Monte-Carlo Planning in Large POMDPs

Paper by
David Silver, Joel Veness (2010)

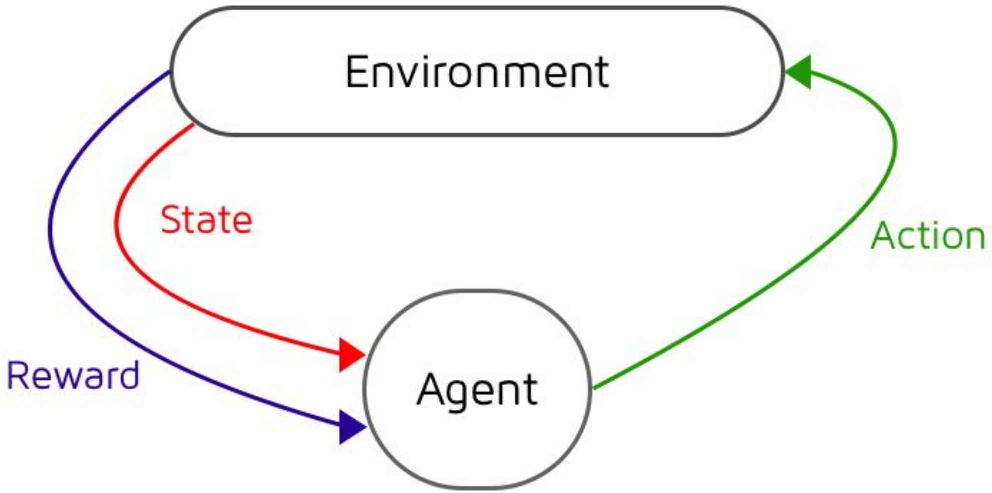
Presented by
Yining (Annie) Zhang
Anqi (Joyce) Yang
Oct 18, 2019

Outline

- Background
  - MDP and POMDP
  - Monte-Carlo Tree Search
- Related Work
- POMCP (Extending MCTS to POMDP)
  - Belief State Update with Particle Filters
  - Full algorithm

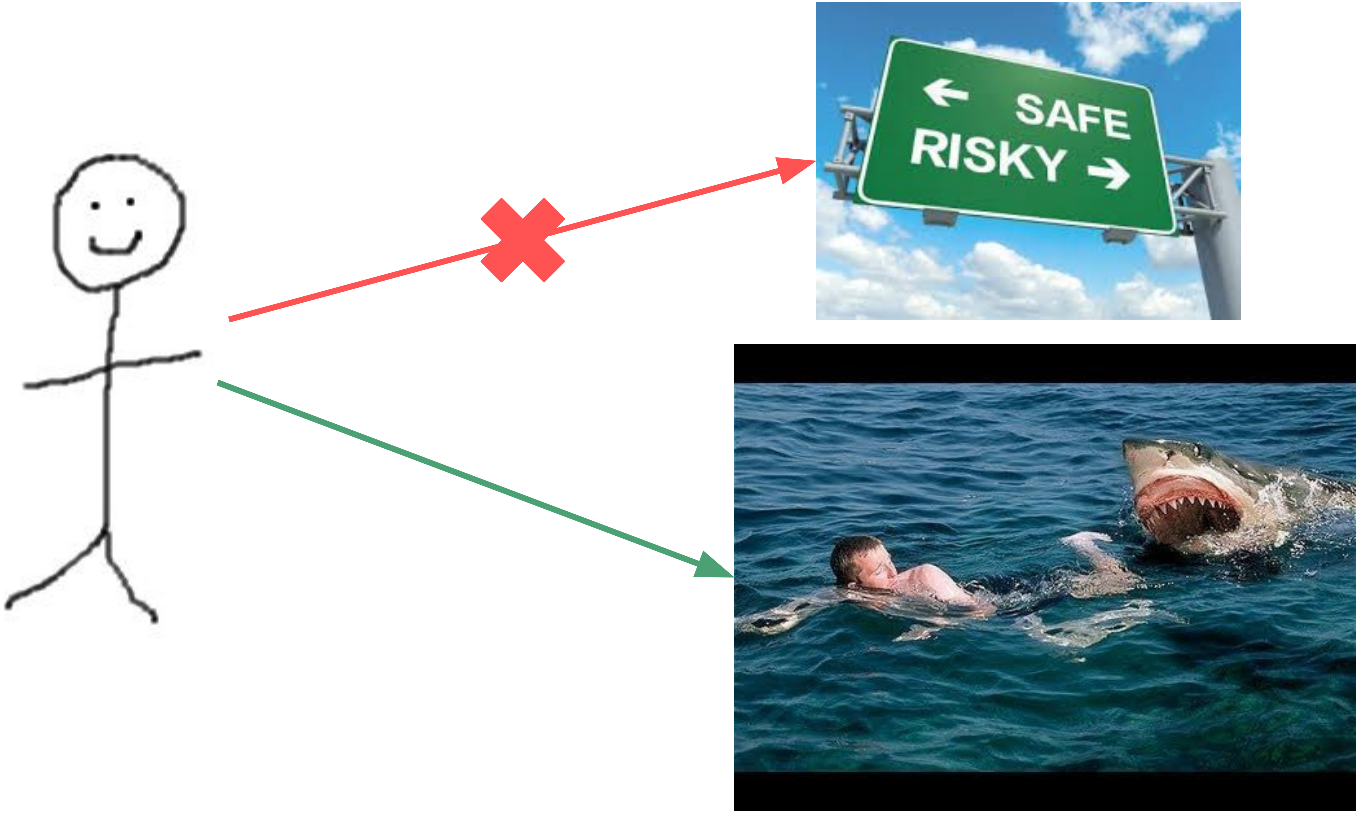
MDP and POMDP

Markov Decision Process (MDP)

