Monte Carlo Tree Search guided by Symbolic Advice for MDPs

31st International Conference on Concurrency Theory
CONCUR 2020

Damien Busatto-Gaston, Debraj Chakraborty
and Jean-Francois Raskin

Université Libre de Bruxelles

September 2, 2020

Introduction

- Controller $\mid$ Stochastic model of environment $\models$ System
- Maximize reward $\leadsto$ exploring the consequences of our decisions
- Very large systems $\leadsto$ sparse exploration, anytime algorithms

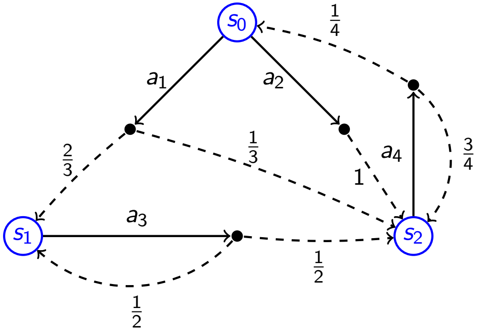
Introduction

- Controller $|$ Stochastic model of environment $\models$ System
- Maximize reward $\rightsquigarrow$ exploring the consequences of our decisions
- Very large systems $\rightsquigarrow$ sparse exploration, anytime algorithms

- Symbolic advice to guide the exploration
- Formal guarantees
- Experimental results

Playing on an MDP

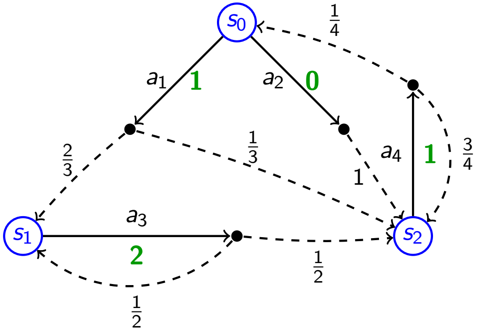
Markov Decision Process



- Path of length 2: $s_0 \xrightarrow{a_1} \overset{\frac{2}{3}}{\dashrightarrow} s_1 \xrightarrow{a_3} \overset{\frac{1}{2}}{\dashrightarrow} s_2$

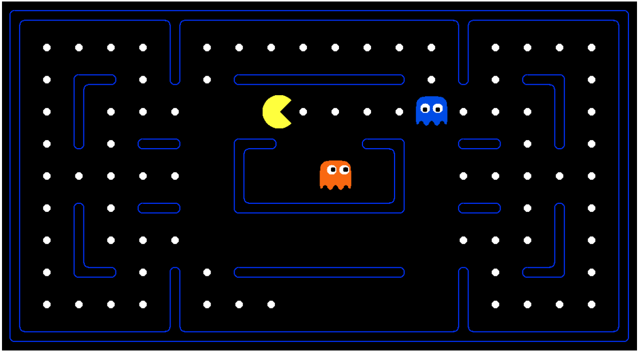
Playing on an MDP

Markov Decision Process



- Path of length 2: $s_0 \xrightarrow{a_1} \overset{\frac{2}{3}}{\dashrightarrow} s_1 \xrightarrow{a_3} \overset{\frac{1}{2}}{\dashrightarrow} s_2$
- Finite-horizon total reward (horizon H)
- $\mathrm{Val}(s_0) = \sup_{\sigma:\mathrm{Paths}\to A} \mathbb{E}\left[\mathrm{Reward}(p)\right]$
  where p is a random variable over $\mathrm{Paths}^H(s_0, \sigma)$
- Link with infinite-horizon average reward for H large enough

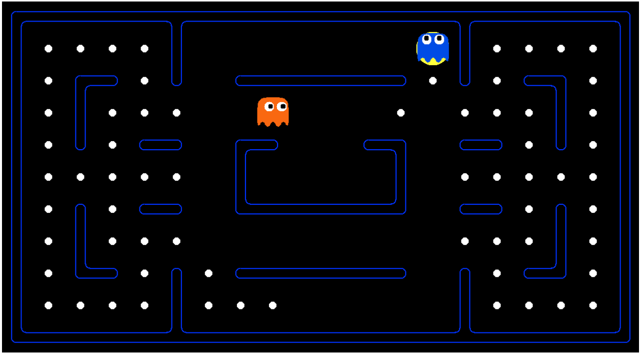
Example: Pac-Man as an MDP



- Controller: Pac-Man
- Probabilistic model of ghosts

- States: position of every agent, what food is left
- Actions: Pac-Man moves
- Stochastic transitions: ghost moves

