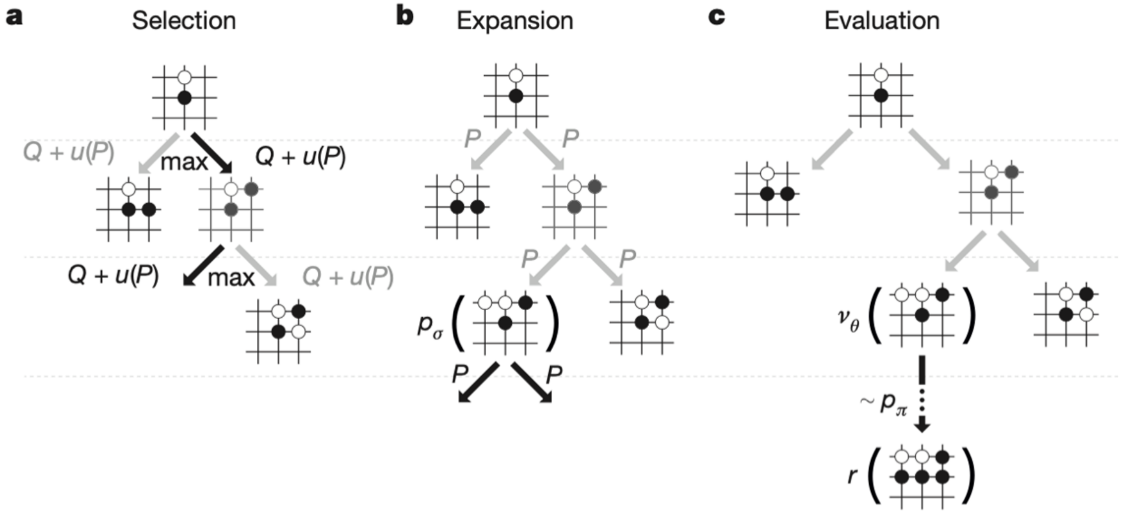
$$u(a) = v(a) + p(a) \cdot pb_c$$

AlphaGo MCTS



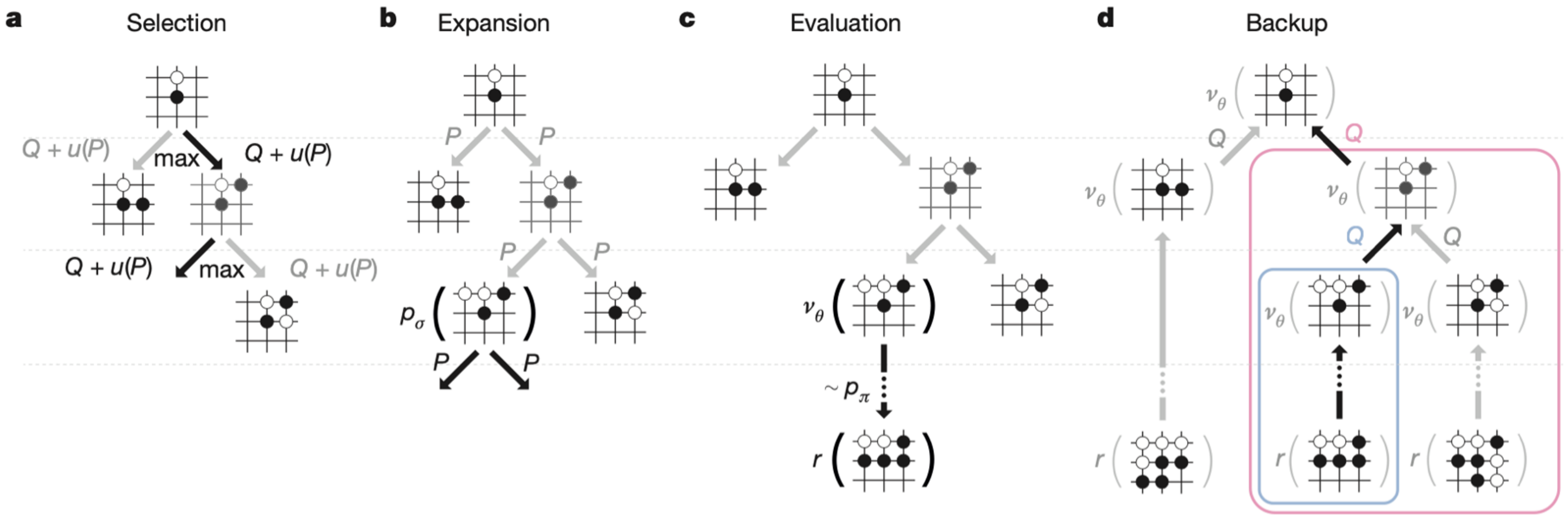
$$u(a) = v(a) + p(a) \cdot pb_c$$

AlphaGo MCTS



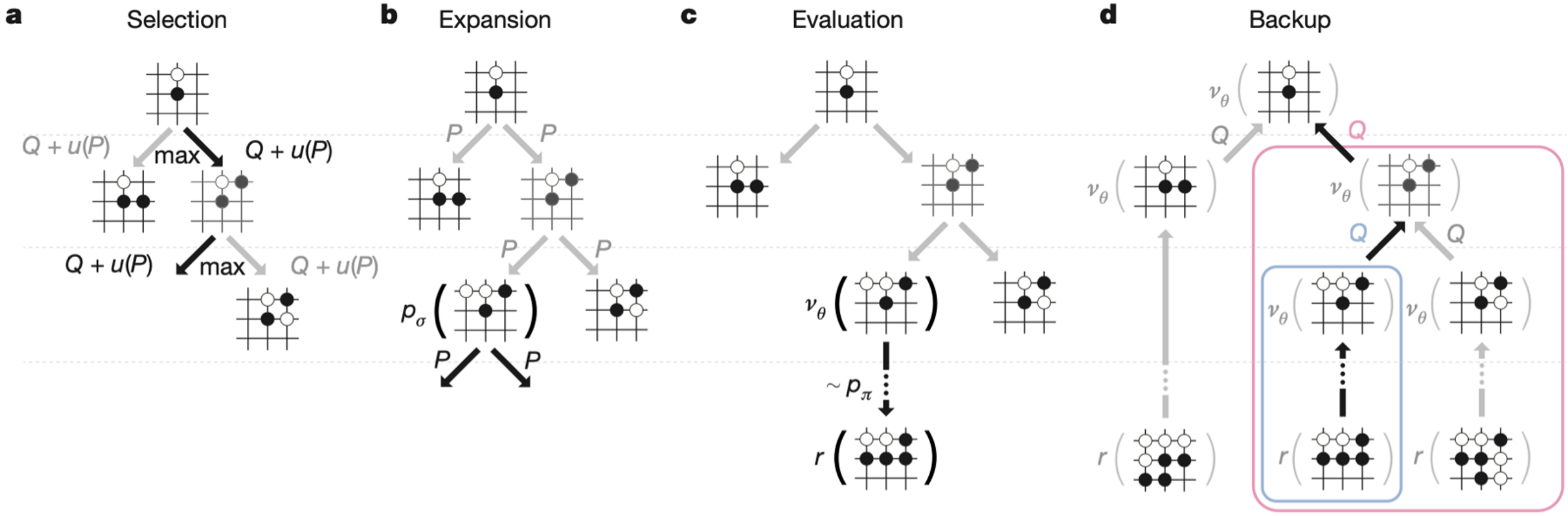
$$u(a) = v(a) + p(a) \cdot pb_c$$

AlphaGo MCTS



$$u(a) = v(a) + p(a) \cdot pb_c$$