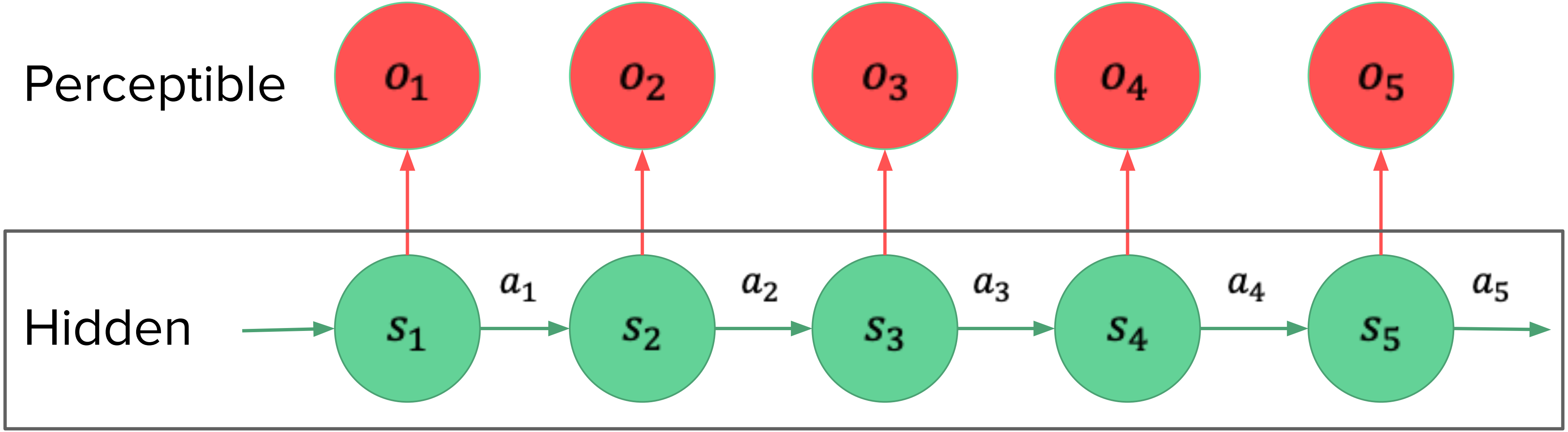


Source: Nervana Deep Reinforcement Learning with Open AI Gym

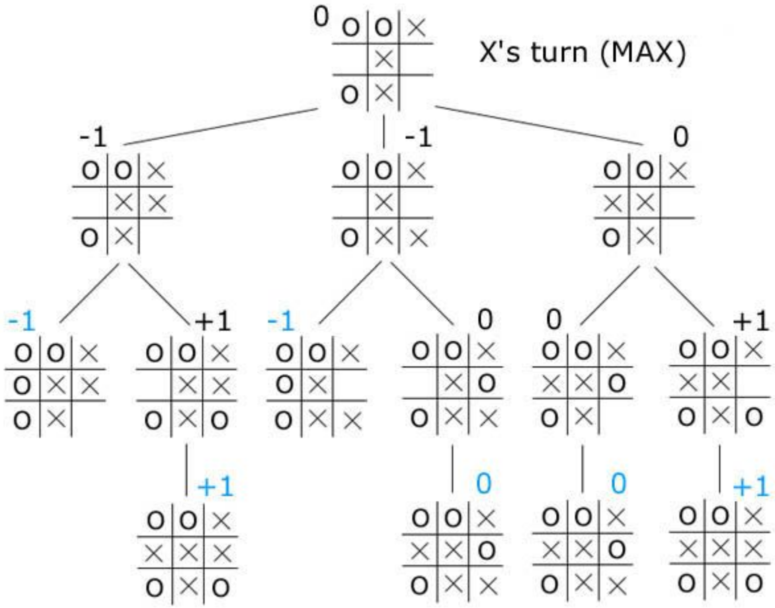
Partially Observable Markov Decision Process (POMDP)



Partially Observable Markov Decision Process (POMDP)



MDP vs POMDP Examples



MDP Example: Tic-Tac-Toe

Source: https://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html

POMDP Example: Poker

Source: Cassius Marcellus Coolidge "Dogs Playing Poker"

Partially Observable Markov Decision Process (POMDP)

- (S, A, O, T, Z, R)
- O: Set of Observations perceptible by agent
- Z: Observation Function

$$\mathcal{Z}^{a}_{s'o} = Pr(o_{t+1} = o | s_{t+1} = s', a_t = a)$$

Partially Observable Markov Decision Process (POMDP)

- (S, A, O, T, Z, R)

- O: Set of Observations perceptible by agent

- Z: Observation Function

$$\mathcal{Z}^{a}_{s'o} = Pr(o_{t+1} = o | s_{t+1} = s', a_t = a)$$

- History

$$h_t = \{a_1, o_1, ..., a_t, o_t\} \text{ or } h_t a_{t+1} = \{a_1, o_1, ..., a_t, o_t, a_{t+1}\}$$

MDP vs POMDP

	MDP	POMDP
Definition	$(\mathcal{S}, \mathcal{A}, T, R)$	$(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R)$
History	N.A.	$h_t = \{a_1, o_1, \ldots, a_t, o_t\}$
Policy	$\pi(s, a) = Pr(a_{t+1} = a \mid s_t = s)$	$\pi(h, a) = Pr(a_{t+1} = a \mid h_t = h)$
Value Function	$V_s^\pi = \mathbb{E}_\pi[R_t \mid s_t = s]$	$V_h^\pi = \mathbb{E}_\pi[R_t \mid h_t = h]$

MDP vs POMDP

	MDP	POMDP
Definition	$(\mathcal{S}, \mathcal{A}, T, R)$	$(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R)$
History	N.A.	$h_t = \{a_1, o_1, \ldots, a_t, o_t\}$
Policy	$\pi(s, a) = Pr(a_{t+1} = a \mid s_t = s)$	$\pi(h, a) = Pr(a_{t+1} = a \mid h_t = h)$
Value Function	$V_s^\pi = \mathbb{E}_\pi[R_t \mid s_t = s]$	$V_h^\pi = \mathbb{E}_\pi[R_t \mid h_t = h]$

Storing the entire history can be memory-intensive!

POMDP Belief State

Belief State

$$\mathcal{B}(s,h) = Pr(s_t = s | h_t = h)$$

POMDP Belief State

Belief State

$$\mathcal{B}(s,h) = Pr(s_t = s | h_t = h)$$

Belief State Update

$$\mathcal{B}_t(s') = \tau(\mathcal{B}_{t-1}, a_{t-1}, o_{t-1})$$

POMDP Belief State

Belief State

$$\mathcal{B}(s, h) = Pr(s_t = s | h_t = h)$$

Belief State Update

$$\mathcal{B}_t(s') = \tau(\mathcal{B}_{t-1}, a_{t-1}, o_{t-1})$$

Policy and Value Function defined on Belief State

$$\pi(b, a) = Pr(a_{t+1} = a | b_t = b)$$

$$V_b^\pi = \mathbb{E}[R_t | b_t = b]$$

Solving POMDPs

- POMDP as Belief-State-MDP

Solving POMDPs

- POMDP as Belief-State-MDP

- BUT

  Belief states are in a continuous space!! **Uncountable**

Solving POMDPs

- POMDP as Belief-State-MDP

- BUT

  Belief states are in a continuous space!! **Uncountable**

- Two approaches, but both are inefficient
  - Discretize belief state space: suffer from curse of dimensionality

  - Exact methods: model belief state space as hyperplanes: intractable

Other Solutions to POMDPs

- Other Online Approaches (point value iteration, etc.)

