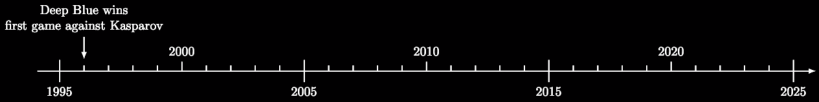
SOLVING QBFS WITH ALPHAZERO

Clemens Hofstadler, supervised by Univ.-Prof. Dr. Martina Seidl
Institute for Algebra, JKU Linz
Seminar Algebra and Discrete Mathematics, 10 June 2021

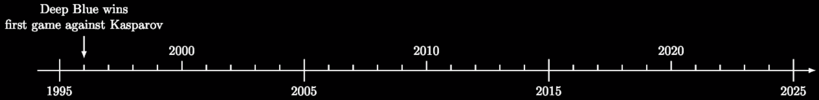
JOHANNES KEPLER
UNIVERSITY LINZ

THE ALPHAZERO FRAMEWORK

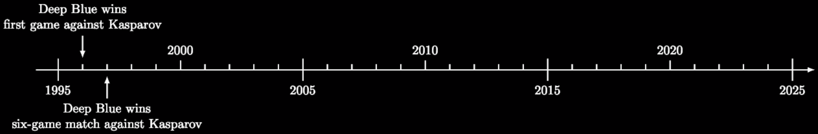
History


Deep Blue wins
first game against Kasparov
2000
2010
2020
1995
2005
2015
2025

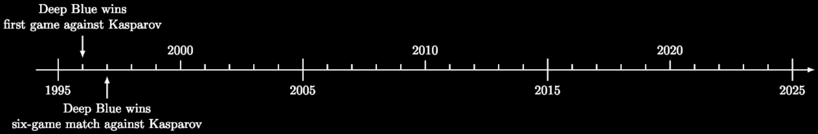
History

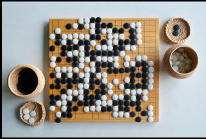

Deep Blue wins
first game against Kasparov
2000
2010
2020
1995
2005
2015
2025

History

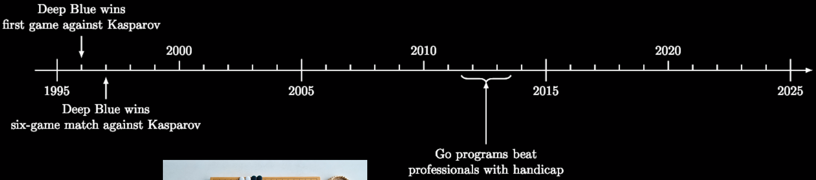

Deep Blue wins
first game against Kasparov
2000
2010
2020
1995
2005
2015
2025
Deep Blue wins
six-game match against Kasparov

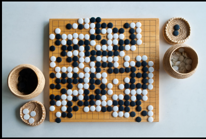
History


Deep Blue wins
first game against Kasparov
2000
2010
2020
1995
2005
2015
2025
Deep Blue wins
six-game match against Kasparov

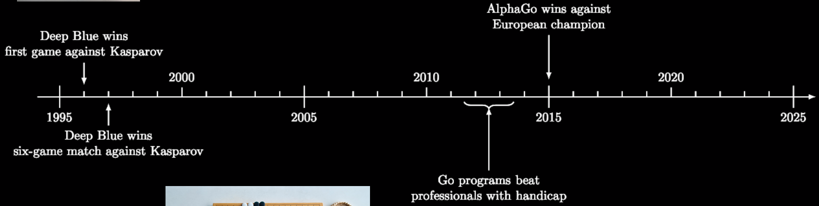


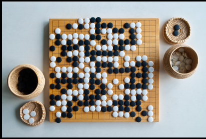
History


Deep Blue wins
first game against Kasparov
2000
2010
2020
1995
2005
2015
2025
Deep Blue wins
six-game match against Kasparov
Go programs beat
professionals with handicap

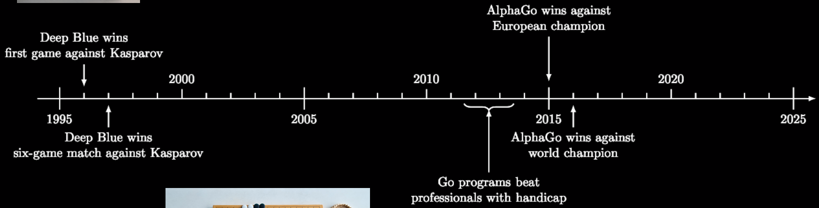


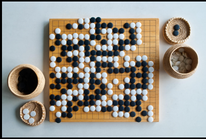
History


Deep Blue wins
first game against Kasparov
AlphaGo wins against
European champion
2000
2010
2020
1995
2005
2015
2025
Deep Blue wins
six-game match against Kasparov
Go programs beat
professionals with handicap



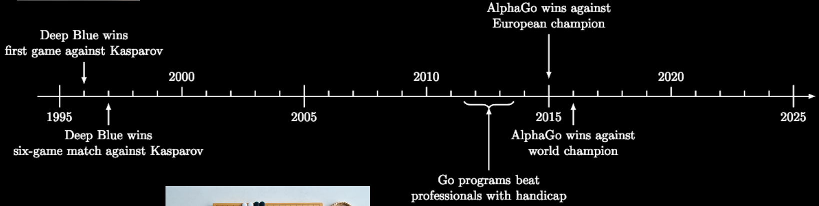
History

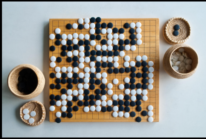

AlphaGo wins against
European champion
Deep Blue wins
first game against Kasparov
2000
2010
2020
1995
2005
2015
2025
Deep Blue wins
six-game match against Kasparov
AlphaGo wins against
world champion
Go programs beat
professionals with handicap



History

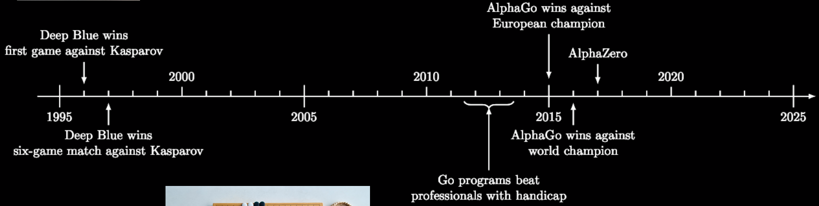

NETFLIX
ALPHAGO

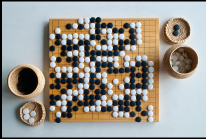

AlphaGo wins against
European champion
Deep Blue wins
first game against Kasparov
2000
2010
2020
1995
2005
2015
2025
Deep Blue wins
six-game match against Kasparov
AlphaGo wins against
world champion
Go programs beat
professionals with handicap



History

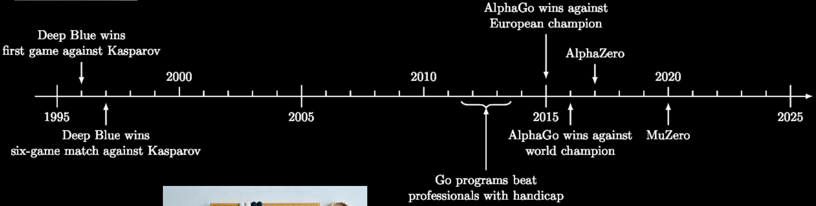

NETFLIX
ALPHAGO

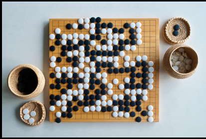

Deep Blue wins
first game against Kasparov
AlphaGo wins against
European champion
AlphaZero
2000
2010
2020
1995
2005
2015
2025
Deep Blue wins
six-game match against Kasparov
AlphaGo wins against
world champion
Go programs beat
professionals with handicap