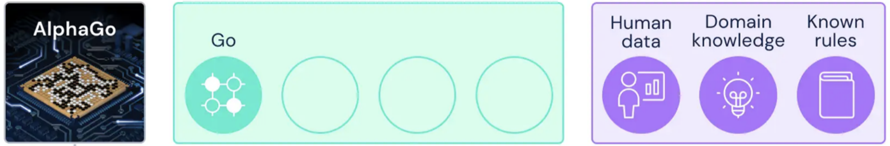
AlphaGo MCTS



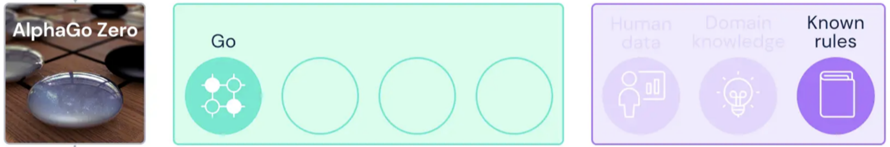
$$u(a) = v(a) + p(a) \cdot pb_c$$


ALPHAGO
01:59:50
LEE SEDOL
AlphaGo
Lee Sedol

Domains | Knowledge



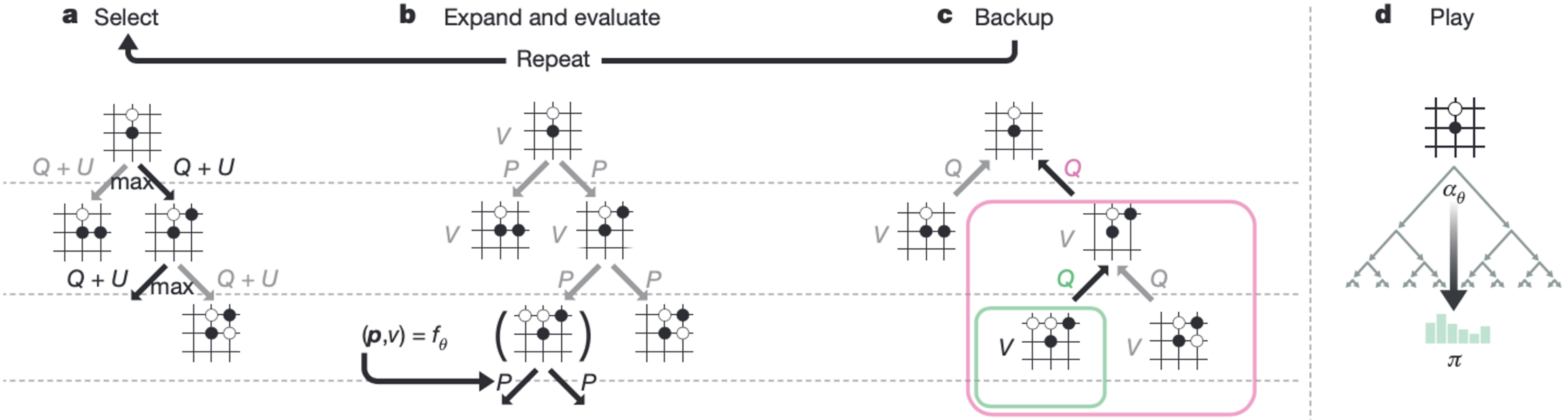
AlphaGo becomes the first program to master Go using neural networks and tree search