  Require explicit model, curse of dimensionality for large state space

  Survey (2008): https://www.aaai.org/Papers/JAIR/Vol32/JAIR-3217.pdf

Monte-Carlo Tree Search

Monte-Carlo Tree Search - Overview

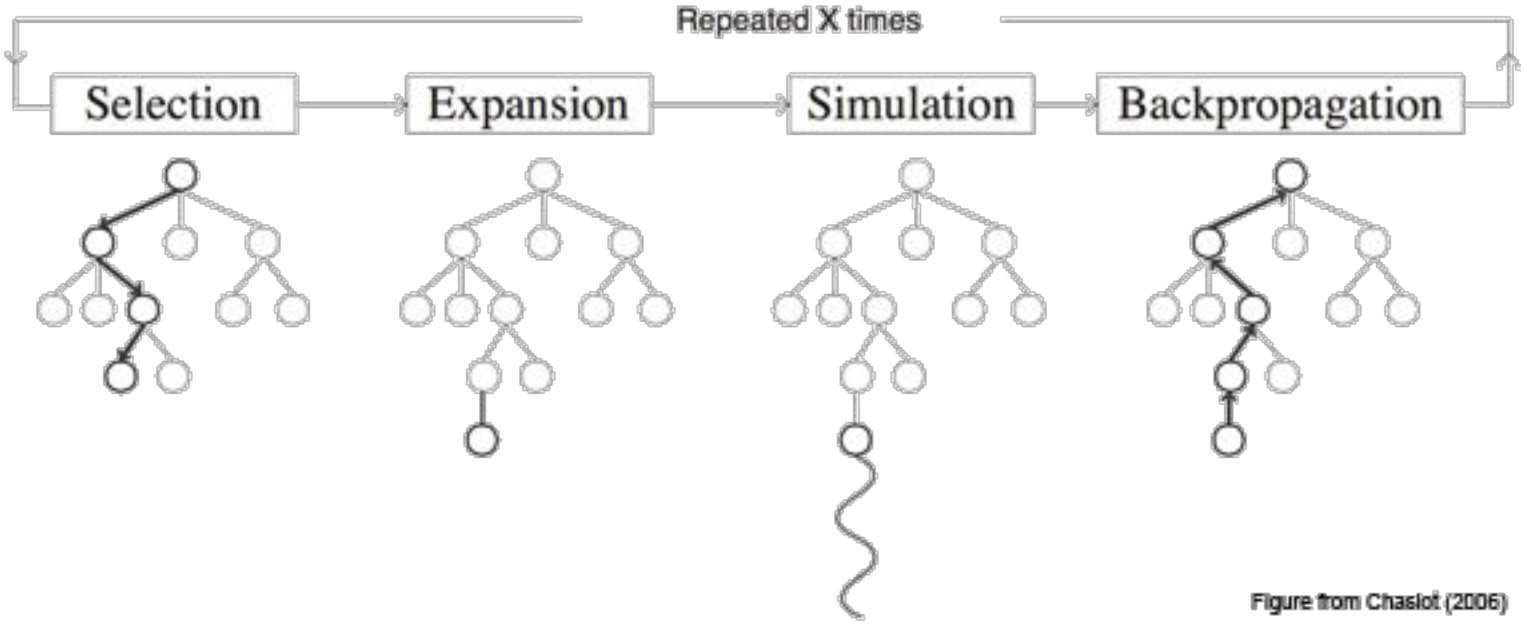
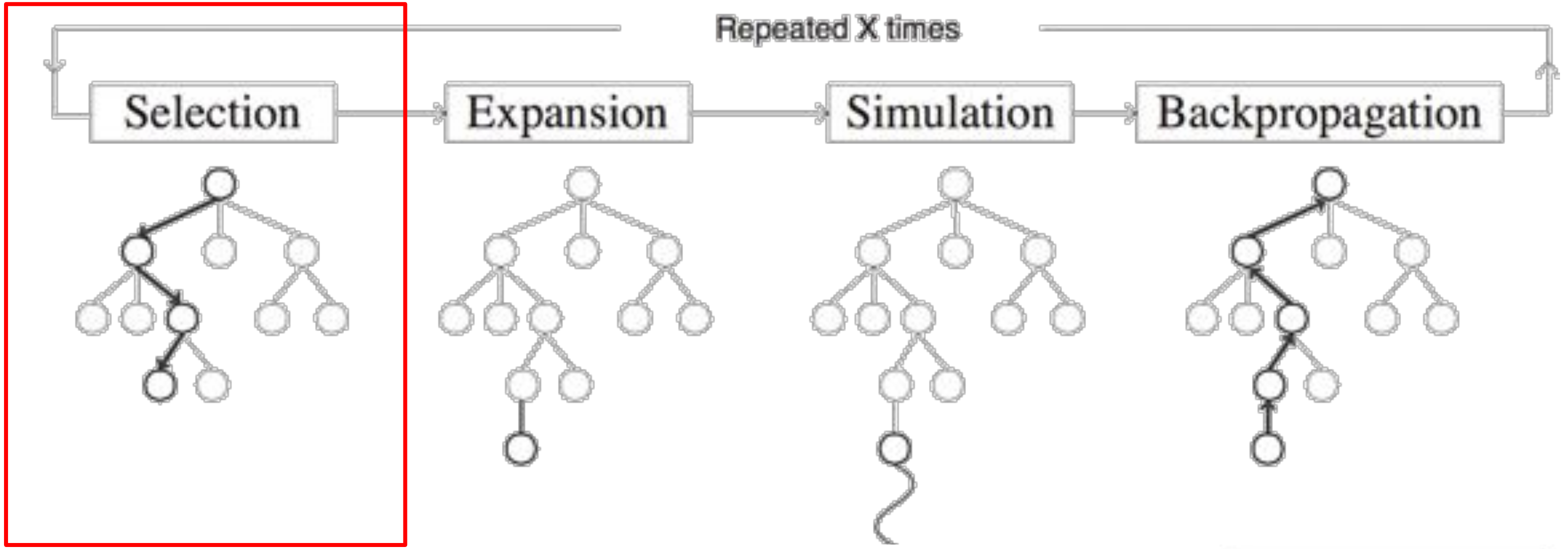


Figure from Chaslot (2006)

Monte-Carlo Tree Search - Selection



$$Q^{\oplus}(s,a) = Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$$

Monte-Carlo Tree Search - Expansion

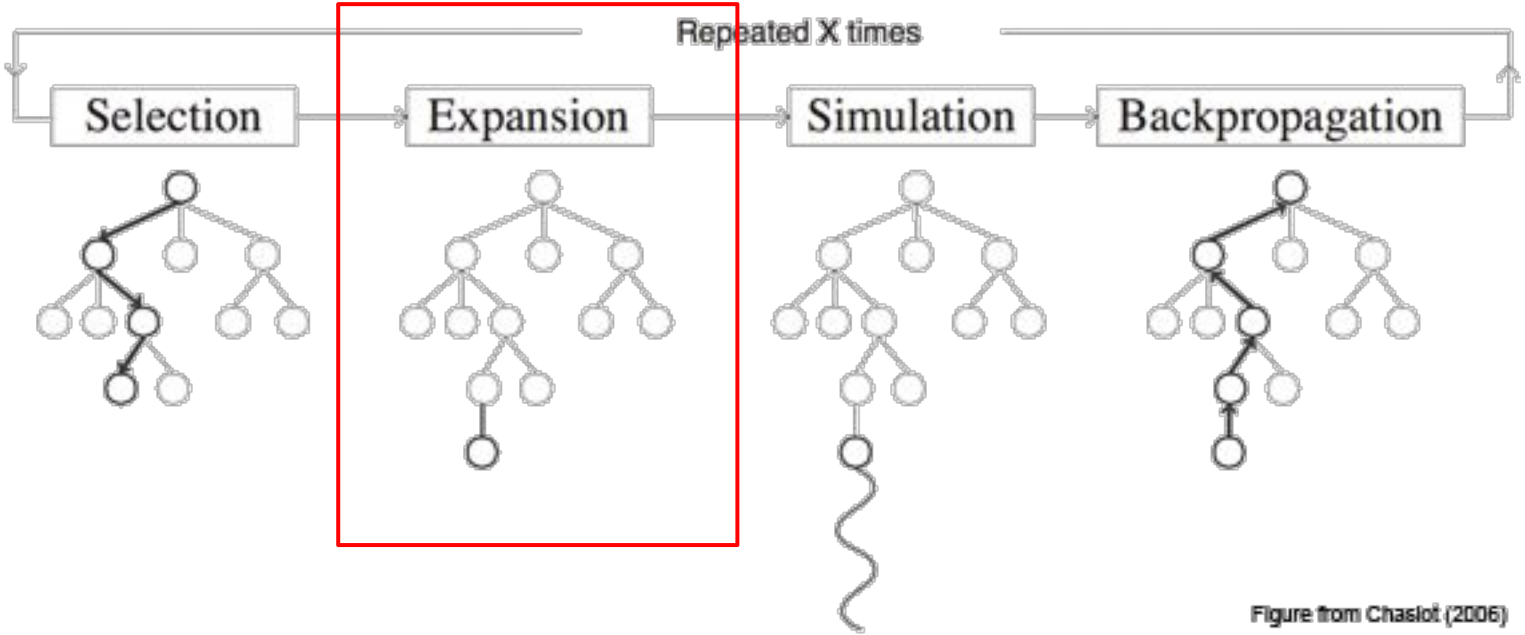


Figure from Chaslot (2006)