History

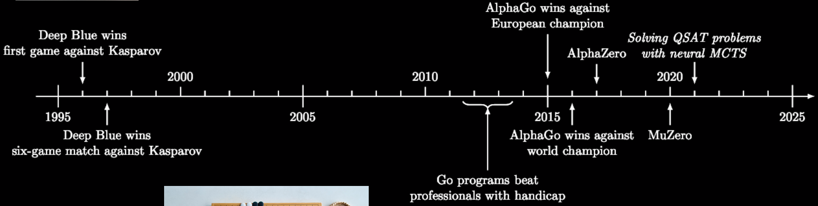

NETFLIX
ALPHAGO

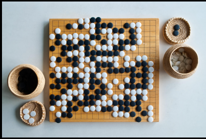

Deep Blue wins
first game against Kasparov
2000
2010
AlphaGo wins against
European champion
AlphaZero
2020
1995
Deep Blue wins
six-game match against Kasparov
2005
2015
AlphaGo wins against
world champion
MuZero
2025
Go programs beat
professionals with handicap



History


NETFLIX
ALPHAGO


Deep Blue wins
first game against Kasparov
AlphaGo wins against
European champion
AlphaZero
Solving QSAT problems
with neural MCTS
1995
2000
2005
2010
2015
2020
2025
Deep Blue wins
six-game match against Kasparov
AlphaGo wins against
world champion
MuZero
Go programs beat
professionals with handicap



AlphaZero

- AI framework to learn two-player, fully-observable, symmetric games
- AlphaZero learns (almost) tabula-rasa

AlphaZero

- AI framework to learn two-player, fully-observable, symmetric games
- AlphaZero learns (almost) tabula-rasa

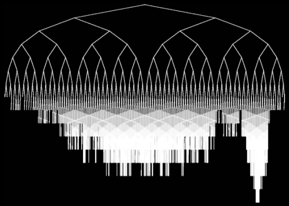
AlphaZero =

Monte Carlo tree search + Self-play reinforcement learning

AlphaZero

- AI framework to learn two-player, fully-observable, symmetric games
- AlphaZero learns (almost) tabula-rasa

$$\text{AlphaZero} = \text{Monte Carlo tree search} + \text{Self-play reinforcement learning}$$

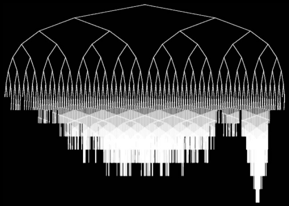


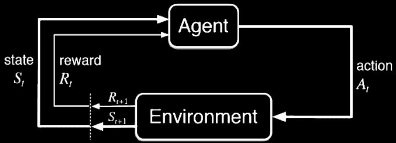
AlphaZero

- AI framework to learn two-player, fully-observable, symmetric games
- AlphaZero learns (almost) tabula-rasa

AlphaZero =

Monte Carlo tree search + Self-play reinforcement learning



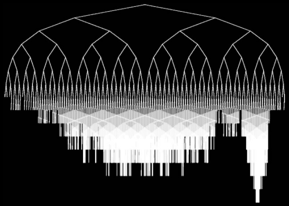


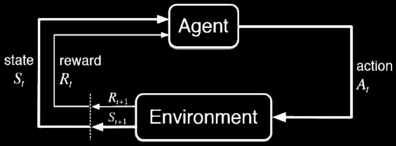
AlphaZero

- AI framework to learn two-player, fully-observable, symmetric games
- AlphaZero learns (almost) tabula-rasa

AlphaZero =

Monte Carlo tree search + Self-play reinforcement learning



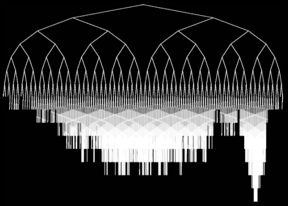


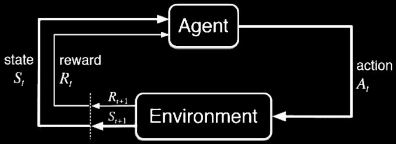
AlphaZero

- AI framework to learn two-player, fully-observable, symmetric games
- AlphaZero learns (almost) tabula-rasa

AlphaZero $=$

Monte Carlo tree search $+$ Self-play reinforcement learning





vs.

AlphaZero

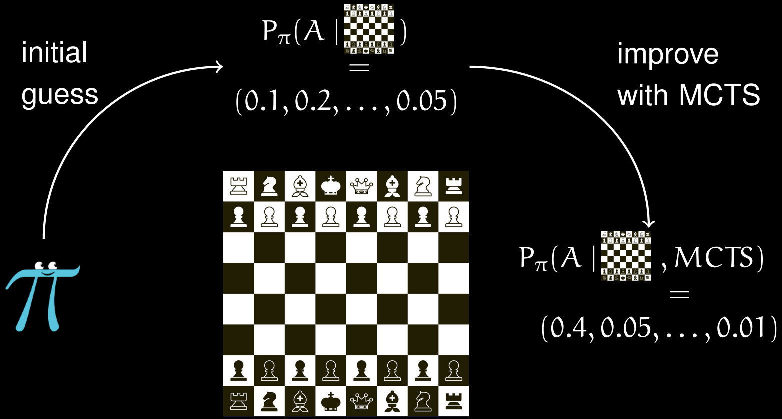
AlphaZero

initial guess

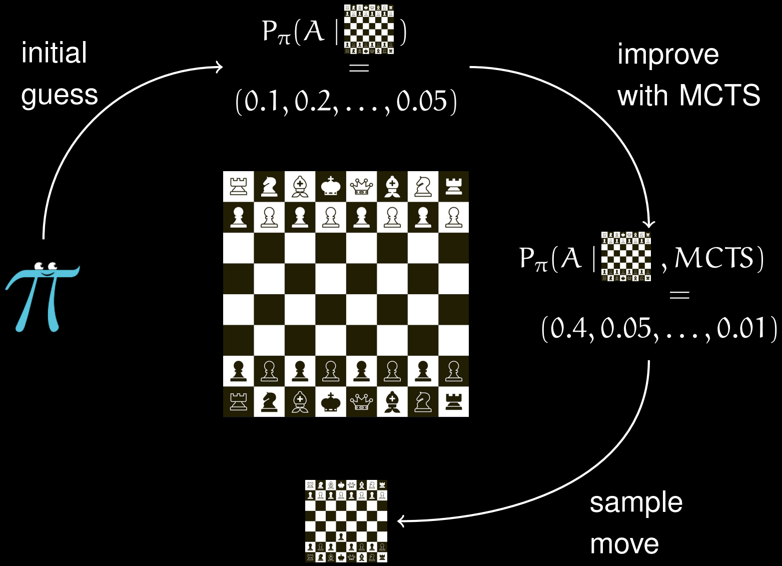
$P_\pi(A \mid$  $)$

$$= (0.1, 0.2, \ldots, 0.05)$$

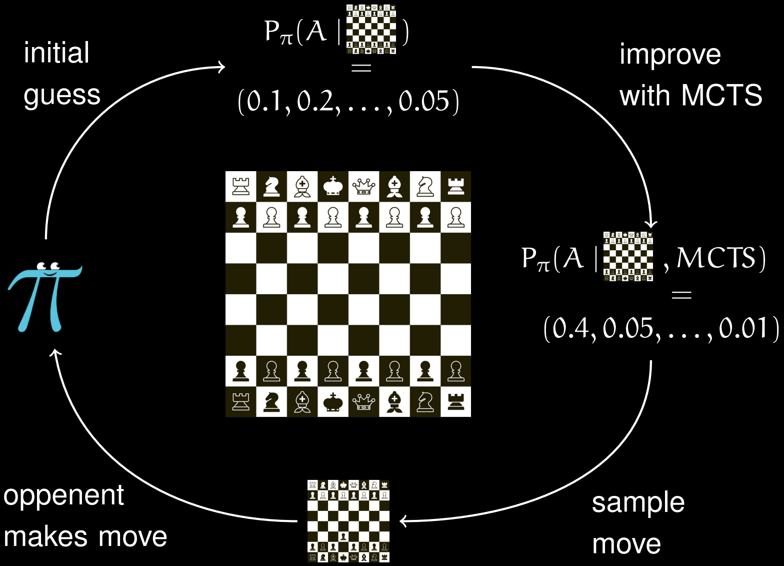
AlphaZero


initial guess
$P_\pi(A \mid \cdot) = (0.1, 0.2, \ldots, 0.05)$
improve with MCTS
$P_\pi(A \mid \cdot, MCTS) = (0.4, 0.05, \ldots, 0.01)$