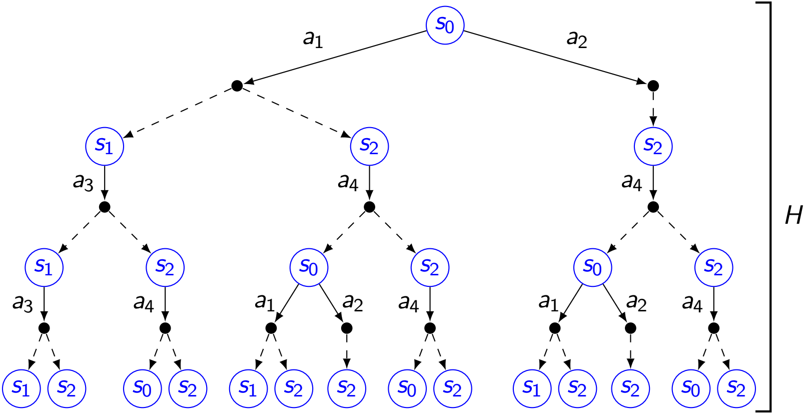
Example: Pac-Man as an MDP



- Controller: Pac-Man
- Probabilistic model of ghosts
- Reward for eating food
- Large penalty for losing

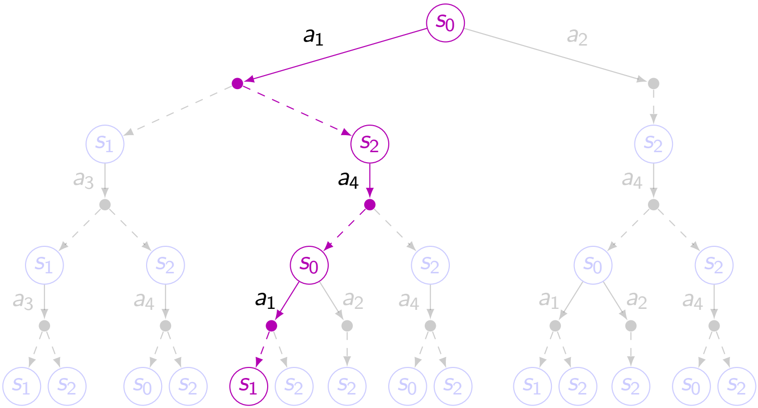
- States: position of every agent, what food is left
- Actions: Pac-Man moves
- Stochastic transitions: ghost moves
- Large MDP: $\sim 10^{16}$ states

Sparse exploration



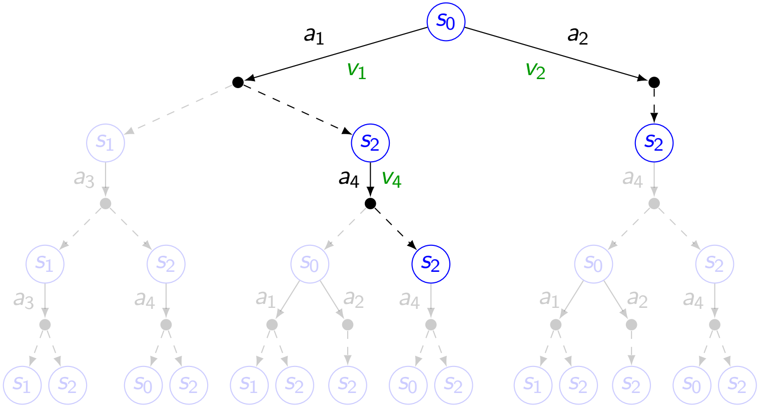
- Large unfolding $\rightsquigarrow$ heuristics