(Jan 2016, Nature)



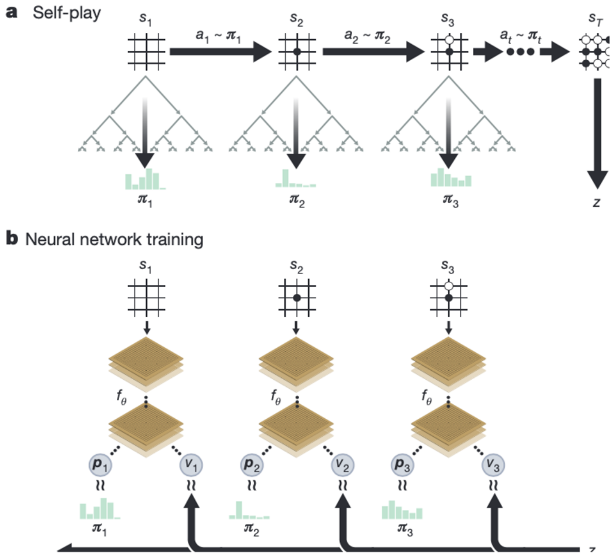
AlphaGo Zero learns to play completely on its own, without human knowledge
(Oct 2017, Nature)

AlphaZero masters three perfect information games using a single algorithm for all games
(Dec 2018, Science)