AlphaZero


initial guess
$P_\pi(A \mid \text{[board]}) = (0.1, 0.2, \ldots, 0.05)$
improve with MCTS
$P_\pi(A \mid \text{[board]}, \text{MCTS}) = (0.4, 0.05, \ldots, 0.01)$
sample move

AlphaZero



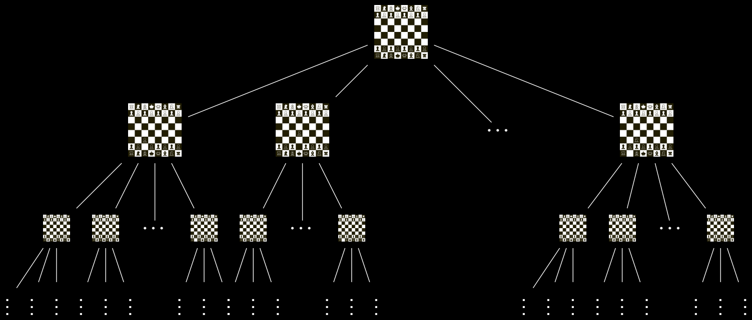
$P_\pi(A \mid \text{board}) = (0.1, 0.2, \ldots, 0.05)$
initial guess
improve with MCTS
$P_\pi(A \mid \text{board}, MCTS) = (0.4, 0.05, \ldots, 0.01)$
sample move
oppenent makes move

MCTS

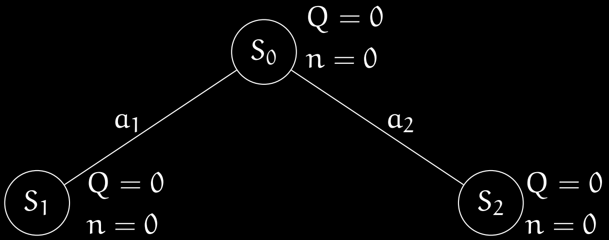
Given: Two-player, fully-observable game with game tree

Task: Find promising action in current state

Example:

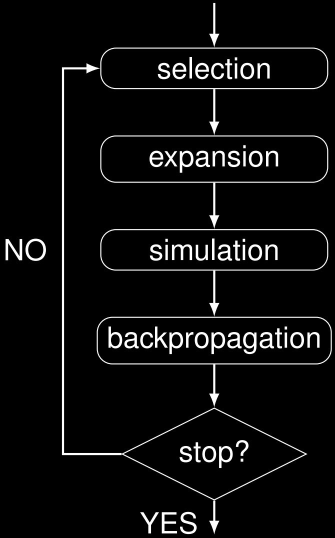


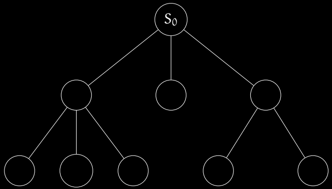
MCTS



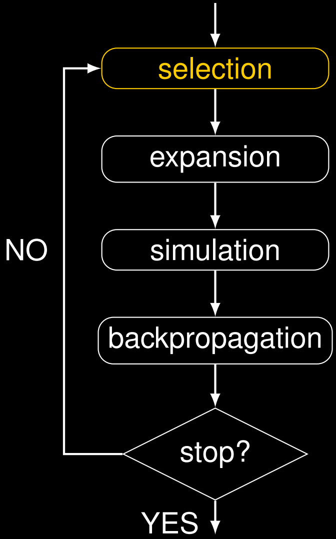
- States: S_0, S_1, S_2
- Actions: a_1, a_2
- Attributes of nodes
  - Q: current value of the node
  - n: number of visits

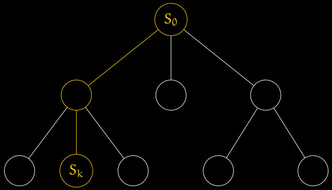
MCTS


selection
expansion
simulation
backpropagation
stop?
NO
YES


s_0

MCTS





Traverse tree following
upper confidence bound

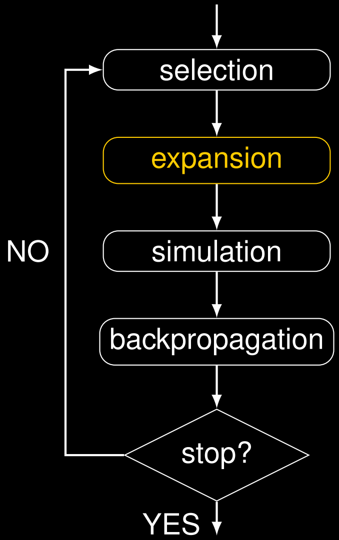
$$\text{UCB}(S) = \frac{Q}{n} + \gamma\sqrt{\frac{\ln N}{n}}$$

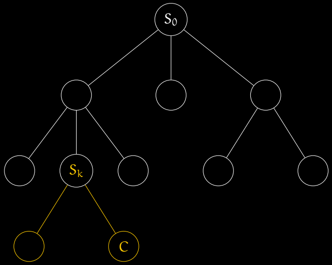
$$\text{UCB}(S) = \infty, \quad \text{if } n = 0.$$

γ . . . hyperparamter

N . . . # visits of parent node

MCTS





if not visited S_k and $k \neq 0$

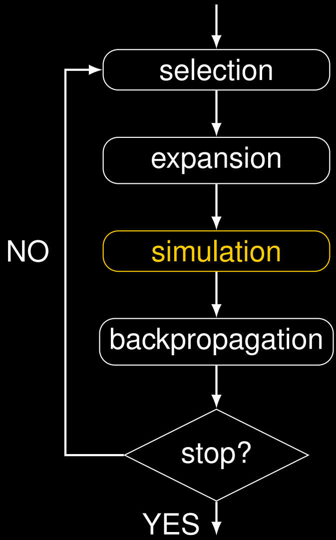
simulate S_k

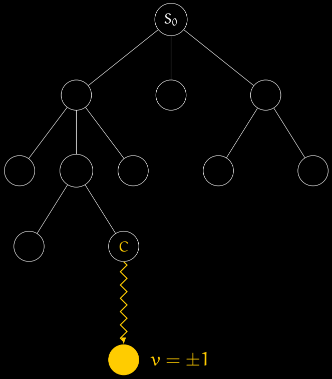
else

expand S_k

simulate child C

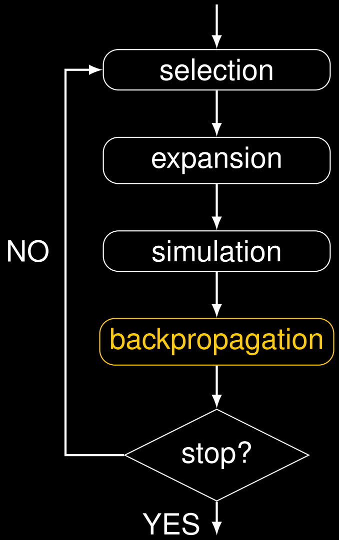
MCTS

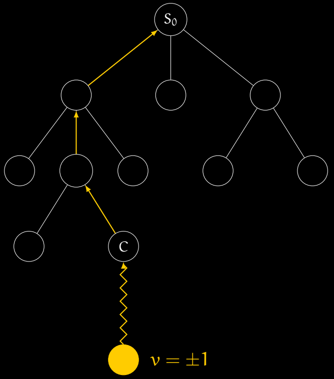




Play random game.