Sparse exploration



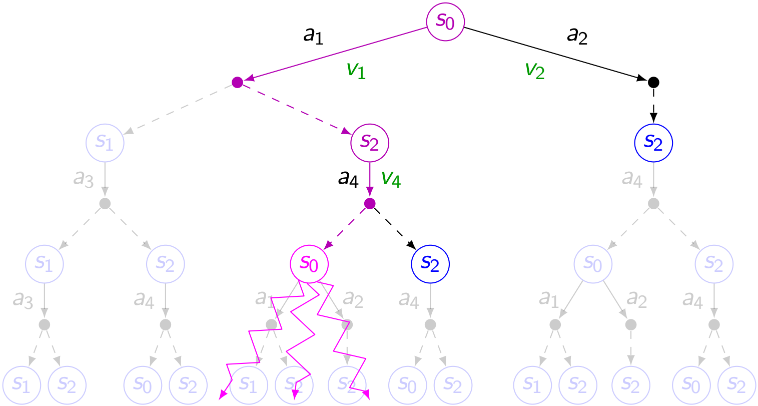
- Large unfolding $\rightsquigarrow$ heuristics
- Uniform simulation: select actions at random to obtain a path
- Average reward over a few simulations $\rightsquigarrow$ estimate of $\mathrm{Val}^H(s_0)$
- No formal guarantees of convergence

Monte Carlo tree search (MCTS)



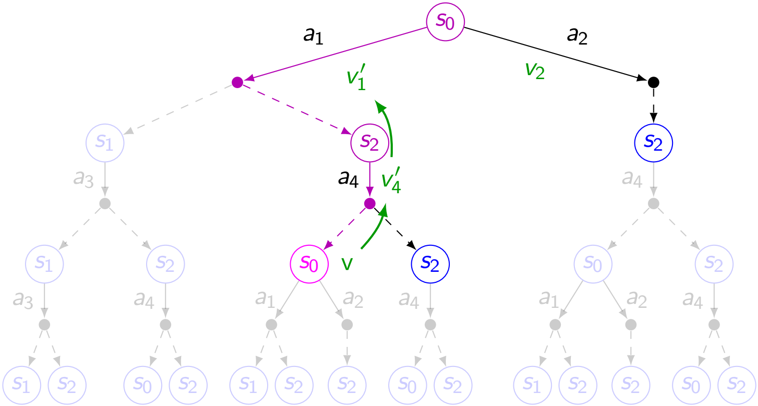
- Iterative construction of a sparse tree with value estimates

Monte Carlo tree search (MCTS)



- Iterative construction of a sparse tree with value estimates
- Selection of a new node $\rightsquigarrow$ simulation

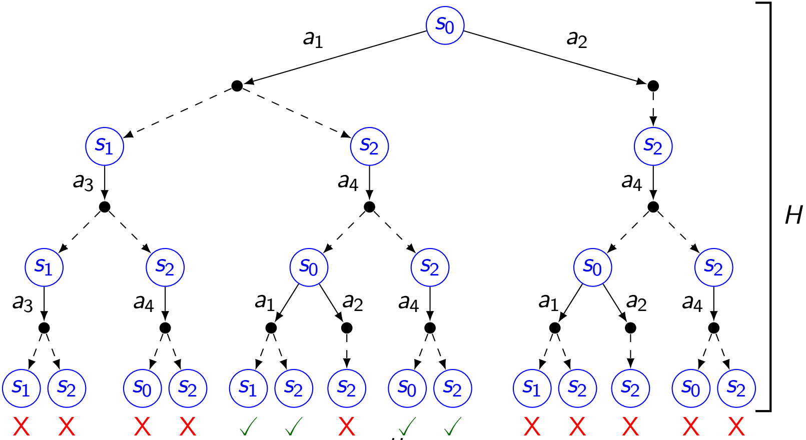
Monte Carlo tree search (MCTS)



- Iterative construction of a sparse tree with value estimates
- Selection of a new node $\rightsquigarrow$ simulation $\rightsquigarrow$ update of the estimates
- MCTS converges to the optimal choice (Kocsis & Szepesvári, 2006)