Monte-Carlo Tree Search - Simulation

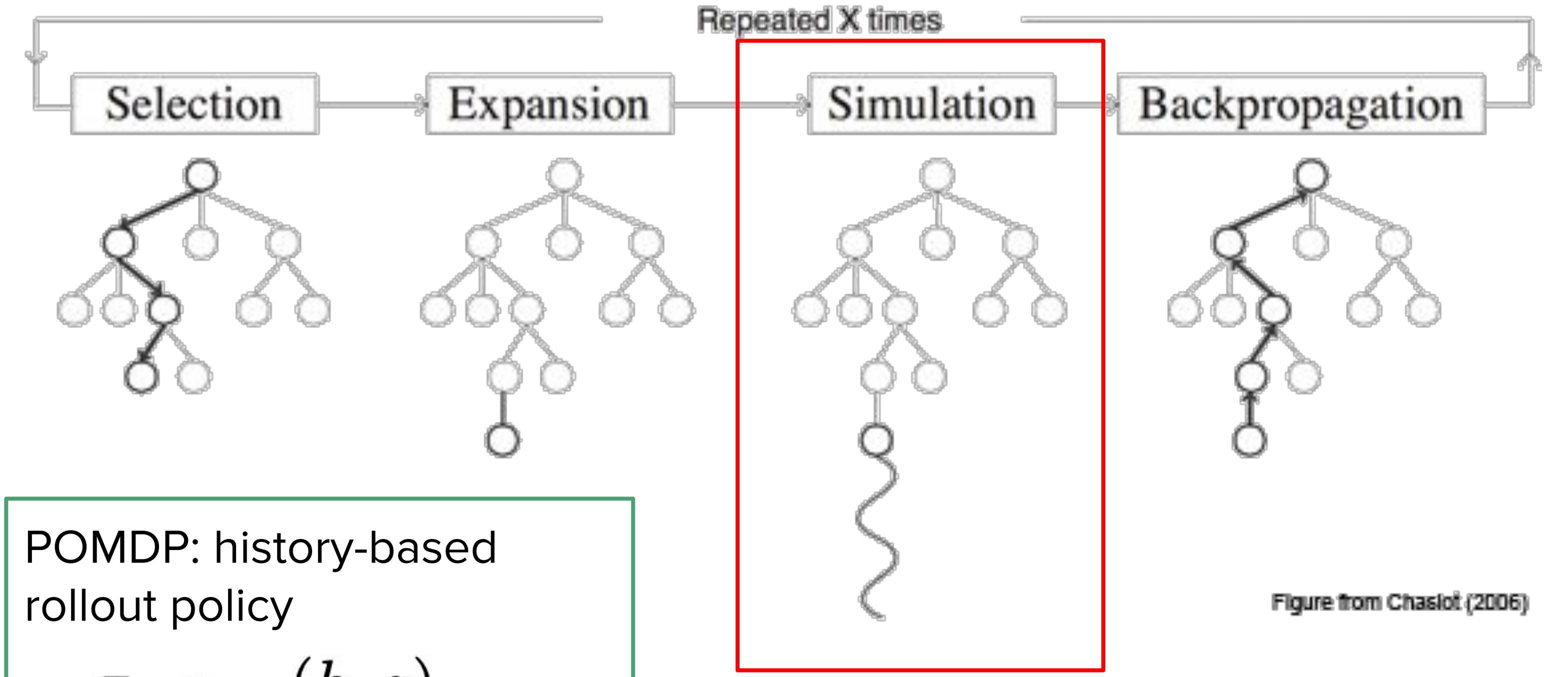


Figure from Chaslot (2006)

POMDP: history-based rollout policy

$$\pi_{rollout}(h, a)$$

Monte-Carlo Tree Search - Selection

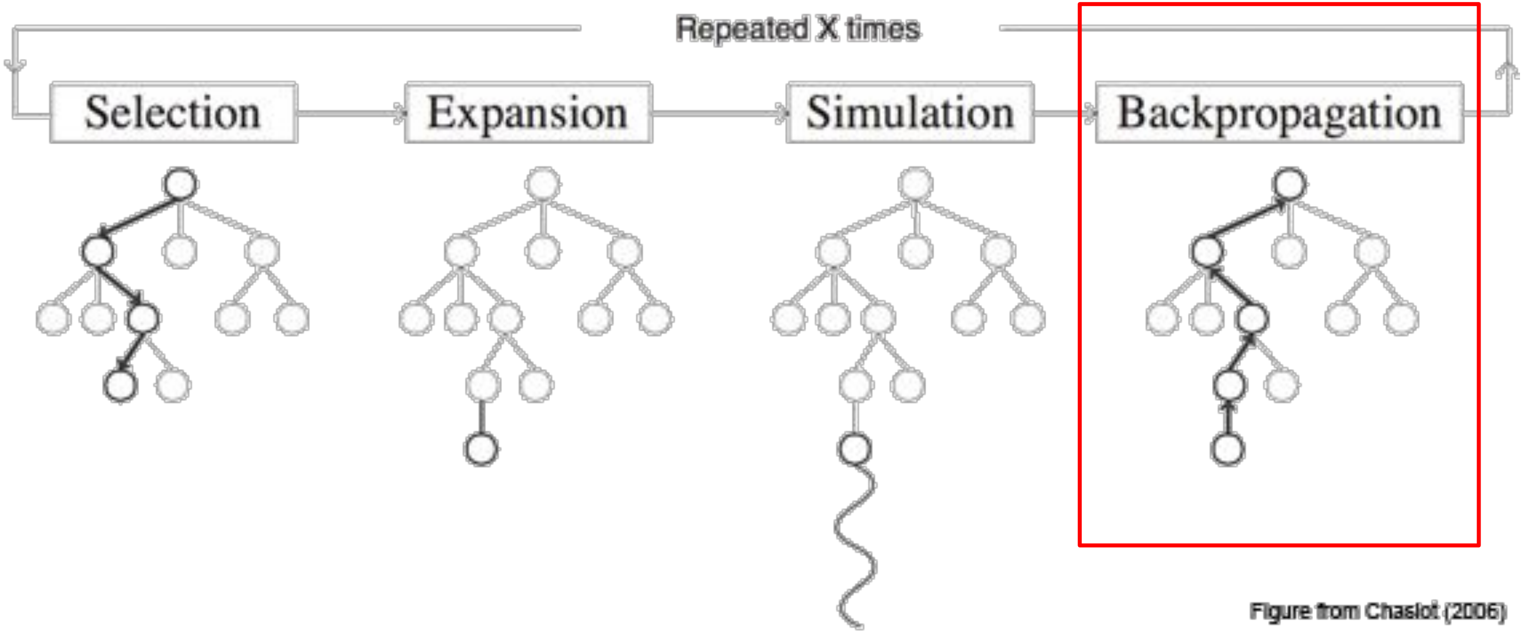


Figure from Chaslot (2006)

Related Work

Related Work

Ross et al. Online planning algorithms for pomdps. 2008.

Couloum. Efficient selectivity and backup operators in Monte-Carlo tree search. 2006.

Sunberg et al. Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces. 2018.

Lee et al. Monte-Carlo Tree Search for Constrained POMDPs. 2018.

Igl et al. Deep Variational Reinforcement Learning for POMDPs. 2018.

Lee et al. Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model. 2019.

POMCP - Partially Observable Monte-Carlo Planning

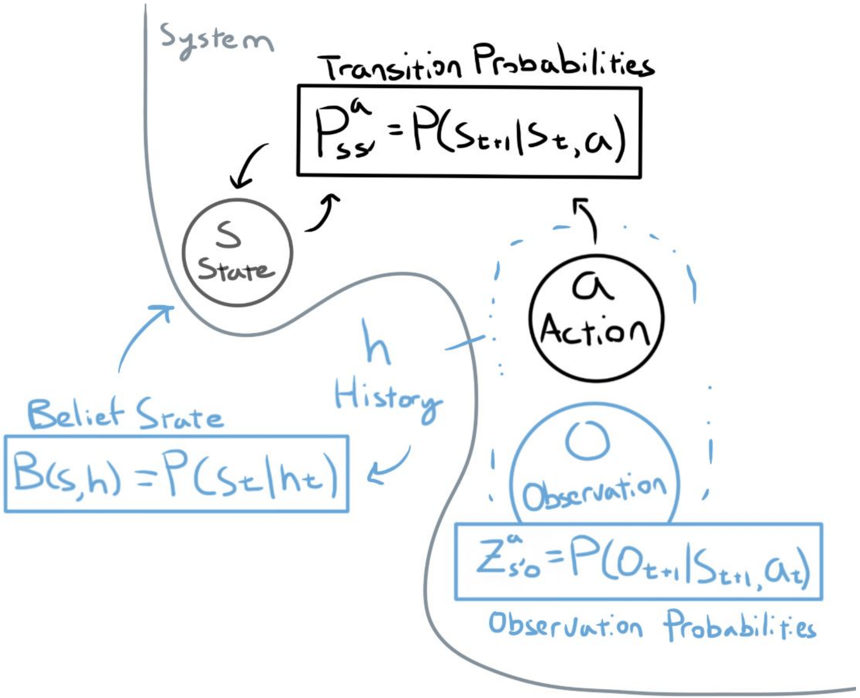
Paper by

David Silver, Joel Veness (2010)

Monte Carlo Tree Search for POMDPs?