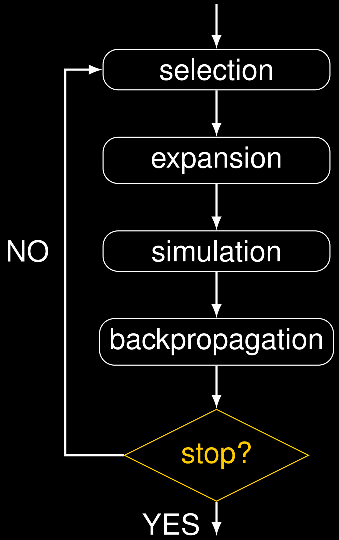
MCTS

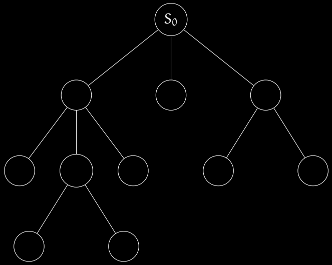




Update Q values for nodes on path from C to S_0 using v.

MCTS



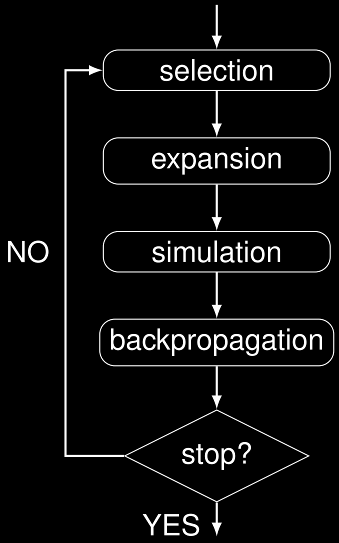


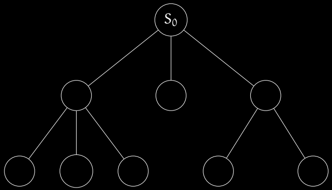
Termination criteria:

- Timeout
- Max. number of iterations exceeded

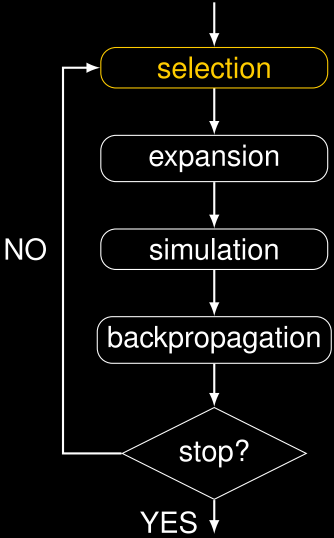
Return relative frequencies

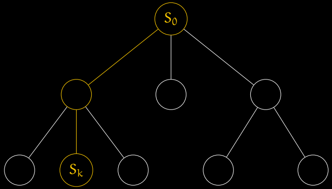
MCTS in AlphaZero





MCTS in AlphaZero





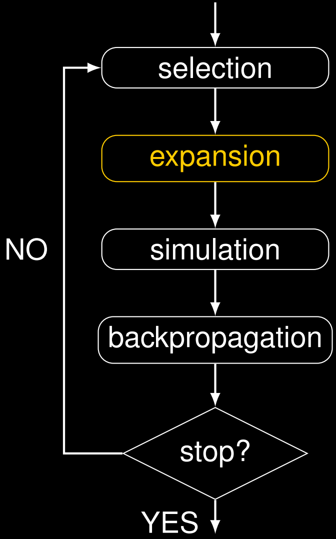
Traverse tree following
Predictor + UCB

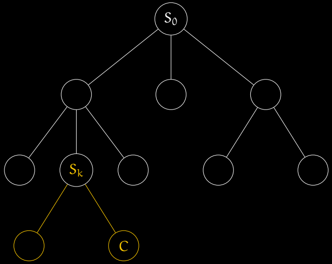
$$\text{PUCB}(S) = \frac{Q}{n+1} + \gamma \frac{P_\pi(a \mid P(S))\sqrt{N}}{n+1}$$

$P(S)$ ... Parent of S

a ... action from $P(S)$ to S

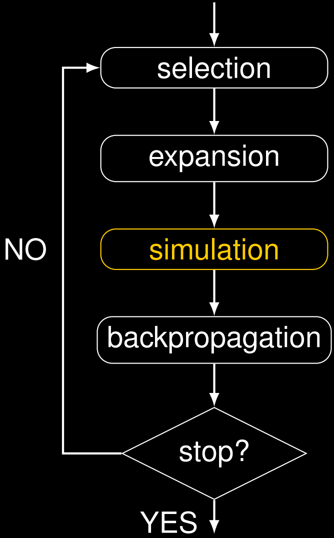
MCTS in AlphaZero

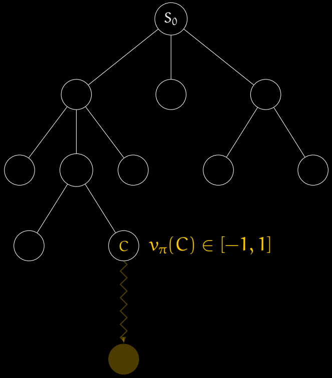




if not visited S_k
  compute $P_\pi(A \mid S_k) = p_\pi(S_k)$
if not visited S_k and $k \neq 0$
  simulate S_k
else
  expand S_k
  simulate child C with action
  $a = \text{argmax}_{a'} P_\pi(a' \mid S_k)$

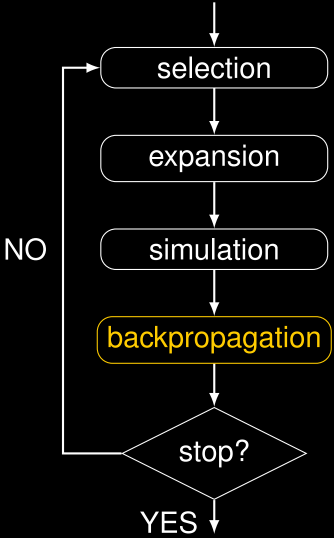
MCTS in AlphaZero

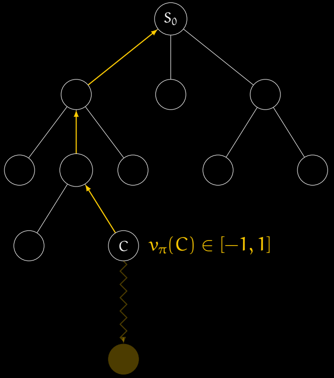




Estimate $v \approx v_\pi(C)$ using the neural network.

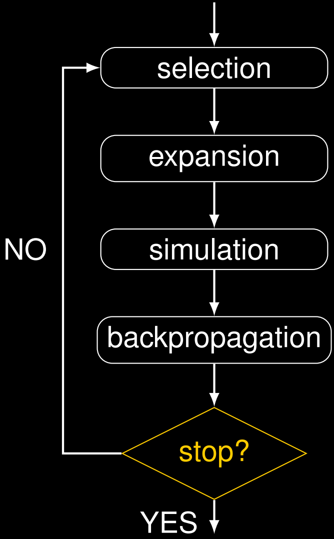
MCTS in AlphaZero

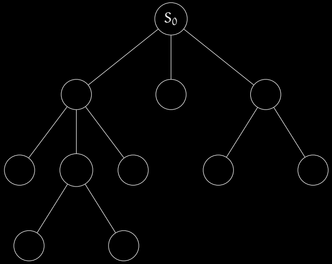




Update Q values for nodes on path from C to S_0 using $v_\pi(C)$.

MCTS in AlphaZero





Termination criteria:

- Timeout
- Max. number of iterations exceeded

Return relative frequencies

Training by self-play

```
Initialize player with random parameters π
for e ← 1, . . . , E do
    Generate data by self-play games
    Update parameters π
    Compare new player to best player
return π
```

Training by self-play

3: Generate data by self-play games

$t \leftarrow 1$
for $k \leftarrow 1, \ldots, N$ **do**
  $S_t \leftarrow$ initial board
  while S_t is not an end state **do**
    $p_t \leftarrow P_\pi(A \mid S_t, MCTS)$
    sample move a_t from p_t
    save data (S_t, p_t, z_t), where z_t is the game result
    $S_{t+1} \leftarrow$ new board after doing move a_t
    $t \leftarrow t + 1$
return $\{(S_t, p_t, z_t) \mid 1 \leq t \leq T\}$

Training by self-play

4: Update parameters π

1: Given data $\{(S_t, p_t, z_t) \mid 1 \leq t \leq T\}$, use gradient descent to update parameters π to minimize

$$L = \sum_{t=1}^{T} \left((z_t - v_\pi(S_t))^2 - p_t^T \log p_\pi(S_t) \right) + c\|\pi\|^2,$$

with hyperparameter c.