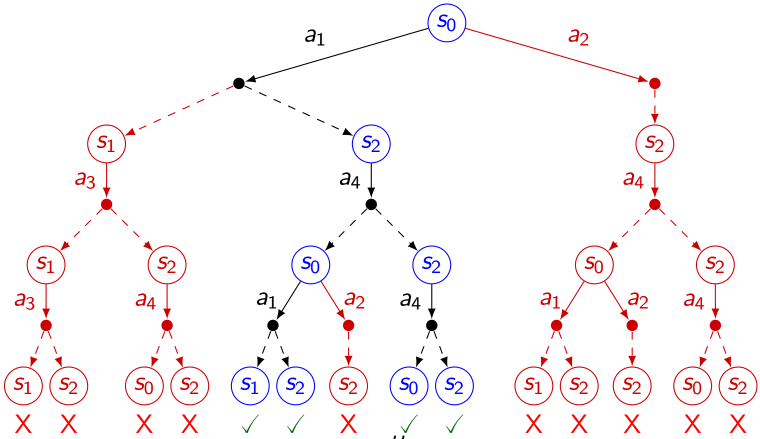
Symbolic advice

Symbolic advice



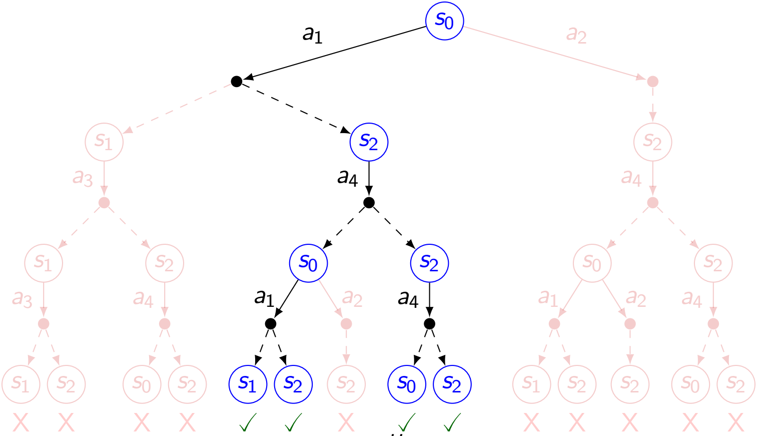
- An advice is a subset of $\text{Paths}^H(s_0)$
- Defined symbolically as a logical formula φ (reachability or safety property, LTL formula over finite traces, regular expression . . .)

Symbolic advice



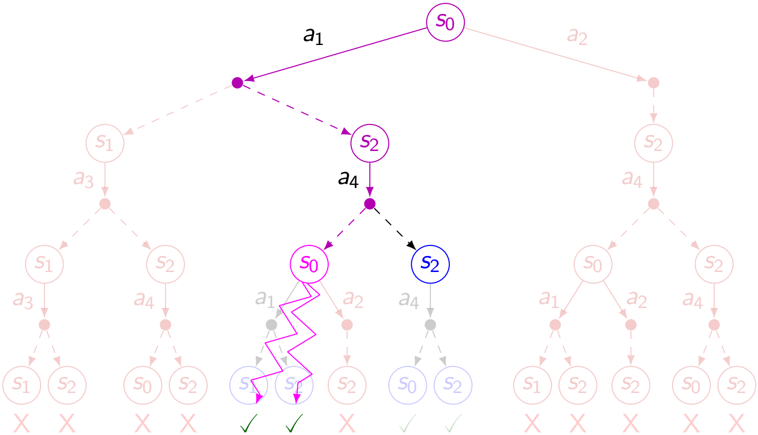
- An advice is a subset of $\text{Paths}^H(s_0)$
- Defined symbolically as a logical formula φ (reachability or safety property, LTL formula over finite traces, regular expression . . .)

Symbolic advice



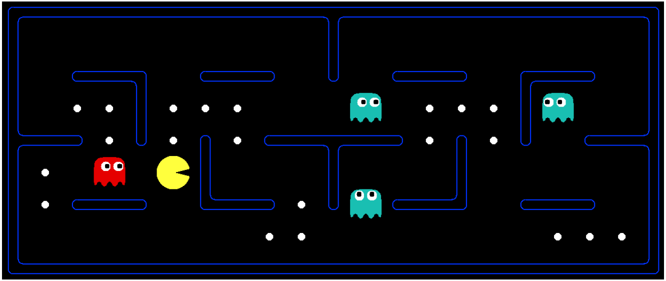
- An advice is a subset of $\text{Paths}^H(s_0)$
- Defined symbolically as a logical formula φ (reachability or safety property, LTL formula over finite traces, regular expression ...)
- φ defines a pruning of the unfolded MDP