How do we represent Monte Carlo Tree without access to states?

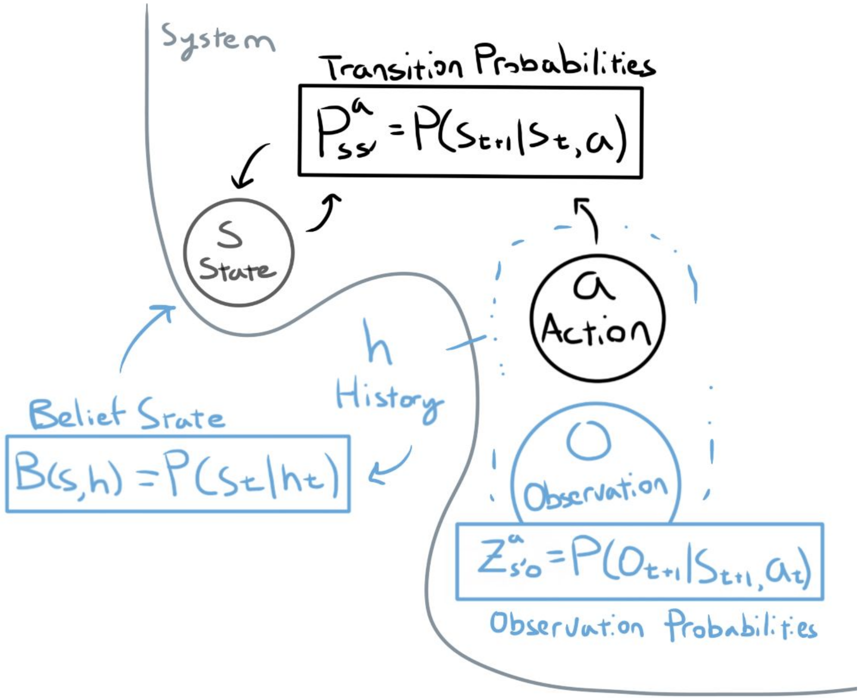
1. History
2. Belief State - P(s|h)



Monte Carlo Tree Search for POMDPs?

How do we represent Monte Carlo Tree without access to states?

1. History - Needs **History based simulator**
2. Belief State - P(s|h) - **Expensive**



Updating Belief State

Exact Update - Bayes' rule:

Sum over possible **current states**.

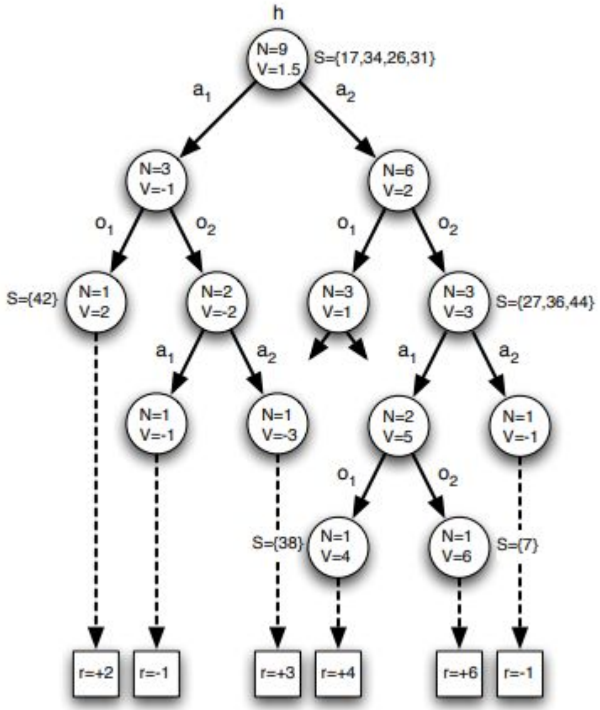
$$\mathcal{B}(s', hao) = \frac{\sum_{s \in \mathcal{S}} \mathcal{Z}^a_{s'o} \mathcal{P}^a_{ss'} \mathcal{B}(s,h)}{\sum_{s \in \mathcal{S}} \sum_{s'' \in \mathcal{S}} \mathcal{Z}^a_{s''o} \mathcal{P}^a_{ss''} \mathcal{B}(s,h)}$$

For each possible **next state**.

Normalize

Infeasible for large State spaces!

Partially Observable Monte-Carlo Planning



Use **history** nodes.

Use **state simulator**

$$(s_{t+1}, o_{t+1}, r_{t+1}) \sim \mathcal{G}(s_t, a_t)$$

To approximate **Belief State P(s|h)**.

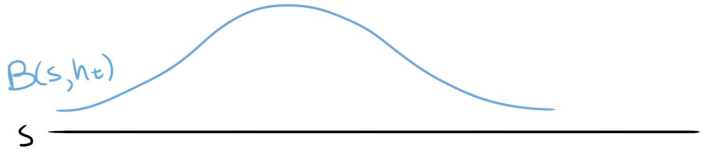
Belief State Update

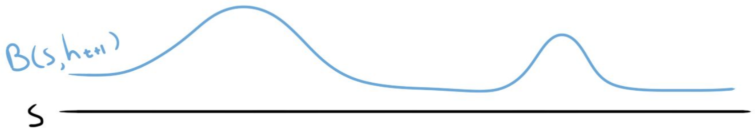
Approximate Update - Particle Filter

If we take **action a** and see **observation o**, what's the new Belief State **P(s|hao)**?

$H_t = h$

$H_{t+1} = hao$



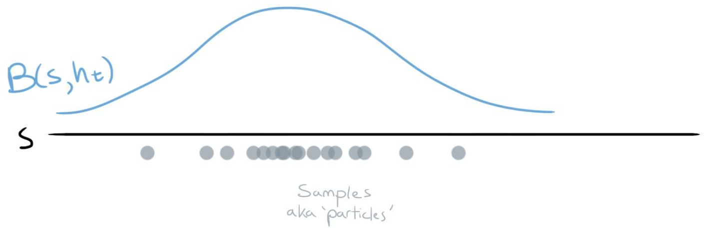


Belief State Update

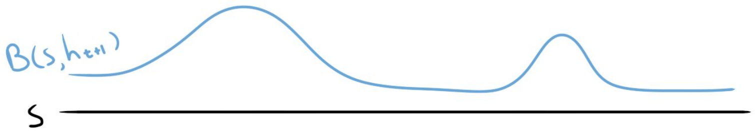
Approximate Update - Particle Filter

If we take **action a** and see **observation o**, what's the new Belief State **P(s|hao)**?

$H_t = h$



$H_{t+1} = hao$

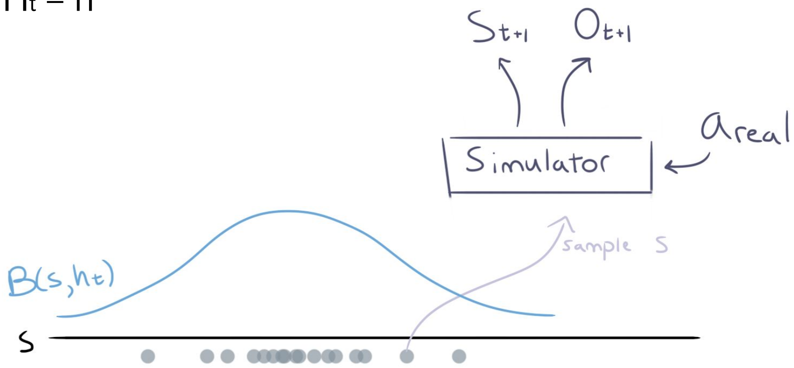


Belief State Update

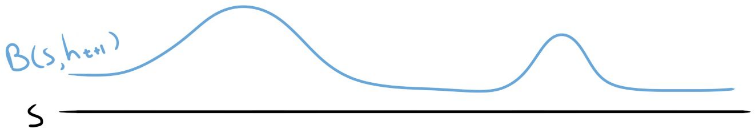
Approximate Update - Particle Filter

If we take **action a** and see **observation o**, what's the new Belief State **P(s|hao)**?