Training by self-play

5: Compare new player to best player

$\pi' \leftarrow$ parameters of the best previous player
let π play M games against π'
if π wins $\geq 55\%$ of these game **then**
    mark π as the best player
else
    $\pi \leftarrow \pi'$
return π

ALPHAZERO FOR QBF SOLVING

QBFs

QBF = Quantified Boolean Formula

- Extension of propositional logic over boolean variables with $\exists, \forall$
- Canonical PSPACE-complete problem
- Many application domains: planning, model checking,...

QBFs

QBF = Quantified Boolean Formula

- Extension of propositional logic over boolean variables with $\exists, \forall$
- Canonical PSPACE-complete problem
- Many application domains: planning, model checking,…

$$\exists v, w \, \forall x, y \, \exists z. \, (x \vee z) \wedge (v \vee \bar{y} \vee \bar{z}) \wedge \bar{w}$$

QBFs

QBF = Quantified Boolean Formula

- Extension of propositional logic over boolean variables with $\exists, \forall$
- Canonical PSPACE-complete problem
- Many application domains: planning, model checking,...

$$\overbrace{\exists v, w \, \forall x, y \, \exists z. \, (x \vee z) \wedge (v \vee \bar{y} \vee \bar{z}) \wedge \bar{w}}^{\text{closed QBF in prenex CNF}}$$

QBFs

QBF = Quantified Boolean Formula

- Extension of propositional logic over boolean variables with $\exists, \forall$
- Canonical PSPACE-complete problem
- Many application domains: planning, model checking,...

$$\overbrace{\underbrace{\exists v, w \, \forall x, y \, \exists z.}_{\text{prefix}} \, (x \vee z) \wedge (v \vee \bar{y} \vee \bar{z}) \wedge \bar{w}}^{\text{closed QBF in prenex CNF}}$$

QBFs

QBF = Quantified Boolean Formula

- Extension of propositional logic over boolean variables with $\exists, \forall$
- Canonical PSPACE-complete problem
- Many application domains: planning, model checking,...

closed QBF in prenex CNF

$$\overbrace{\underbrace{\exists v, w\, \forall x, y\, \exists z.}_{\text{prefix}}\ \underbrace{(x \vee z) \wedge (v \vee \bar{y} \vee \bar{z}) \wedge \bar{w}}_{\text{matrix (CNF)}}}^{\text{closed QBF in prenex CNF}}$$

QBFs

QBF = Quantified Boolean Formula

- Extension of propositional logic over boolean variables with $\exists, \forall$
- Canonical PSPACE-complete problem
- Many application domains: planning, model checking,...

$$\overbrace{\underbrace{\exists v, w\, \forall x, y\, \exists z.}_{\text{prefix}}\ \underbrace{(x \vee z) \wedge (v \vee \bar{y} \vee \bar{z}) \wedge \bar{w}}_{\text{matrix (CNF)}}}^{\text{closed QBF in prenex CNF}}$$

clause: $(v \vee \bar{y} \vee \bar{z})$

QBF semantics

- A closed QBF can either be true or false.

QBF semantics

- A closed QBF can either be true or false.
- The QBF $\top$ is true.
- The QBF $\bot$ is false.

QBF semantics

- A closed QBF can either be true or false.
- The QBF $\top$ is true.
- The QBF $\bot$ is false.
- The QBF $\forall x\, \mathcal{Q}.\, \varphi$ is true $\iff$

  $\mathcal{Q}.\, \varphi[x = \top]$ and $\mathcal{Q}.\, \varphi[x = \bot]$ are true.

QBF semantics

- A closed QBF can either be true or false.
- The QBF $\top$ is true.
- The QBF $\bot$ is false.
- The QBF $\forall x\, \mathcal{Q}.\, \varphi$ is true $\iff$
  $\mathcal{Q}.\, \varphi[x = \top]$ and $\mathcal{Q}.\, \varphi[x = \bot]$ are true.
- The QBF $\exists x\, \mathcal{Q}.\, \varphi$ is true $\iff$
  $\mathcal{Q}.\, \varphi[x = \top]$ or $\mathcal{Q}.\, \varphi[x = \bot]$ is true.

QBF semantics

- A closed QBF can either be true or false.
- The QBF $\top$ is true.
- The QBF $\bot$ is false.
- The QBF $\forall x\, \mathcal{Q}.\, \varphi$ is true $\iff$

  $\mathcal{Q}.\, \varphi[x = \top]$ and $\mathcal{Q}.\, \varphi[x = \bot]$ are true.
- The QBF $\exists x\, \mathcal{Q}.\, \varphi$ is true $\iff$

  $\mathcal{Q}.\, \varphi[x = \top]$ or $\mathcal{Q}.\, \varphi[x = \bot]$ is true.

Task: Given QBF $\mathcal{Q}.\varphi$, determine whether $\mathcal{Q}.\varphi$ is true or false.

QBF solving as a game

Task: Given QBF $\mathcal{Q}.\varphi$, determine whether $\mathcal{Q}.\varphi$ is true or false.

QBF solving as a game

Task: Given QBF $\mathcal{Q}.\varphi$, determine whether $\mathcal{Q}.\varphi$ is true or false.

while $\varphi \notin \{\top, \bot\}$
  if $\mathcal{Q} = \exists x \, \mathcal{Q}'$
    Existential player chooses assignment $\mathsf{T} \in \{\top, \bot\}$ for x.
  else
    Universal player chooses assignment $\mathsf{T} \in \{\top, \bot\}$ for x.
  $\mathcal{Q} \leftarrow \mathcal{Q}', \quad \varphi \leftarrow \varphi[x = \mathsf{T}]$
if $\varphi = \top$
  return Existential player wins
else
  return Universal player wins

QBF solving as a game

Task: Given QBF $\mathcal{Q}.\varphi$, determine whether $\mathcal{Q}.\varphi$ is true or false.

while $\varphi \notin \{\top, \bot\}$
  if $\mathcal{Q} = \exists x\, \mathcal{Q}'$
    Existential player chooses assignment $\mathsf{T} \in \{\top, \bot\}$ for x.
  else
    Universal player chooses assignment $\mathsf{T} \in \{\top, \bot\}$ for x.
  $\mathcal{Q} \leftarrow \mathcal{Q}', \quad \varphi \leftarrow \varphi[x = \mathsf{T}]$
if $\varphi = \top$
  return Existential player wins
else
  return Universal player wins

- $\mathcal{Q}.\varphi$ is true iff existential player has winning strategy
- $\mathcal{Q}.\varphi$ is false iff universal player has winning strategy

QBF solving as a game

$$\forall x \exists y. (x \vee \bar{y}) \wedge (\bar{x} \vee y)$$

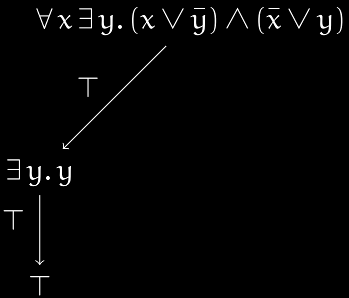
QBF solving as a game

$$\forall x \exists y. (x \vee \bar{y}) \wedge (\bar{x} \vee y)$$

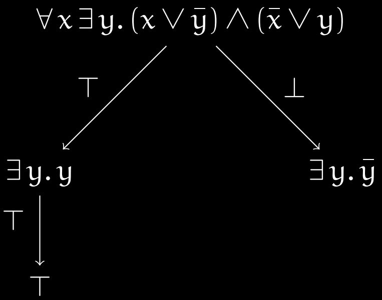
$\top$

$$\exists y. y$$

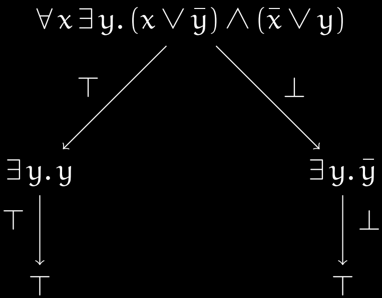
QBF solving as a game



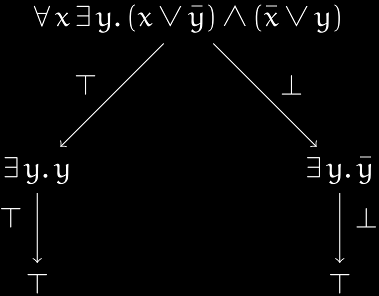
QBF solving as a game



QBF solving as a game



QBF solving as a game



Given two perfect players, one game suffices to solve a QBF.