MCTS under advice



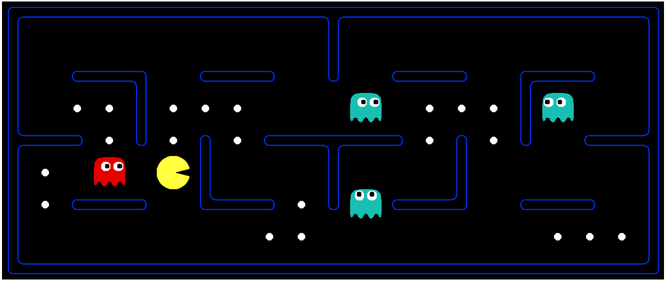
- Select actions in the unfolding pruned by a selection advice φ
- Simulation is restricted according to a simulation advice ψ

Safety property



- Some states are unsafe and should be avoided
- Advice ψ: set of safe paths $\quad\quad \mathbf{G}.\ (x,y)_p \neq (x,y)_g$

Safety property



- Some states are unsafe and should be avoided
- Advice ψ: set of safe paths $\quad\quad \mathbf{G}.\ (x,y)_p \neq (x,y)_g$
- Stronger property: safety is ensured no matter what stochastic transitions are taken
- Enforceable advice φ: set of paths so that every action chosen is compatible with a strategy that enforces safety with horizon H

Boolean Solvers

- The safety property ψ can be encoded as a Boolean Formula

Boolean Solvers

- The safety property ψ can be encoded as a Boolean Formula

QBF solver

- A first action a_0 is compatible with φ iff

$$\forall s_1 \exists a_1 \forall s_2 \ldots, \; s_0 a_0 s_1 a_1 s_2 \ldots \models \psi$$

- Inductive way of constructing paths that satisfy the enforceable advice φ
- Alternation of quantifiers $\rightsquigarrow$ guarantee safety for $h < H$

Boolean Solvers

- The safety property ψ can be encoded as a Boolean Formula

QBF solver

- A first action a_0 is compatible with φ iff

$$\forall s_1 \exists a_1 \forall s_2 \ldots, \; s_0 a_0 s_1 a_1 s_2 \ldots \models \psi$$

- Inductive way of constructing paths that satisfy the enforceable advice φ
- Alternation of quantifiers $\rightsquigarrow$ guarantee safety for $h < H$

Weighted sampling

- Simulation of safe paths according to ψ
- Weighted SAT sampling (Chakraborty, Fremont, Meel, Seshia, & Vardi, 2014)

Theoretical guarantees

Multi-armed bandit and UCB algorithm

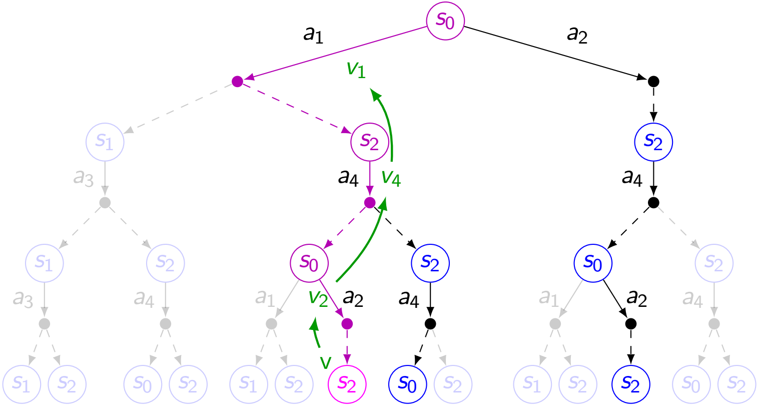
- Finite set of machines (actions), that give rewards when played
- Every machine has a hidden reward distribution
- How to find the best machine (expected reward)?
- Take samples according to a strategy, try to minimize regret

Multi-armed bandit and UCB algorithm

- Finite set of machines (actions), that give rewards when played
- Every machine has a hidden reward distribution
- How to find the best machine (expected reward)?
- Take samples according to a strategy, try to minimize regret