AlphaGo Zero MCTS



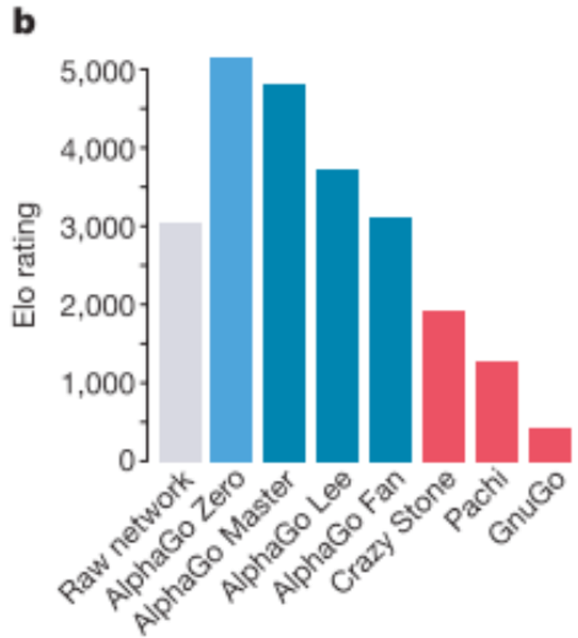
AlphaZero Training



$$(\mathbf{p}, v) = f_\theta(s),$$

$$l = (z - v)^2 - \boldsymbol{\pi}^\top \log \mathbf{p} + c||\theta||^2$$

AlphaGo Zero Results



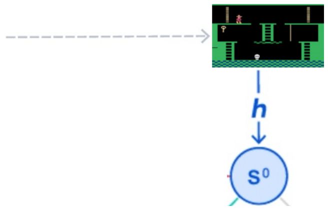
Atari

- Image Input
- No Access to rules

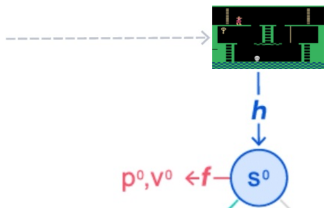
MuZero Planning

MuZero Planning

representation $h_\theta(o_1, .., o_t) = s^0$



MuZero Planning



representation $h_\theta(o_1, .., o_t) = s^0$

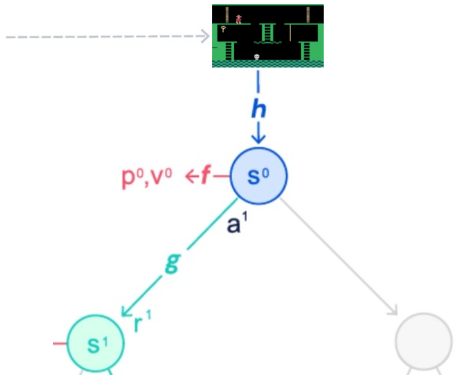
prediction $f_\theta(s^k) = \mathbf{p}^k, v^k$

MuZero Planning

representation $h_\theta(o_1, .., o_t) = s^0$

prediction $f_\theta(s^k) = \mathbf{p}^k, v^k$

dynamics $g_\theta(s^{k-1}, a^k) = r^k, s^k$

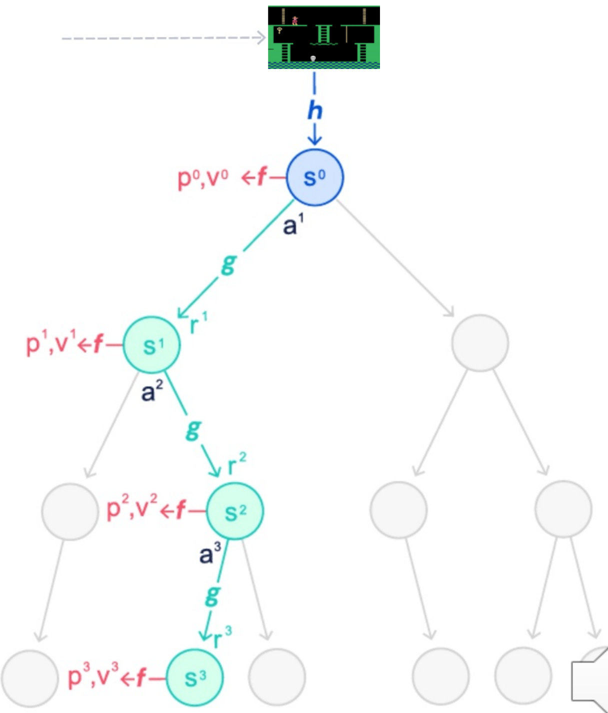


MuZero Planning

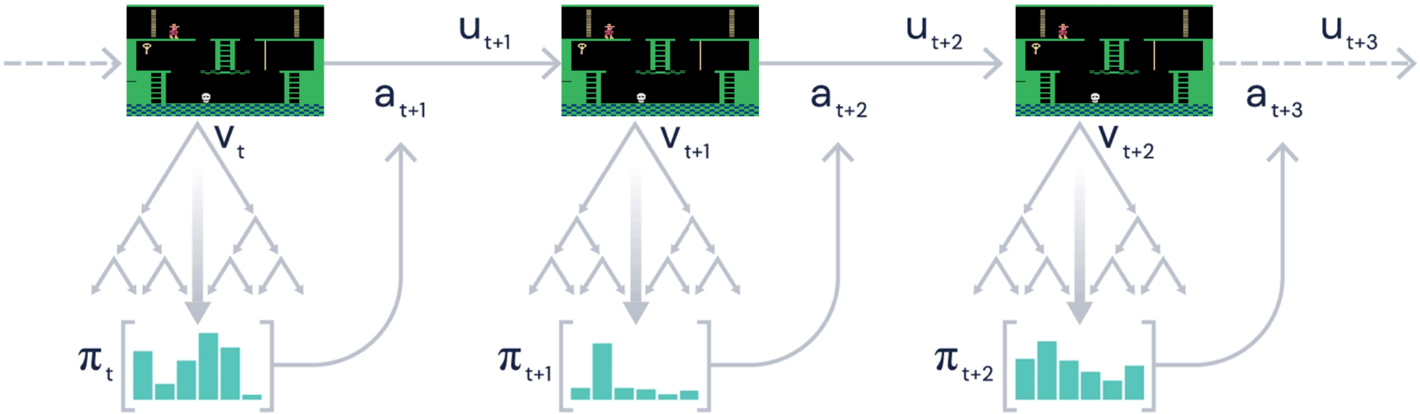
representation $h_\theta(o_1, .., o_t) = s^0$