Solving QBFs with AlphaZero

Idea:

1. Use AlphaZero to learn two (perfect) players $\pi_\exists$, $\pi_\forall$.
2. Given $\mathcal{Q}.\varphi$, let $\pi_\exists$ play against $\pi_\forall$.
3. Predict $\mathcal{Q}.\varphi$ to be true if $\pi_\exists$ wins and to be false if $\pi_\forall$ wins.

Solving QBFs with AlphaZero

Idea:

1. Use AlphaZero to learn two (perfect) players $\pi_\exists$, $\pi_\forall$.
2. Given $\mathcal{Q}.\varphi$, let $\pi_\exists$ play against $\pi_\forall$.
3. Predict $\mathcal{Q}.\varphi$ to be true if $\pi_\exists$ wins and to be false if $\pi_\forall$ wins.

Problems:

- How to represent QBFs?

Solving QBFs with AlphaZero

Idea:

1. Use AlphaZero to learn two (perfect) players $\pi_\exists$, $\pi_\forall$.
2. Given $\mathcal{Q}.\varphi$, let $\pi_\exists$ play against $\pi_\forall$.
3. Predict $\mathcal{Q}.\varphi$ to be true if $\pi_\exists$ wins and to be false if $\pi_\forall$ wins.

Problems:

- How to represent QBFs?
- QBF solving is asymmetric

How to represent QBFs?

Answer: as a graph

How to represent QBFs?

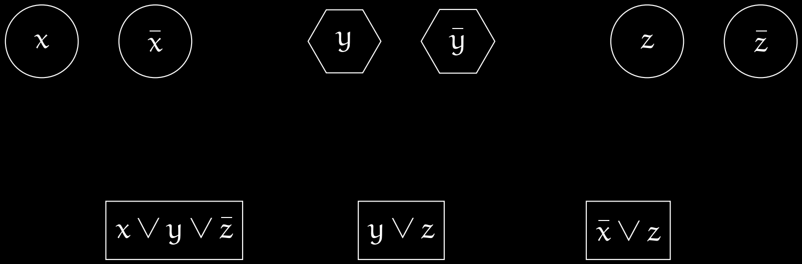
Answer: as a graph

$$\exists x \forall y \exists z. (x \vee y \vee \bar{z}) \wedge (y \vee z) \wedge (\bar{x} \vee z)$$

How to represent QBFs?

Answer: as a graph

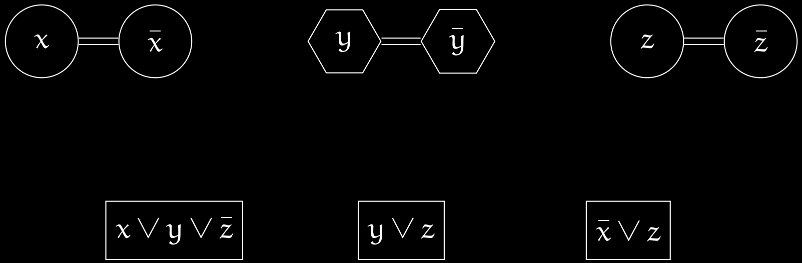
$$\exists x \forall y \exists z. (x \vee y \vee \bar{z}) \wedge (y \vee z) \wedge (\bar{x} \vee z)$$



How to represent QBFs?

Answer: as a graph

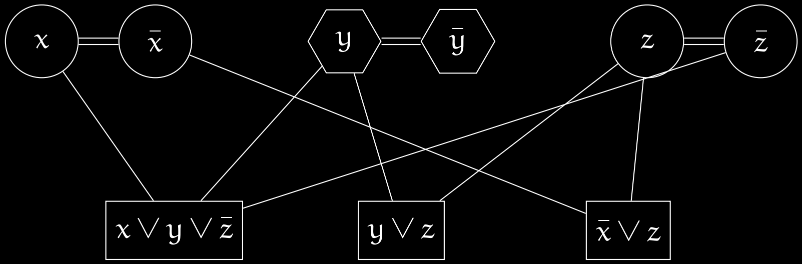
$$\exists x \forall y \exists z. (x \vee y \vee \bar{z}) \wedge (y \vee z) \wedge (\bar{x} \vee z)$$



How to represent QBFs?

Answer: as a graph

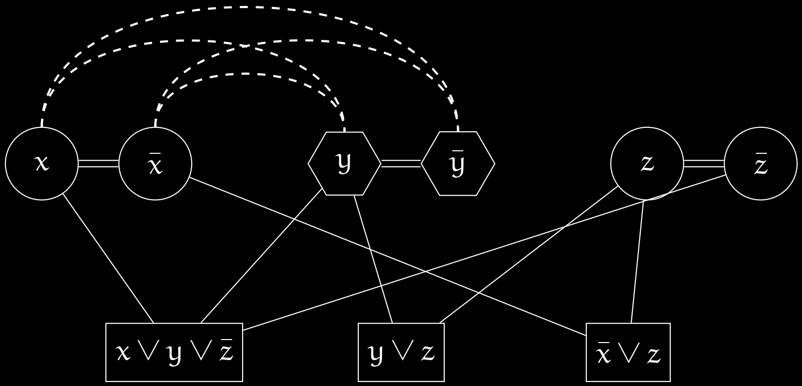
$$\exists x \forall y \exists z. (x \vee y \vee \bar{z}) \wedge (y \vee z) \wedge (\bar{x} \vee z)$$



How to represent QBFs?

Answer: as a graph

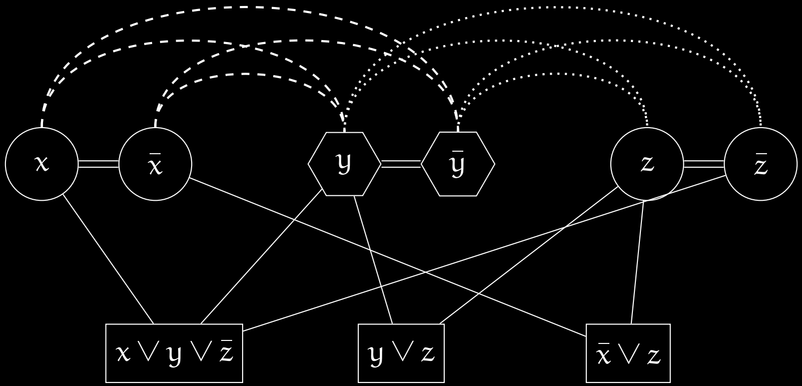
$$\exists x \forall y \exists z. (x \vee y \vee \bar{z}) \wedge (y \vee z) \wedge (\bar{x} \vee z)$$



How to represent QBFs?