$H_t = h$



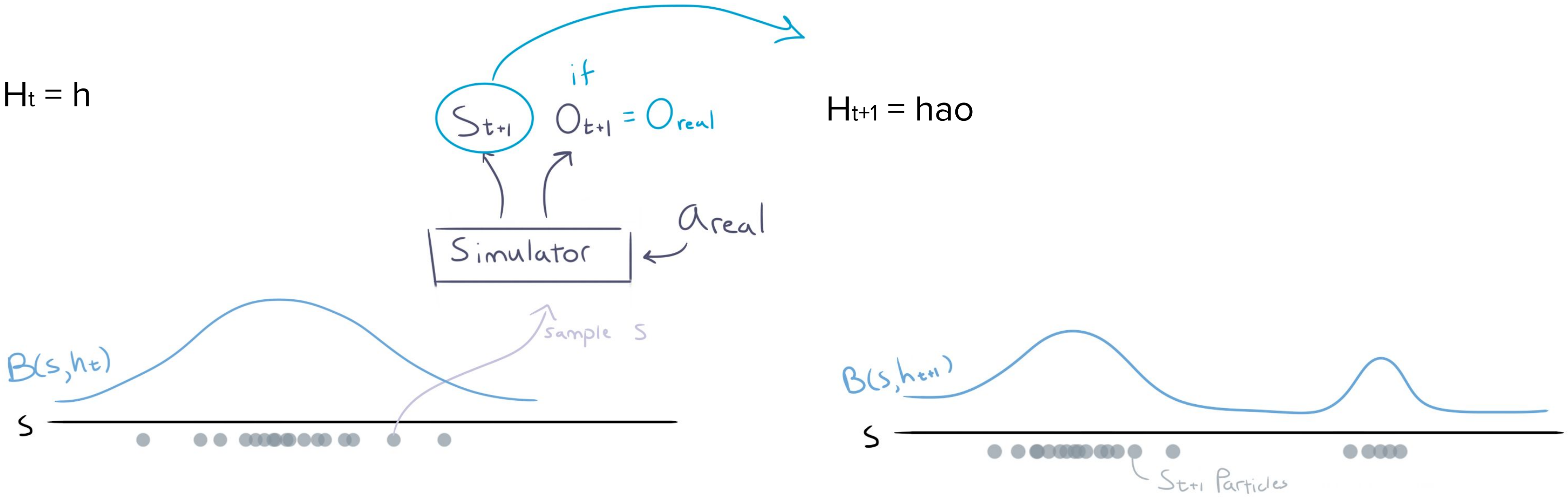
$H_{t+1} = hao$



Belief State Update

Approximate Update - Particle Filter

Take real **action a** and see real **observation o**, what’s new Belief State **B(s|hao)**?



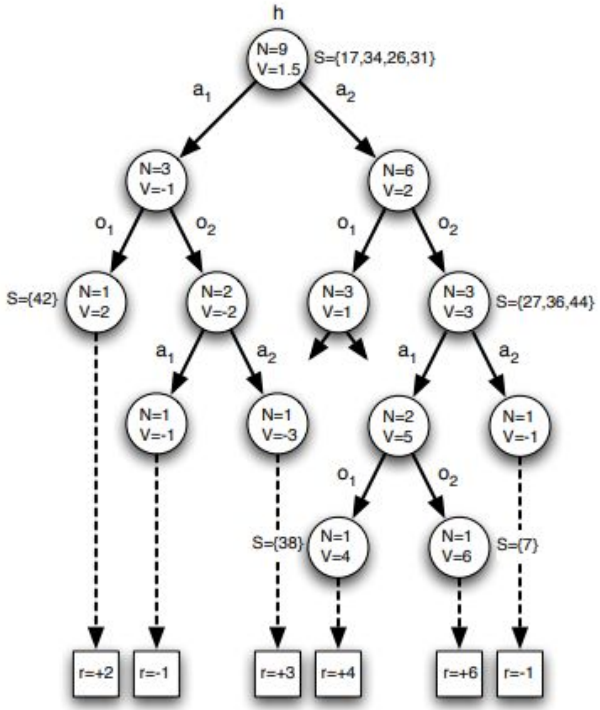
POMCP

Combine Monte Carlo Tree Search and Particle Filter Belief Updates and share the simulations.

Each node stores:

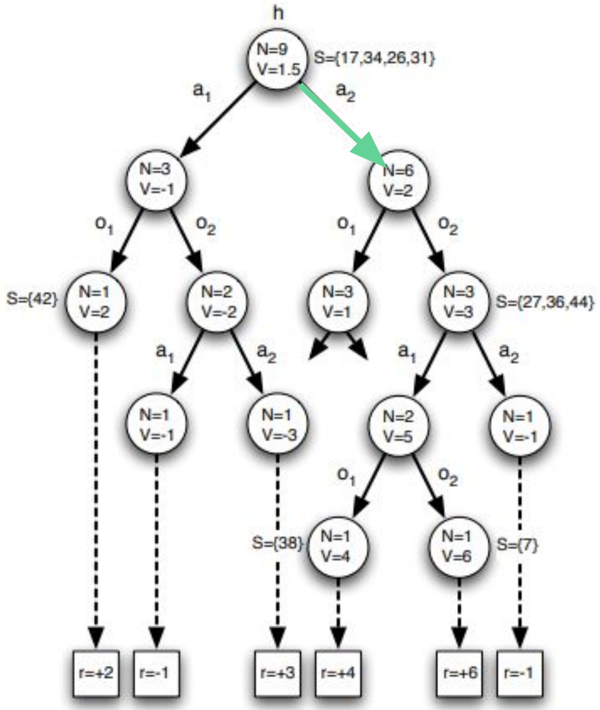
- N
- V
- B(s|h)

Partially Observable Monte-Carlo Planning



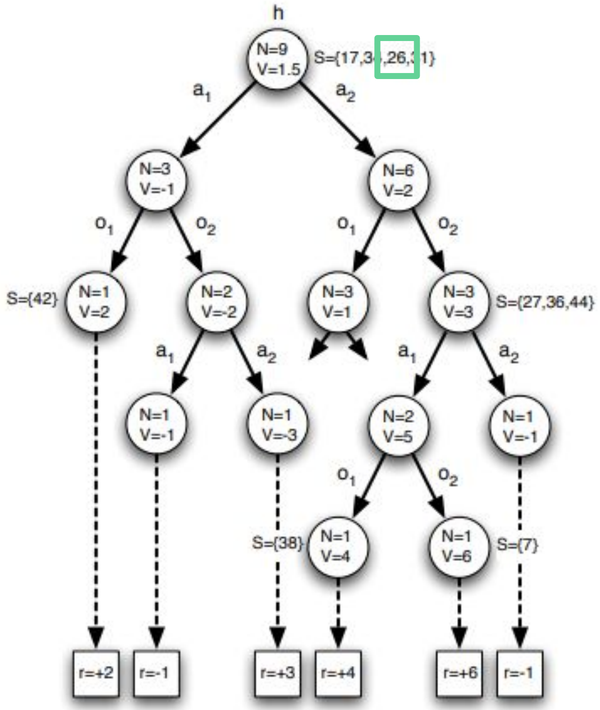
1. Selection (UCT)
   - ➔ **(PO-)UCT**
   - ➔ Sample particle **s** from **B(s|h)**
   - ➔ Black box **Simulator for s' and o'**
2. Expansion
3. Simulation (Rollout)
4. Backprop
   - ➔ **Update** each **B(s'|hao)** by adding corresponding simulated **s'** to particle set.