prediction $f_\theta(s^k) = \mathbf{p}^k, v^k$

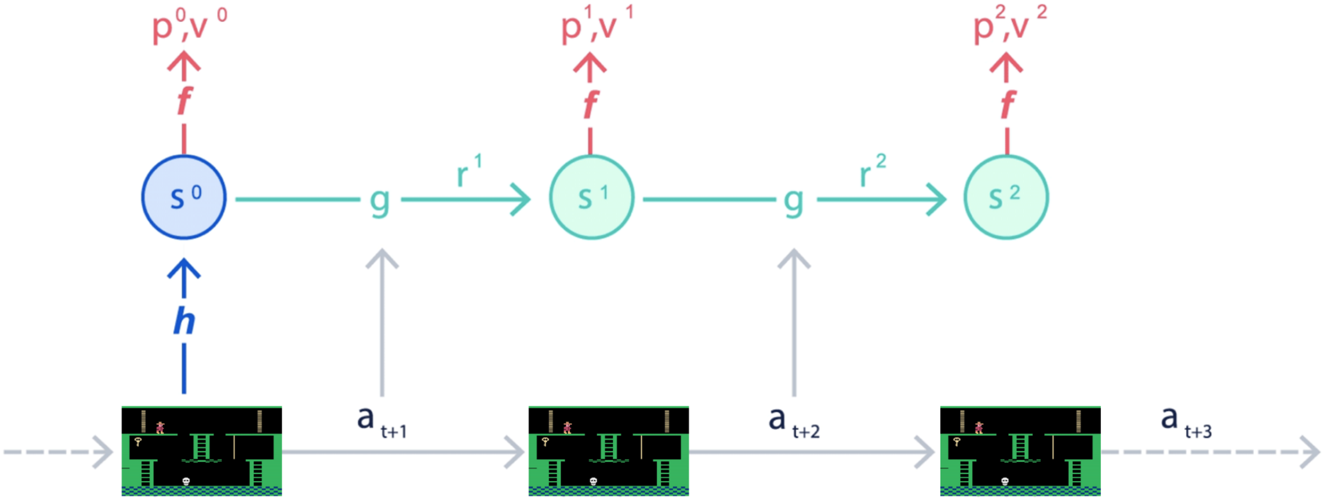
dynamics $g_\theta(s^{k-1}, a^k) = r^k, s^k$