Answer: as a graph

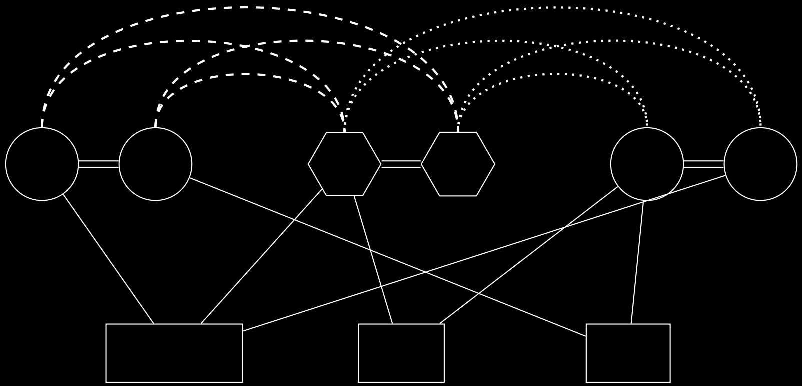
$$\exists x \forall y \exists z. (x \vee y \vee \bar{z}) \wedge (y \vee z) \wedge (\bar{x} \vee z)$$



How to represent QBFs?

Answer: as a graph

$$\exists x \forall y \exists z. (x \vee y \vee \bar{z}) \wedge (y \vee z) \wedge (\bar{x} \vee z)$$



How to represent QBFs?

Process QBF graphs with a gated graph neural network.

1: Initialize $h_v^{(0)}$ for each vertex v with a vector in $\mathbb{R}^N$

2: **for** $t \leftarrow 0, \ldots, T-1$ **do**

3: $\quad m_v^{(t+1)} \leftarrow \sum_{w \in N(v)} A_{e_{vw}} h_w^{(t)}$ for each vertex v

4: $\quad h_v^{(t+1)} \leftarrow \mathrm{GRU}(h_v^{(t)}, m_v^{(t+1)})$ for each vertex v

5: $p_\pi \leftarrow \sum_{v \in V} f_1(h_v^{(T)}, h_v^{(0)}), \quad v_\pi \leftarrow \sum_{v \in V} f_2(h_v^{(T)}, h_v^{(0)})$,
with neural networks f_1, f_2

6: **return** $\mathrm{softmax}(p_\pi), \tanh(v_\pi)$

How to represent QBFs?

Process QBF graphs with a gated graph neural network.

Initialize $h_v^{(0)}$ for each vertex v with a vector in $\mathbb{R}^N$

for $t \leftarrow 0, \ldots, T-1$ **do**

  $m_v^{(t+1)} \leftarrow \sum_{w \in N(v)} A_{e_{vw}} h_w^{(t)}$ for each vertex v

  $h_v^{(t+1)} \leftarrow \mathrm{GRU}(h_v^{(t)}, m_v^{(t+1)})$ for each vertex v

$p_\pi \leftarrow \sum_{v \in V} f_1(h_v^{(T)}, h_v^{(0)}), \quad v_\pi \leftarrow \sum_{v \in V} f_2(h_v^{(T)}, h_v^{(0)})$,
with neural networks f_1, f_2

return $\mathrm{softmax}(p_\pi), \tanh(v_\pi)$

QBF solving is asymmetric

- Players have different goals
- Not necessarily evenly alternating

QBF solving is asymmetric

- Players have different goals
- Not necessarily evenly alternating

Solution:

- Extend AlphaZero to train two networks with different goals simultaneously
- Take more care during the MCTS

Training by 2-player-self-play

Initialize players with random parameters $\pi_\exists, \pi_\forall$

for $e \leftarrow 1, \ldots, E$ **do**

    Generate data by self-play games

    Update parameters $\pi_\exists$

    Compare new player to best existential player

    Generate data by self-play games

    Update parameters $\pi_\forall$

    Compare new player to best universal player

return $\pi_\exists, \pi_\forall$

EXPERIMENTAL RESULTS

Training

Starting point

`https://github.com/suragnair/alpha-zero-general`

Training

Starting point

`https://github.com/suragnair/alpha-zero-general`

After doing the adaptations, we trained two players for

- 16 epochs with
- 20 self-play games each
- using 40 MCTS iterations per move

Training

Starting point

`https://github.com/suragnair/alpha-zero-general`

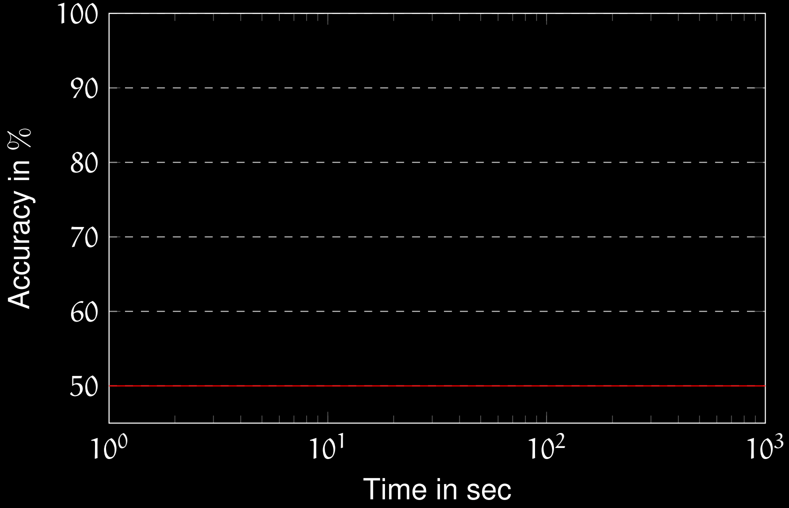
After doing the adaptations, we trained two players for

- 16 epochs with
- 20 self-play games each
- using 40 MCTS iterations per move

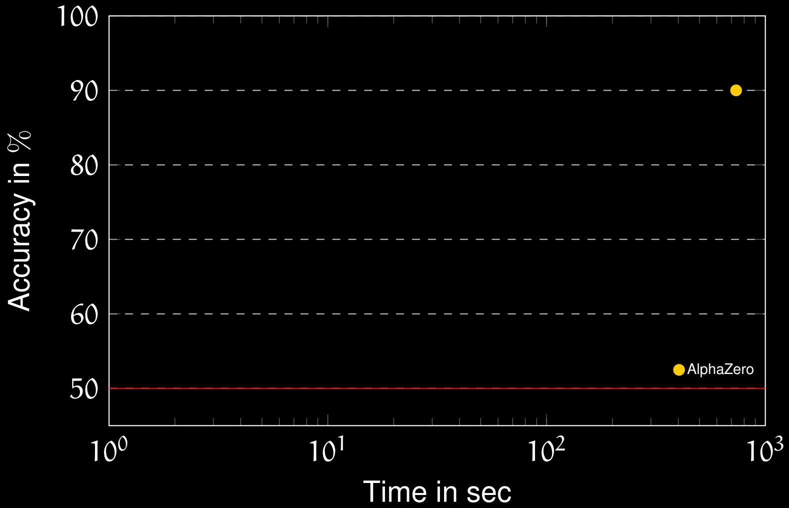
on a training set consisting of

- 100 random QBFs (50 true, 50 false) with
- 10 – 40 variables and
- 5 – 20 clauses