Partially Observable Monte-Carlo Planning



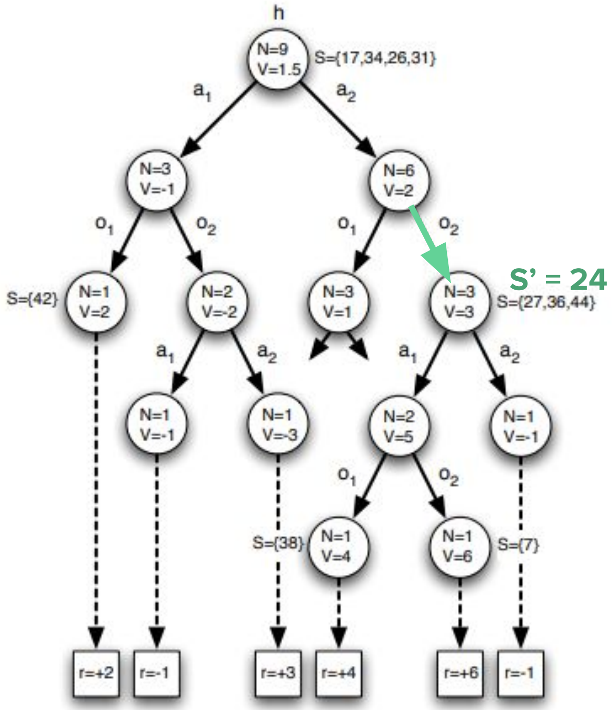
1. Selection (UCT)
   - ➜ **(PO-)UCT**
   - ➜ Sample particle **s** from **B(s|h)**
   - ➜ Black box **Simulator for s' and o'**
2. Expansion
3. Simulation (Rollout)
4. Backprop
   - ➜ **Update** each **B(s'|hao)** by adding corresponding simulated **s'** to particle set.

Partially Observable Monte-Carlo Planning



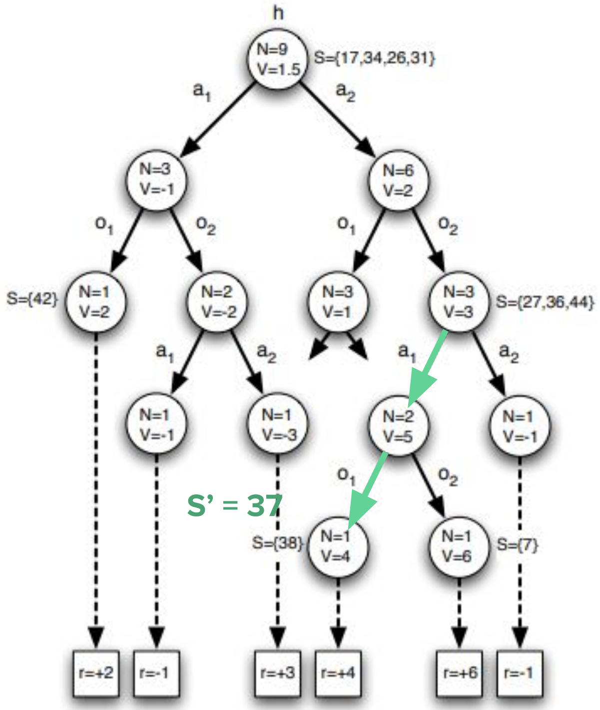
1. Selection (UCT)
   - → **(PO-)UCT**
   - → Sample particle **s** from **B(s|h)**
   - → Black box **Simulator for s' and o'**
2. Expansion
3. Simulation (Rollout)
4. Backprop
   - → **Update** each **B(s'|hao)** by adding corresponding simulated **s'** to particle set.

Partially Observable Monte-Carlo Planning



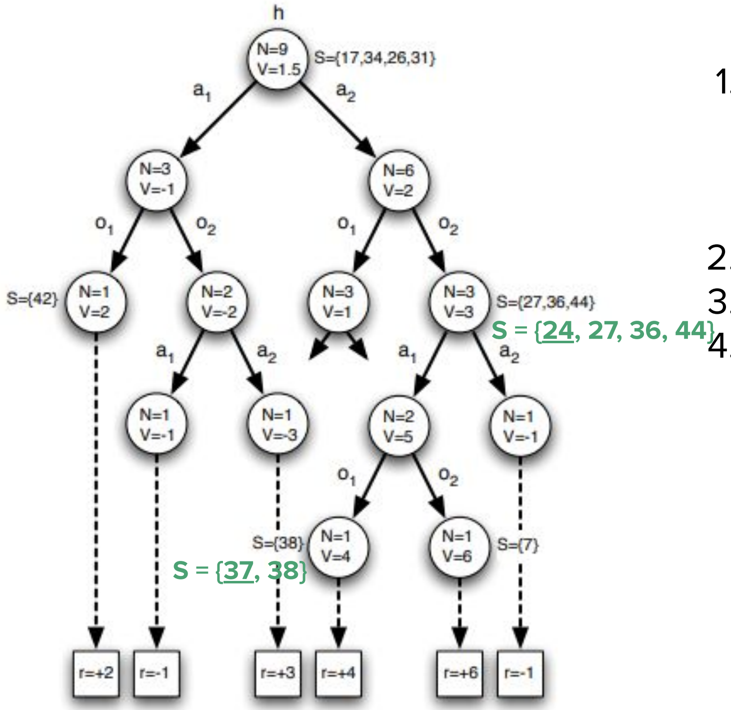
1. Selection (UCT)
   - ➜ **(PO-)UCT**
   - ➜ Sample particle **s** from **B(s|h)**
   - ➜ Black box **Simulator for s' and o'**
2. Expansion
3. Simulation (Rollout)
4. Backprop
   - ➜ **Update** each **B(s'|hao)** by adding corresponding simulated **s'** to particle set.

Partially Observable Monte-Carlo Planning



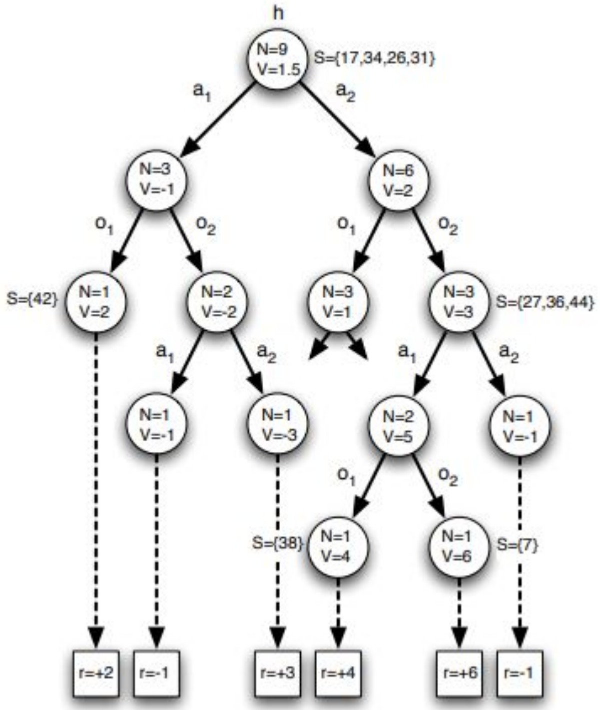
1. Selection (UCT)
   - **(PO-)UCT**
   - Sample particle **s** from **B(s|h)**
   - Black box **Simulator for s' and o'**
2. Expansion
3. Simulation (Rollout)
4. Backprop
   - **Update** each **B(s'|hao)** by adding corresponding simulated **s'** to particle set.