MuZero Training Data Generation

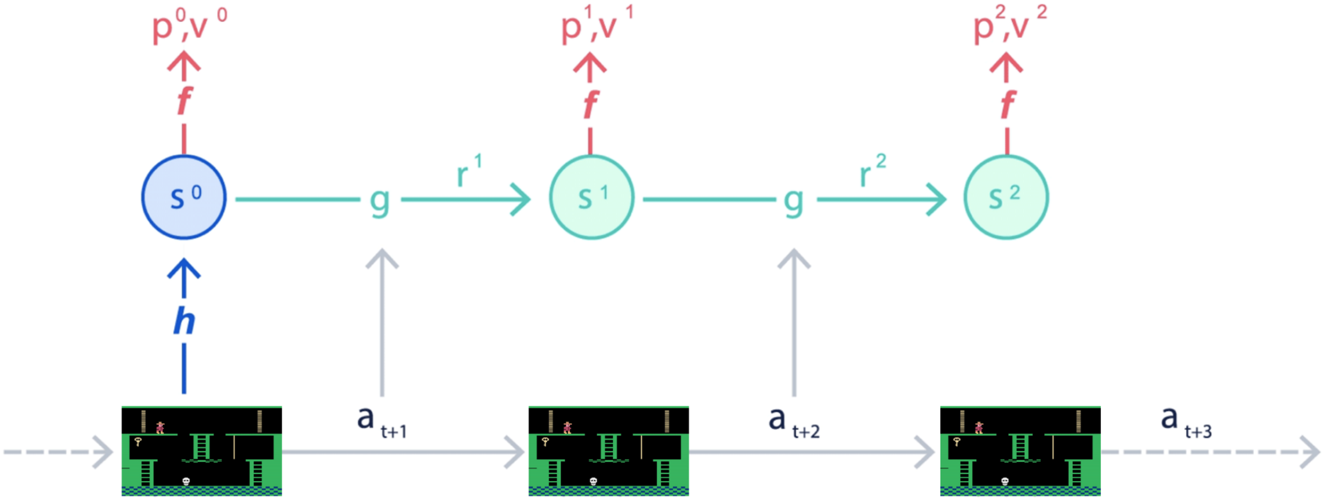


MuZero Training



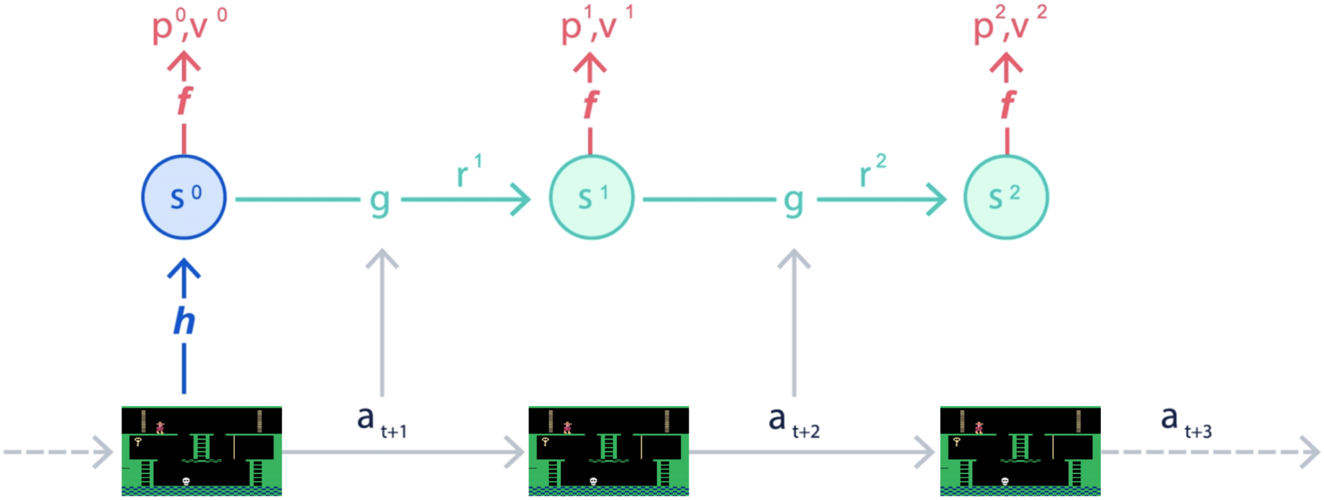
$$l_t(\theta) = \sum_{k=0}^{K} l^{p}(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^{v}(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^{r}(u_{t+k}, r_t^k) + c||\theta||^2$$

MuZero Training



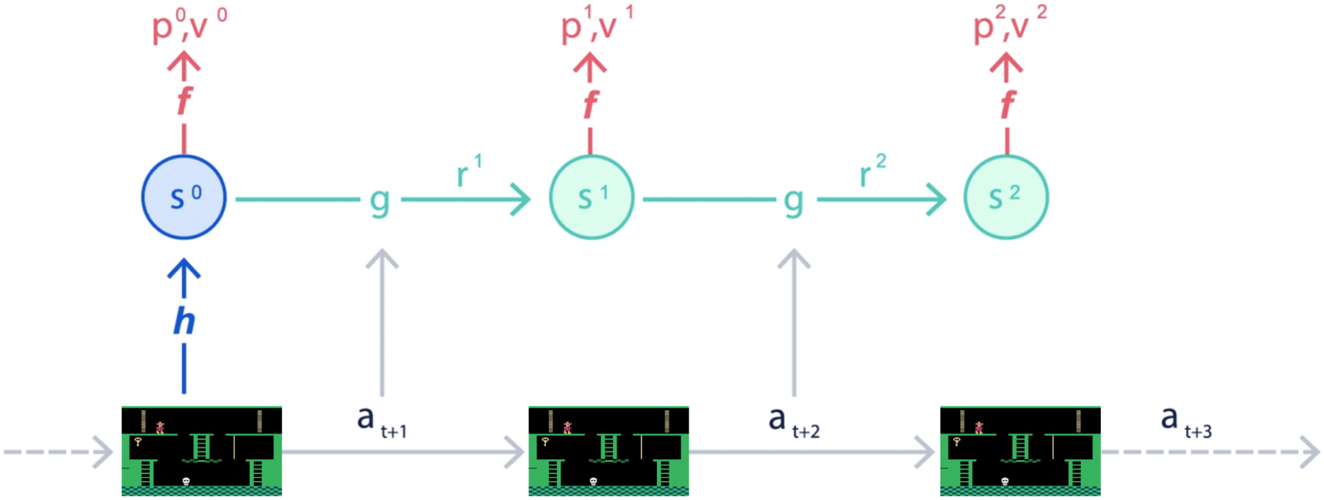
$$l_t(\theta) = \sum_{k=0}^{K} l^{\mathrm{p}}(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^{\mathrm{v}}(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^{\mathrm{r}}(u_{t+k}, r_t^k) + c||\theta||^2$$