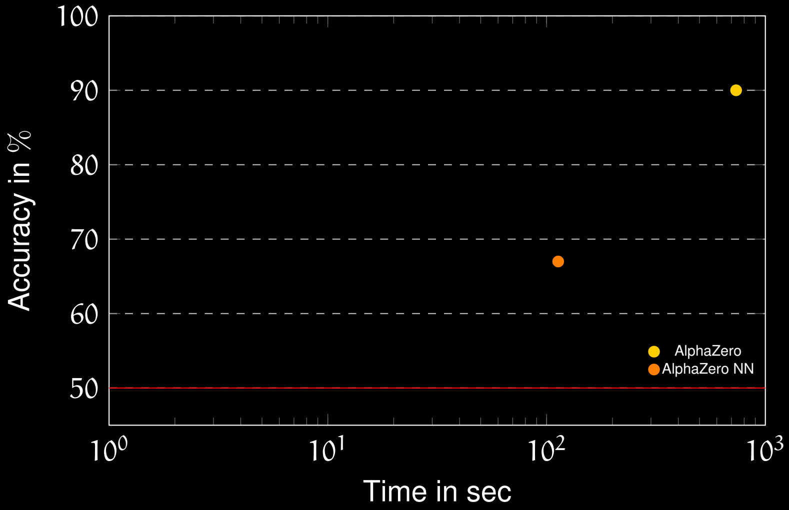
Accuracy on random test data



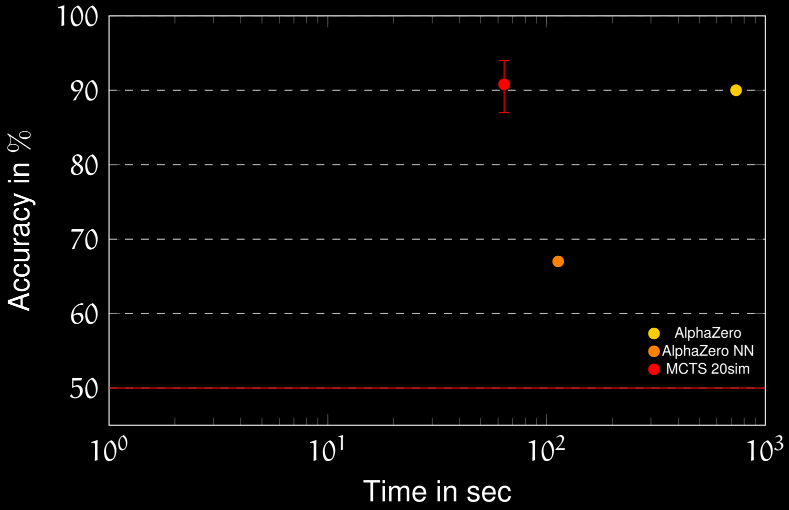
Accuracy on random test data



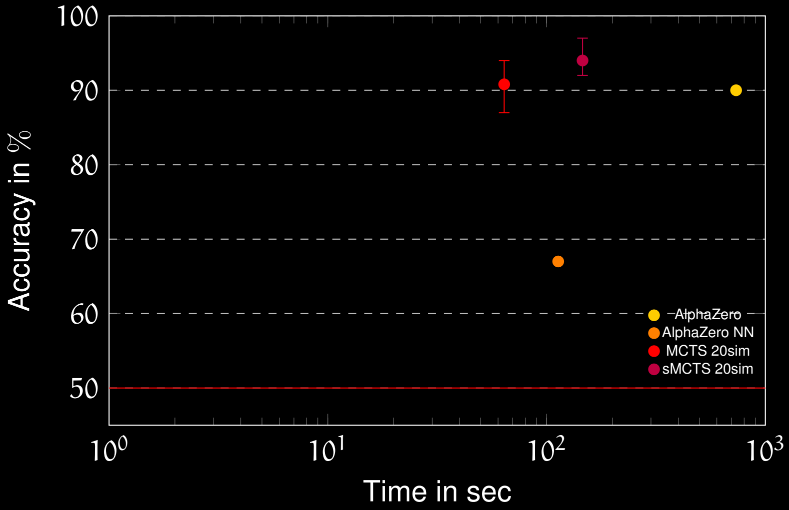
Accuracy on random test data



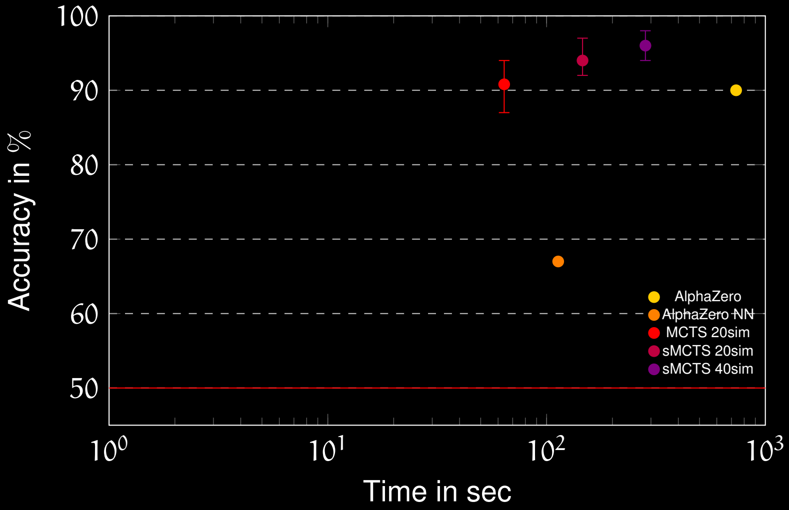
Accuracy on random test data



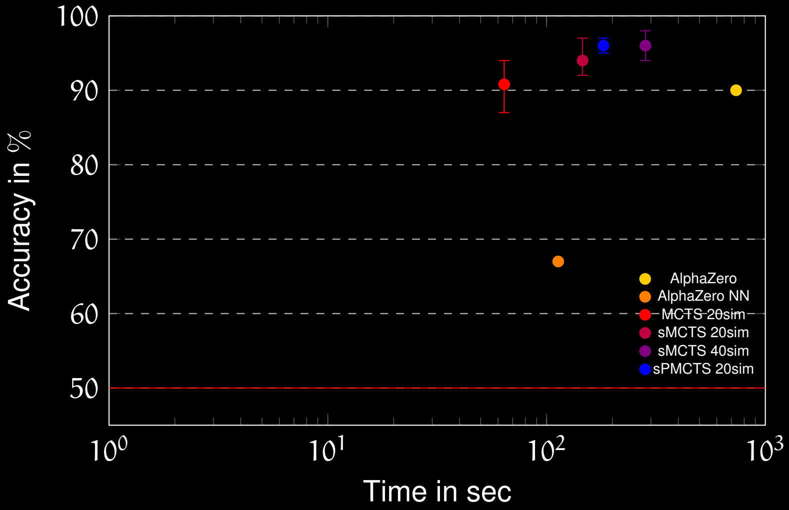
Accuracy on random test data



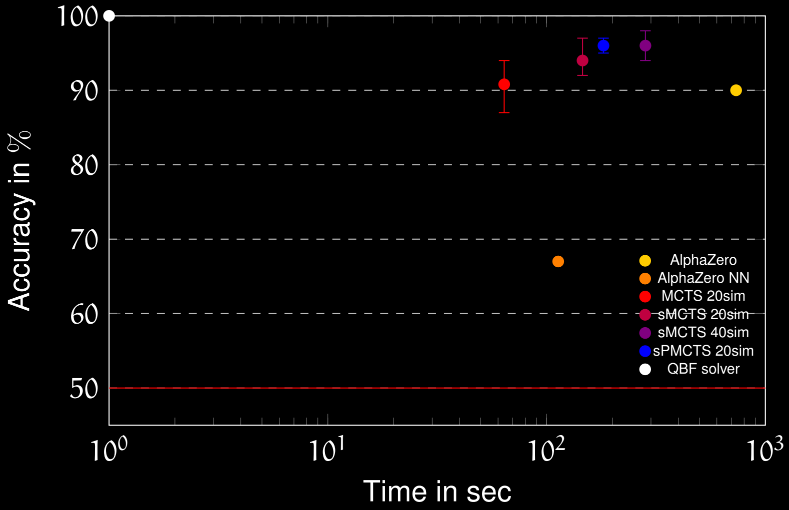
Accuracy on random test data



Accuracy on random test data



Accuracy on random test data



Conclusion & Outlook

- AlphaZero can be adapted to successfully solve QBFs

Conclusion & Outlook

- AlphaZero can be adapted to successfully solve QBFs
- In fact, MCTS can be used to successfully solve QBFs

Conclusion & Outlook

- AlphaZero can be adapted to successfully solve QBFs
- In fact, MCTS can be used to successfully solve QBFs
- Existential player seems to be better than universal player

Conclusion & Outlook

- AlphaZero can be adapted to successfully solve QBFs
- In fact, MCTS can be used to successfully solve QBFs
- Existential player seems to be better than universal player
- Try to find good predictor heuristics

Conclusion & Outlook

- AlphaZero can be adapted to successfully solve QBFs
- In fact, MCTS can be used to successfully solve QBFs
- Existential player seems to be better than universal player
- Try to find good predictor heuristics
- Try to integrate MCTS into QBF solver (e.g. to find good partial assigments)