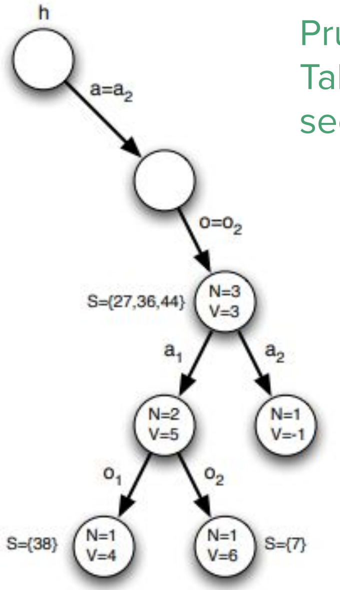
Partially Observable Monte-Carlo Planning



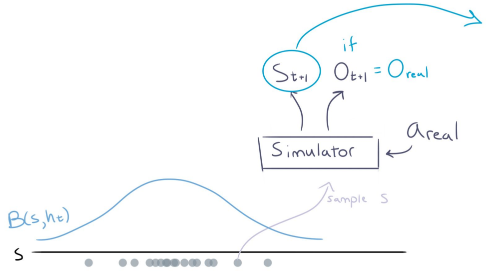
1. Selection (UCT)
   - ➔ **(PO-)UCT**
   - ➔ Sample particle **s** from **B(s|h)**
   - ➔ Black box **Simulator for s' and o'**
2. Expansion
3. Simulation (Rollout)
4. Backprop
   - ➔ **Update** each **B(s'|hao)** by adding corresponding simulated **s'** to particle set.

Partially Observable Monte-Carlo Planning





Prune Tree after
Taking Action a2 and
seeing observation o2

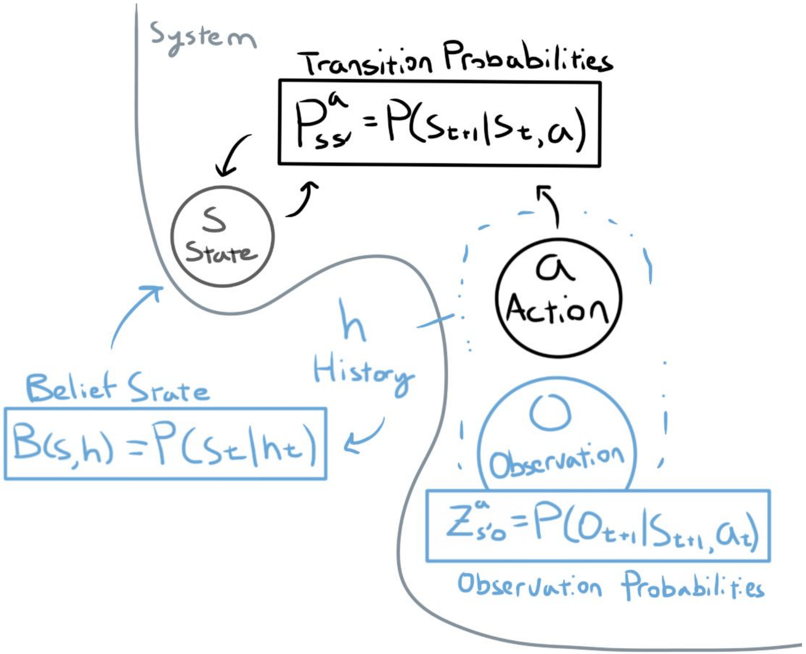


Discussion

Break “curse of dimensionality”.

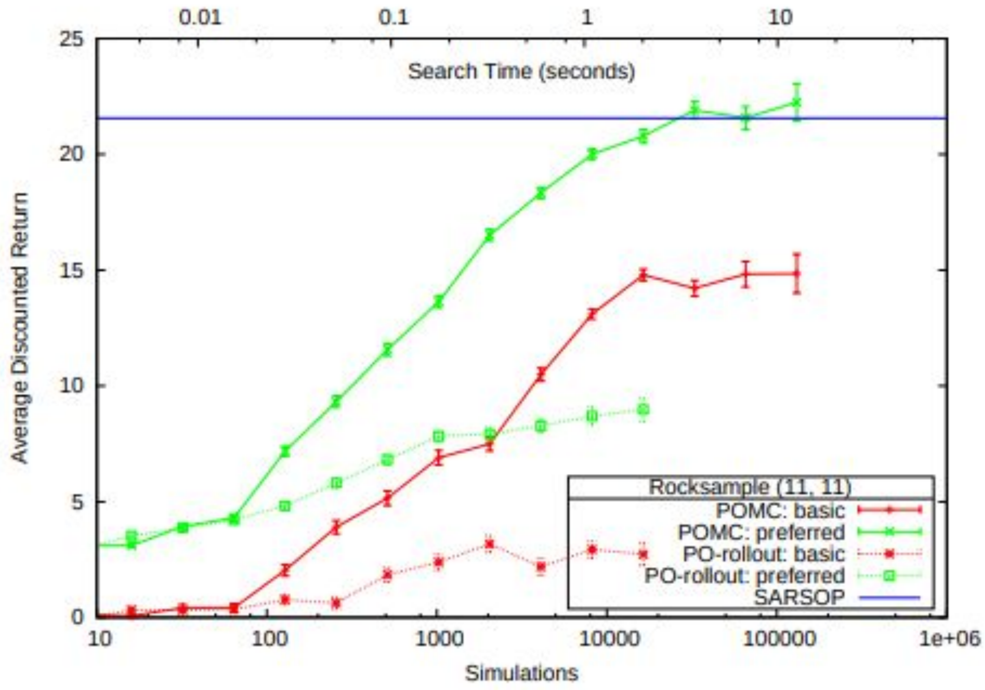
Requires only black box simulator.

$$(s_{t+1}, o_{t+1}, r_{t+1}) \sim \mathcal{G}(s_t, a_t)$$



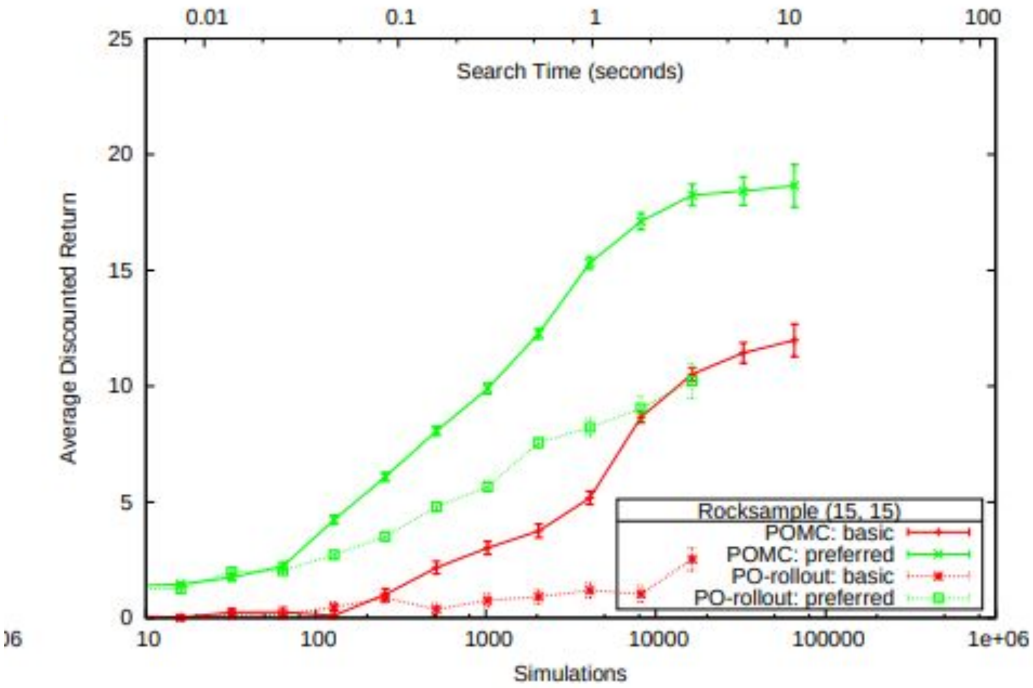
Thank you!

Experiments - Rocksample



~250k states

Experiments - Rocksample



7M states