MuZero Training



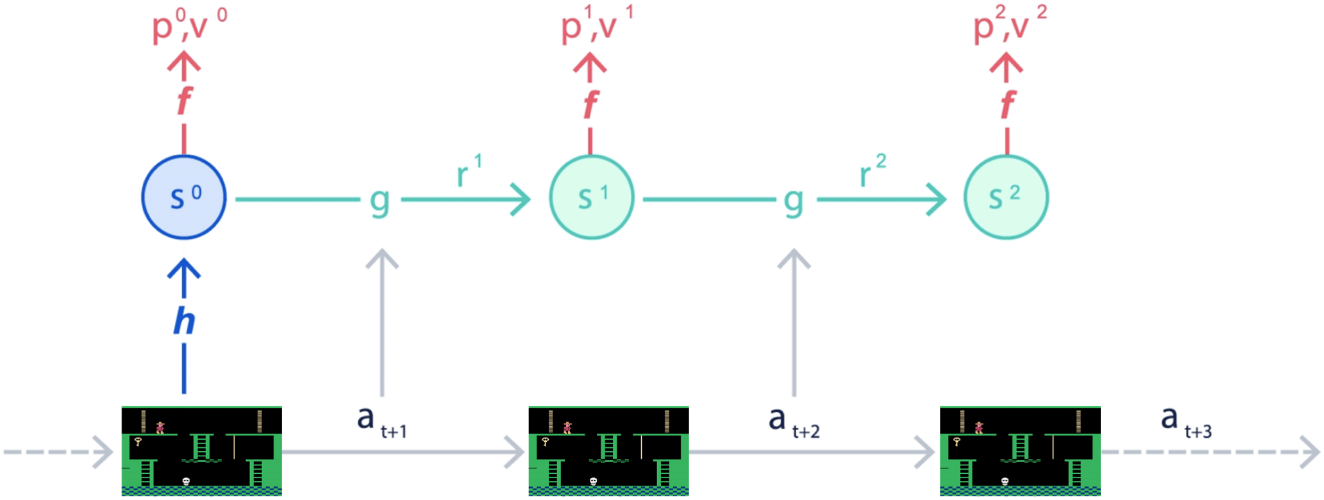
$$l_t(\theta) = \sum_{k=0}^{K} l^{\mathrm{p}}(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^{\mathrm{v}}(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^{\mathrm{r}}(u_{t+k}, r_t^k) + c\|\theta\|^2$$

MuZero Training



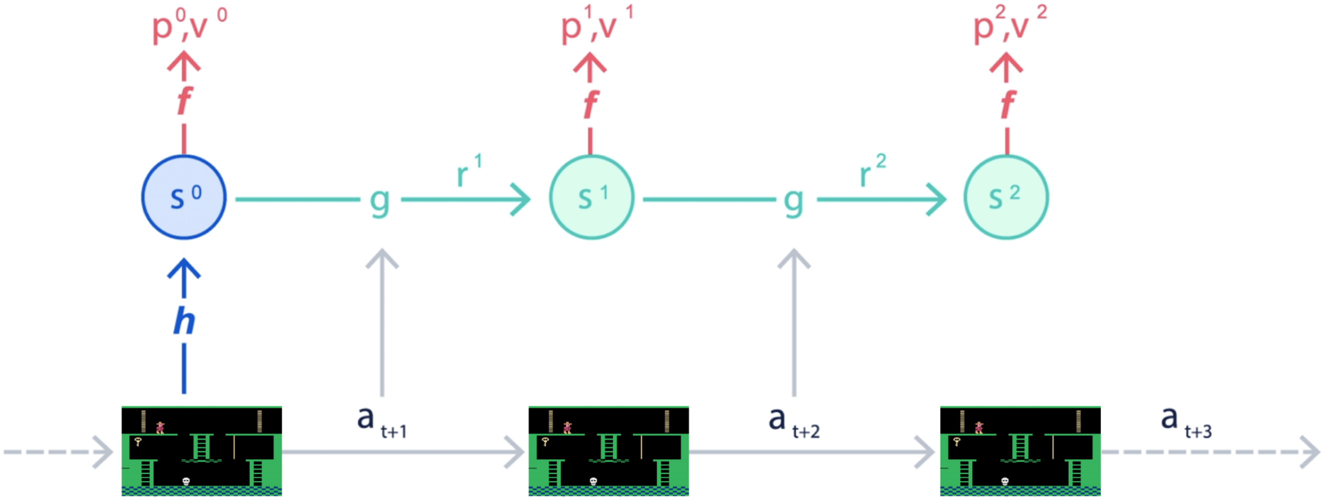
$$l_t(\theta) = \sum_{k=0}^{K} l^{\mathrm{p}}(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^{\mathrm{v}}(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^{\mathrm{r}}(u_{t+k}, r_t^k) + c\|\theta\|^2$$

MuZero Training



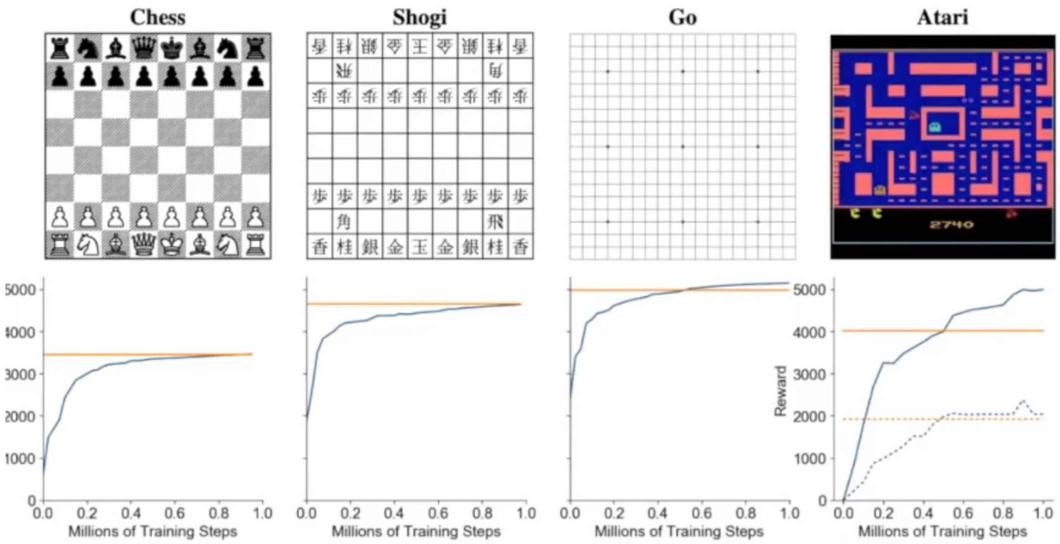
$$l_t(\theta) = \sum_{k=0}^{K} l^{\mathrm{p}}(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^{\mathrm{v}}(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^{\mathrm{r}}(u_{t+k}, r_t^k) + c||\theta||^2$$

MuZero Training

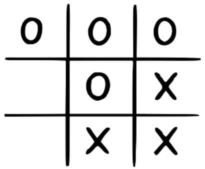


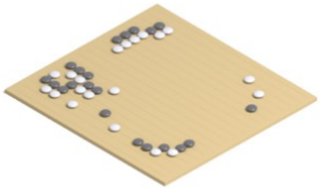
$$l_t(\theta) = \sum_{k=0}^{K} l^{\mathrm{p}}(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^{\mathrm{v}}(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^{\mathrm{r}}(u_{t+k}, r_t^k) + c||\theta||^2$$

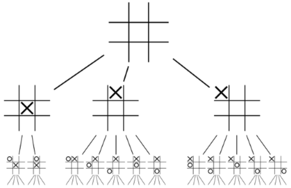
MuZero Results

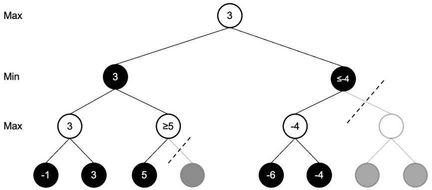


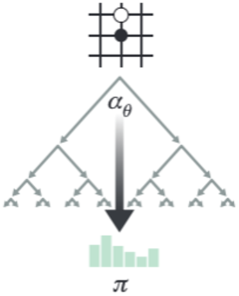
Summary

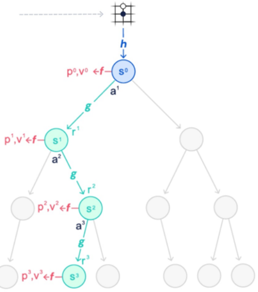












References

- 2020-10-22 MuZero ICAPS talk.pdf
- AlphaZero: Shedding new light on the grand games of chess, shogi and Go
- MuZero: Mastering Go, chess, shogi and Atari without rules
- https://en.wikipedia.org/wiki/Alpha–beta_pruning
- https://www.nature.com/articles/nature16961.pdf
- https://www.nature.com/articles/nature24270.epdf?author_access_token=VJXbVjaSHxFoctQQ4p2k4tRgN0jAjWel9jnR3ZoTv0PVW4gB86EEpGqTRDtpIz-2rmo8-KG06gqVobU5NSCFeHILHcVFUeMsbvwS-lxjqQGg98faovwjxeTUgZAUMnRQ