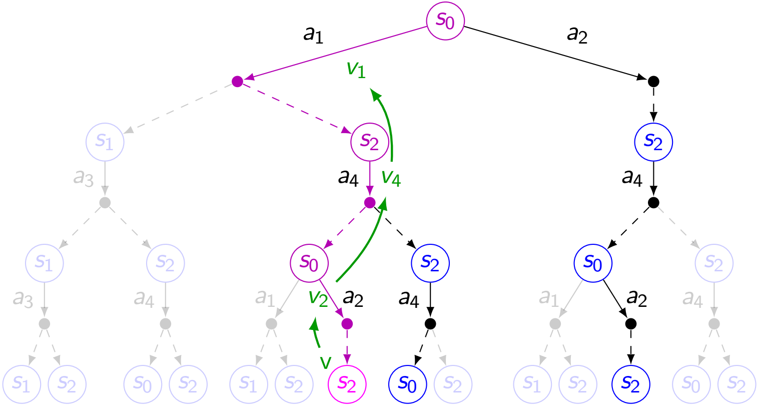
- UCB (Auer, Cesa-Bianchi, & Fischer, 2002) is a popular strategy
- It offers a solution to the exploitation/exploration trade-off
- Optimal: regret is bounded logarithmically

The MCTS algorithm using UCB (Kocsis & Szepesvári, 2006)



- Every state is seen as an instance of a bandit problem
- Selecting an action $\rightsquigarrow$ reward in the backwards propagation phase

The MCTS algorithm using UCB (Kocsis & Szepesvári, 2006)



- Every state is seen as an instance of a bandit problem
- Selecting an action $\rightsquigarrow$ reward in the backwards propagation phase
- Using UCB for selection $\rightsquigarrow$ the rewards change over time
- Non-stationary bandits with Drift conditions

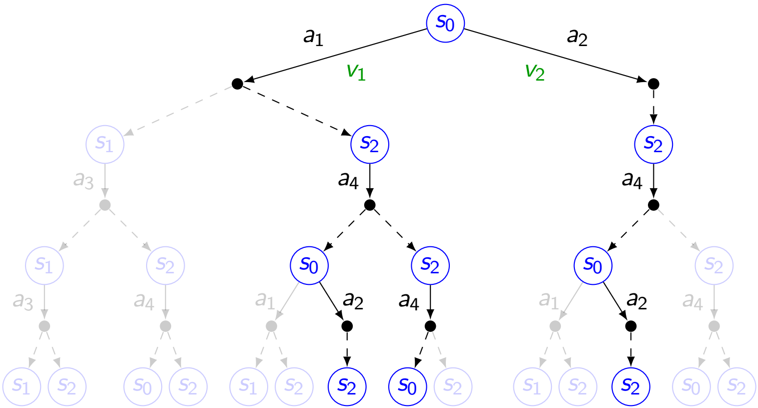
Non-stationary bandits and drift conditions

- The reward distributions change after each play
- They must follow some assumptions (Drift conditions):
  - The expected average reward of the first n plays of a converges
  - Tail inequalities: upper bound on the probability that we observe an average reward for action a after n plays that deviates from its expected value by more than a certain amount.

Non-stationary bandits and drift conditions

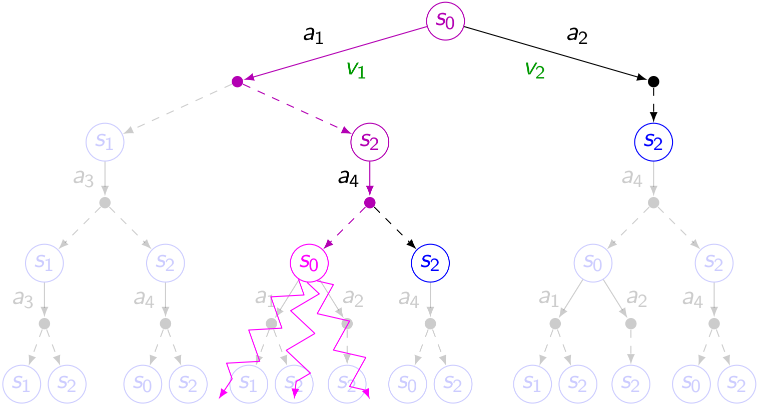
- The reward distributions change after each play
- They must follow some assumptions (Drift conditions):
  - The expected average reward of the first n plays of a converges
  - Tail inequalities: upper bound on the probability that we observe an average reward for action a after n plays that deviates from its expected value by more than a certain amount.
- UCB can be extended under these assumptions
- When using UCB for selecting actions in MCTS, the reward distributions satisfy the drift conditions (Kocsis & Szepesvári, 2006)

Convergence of MCTS (Kocsis & Szepesvári, 2006)



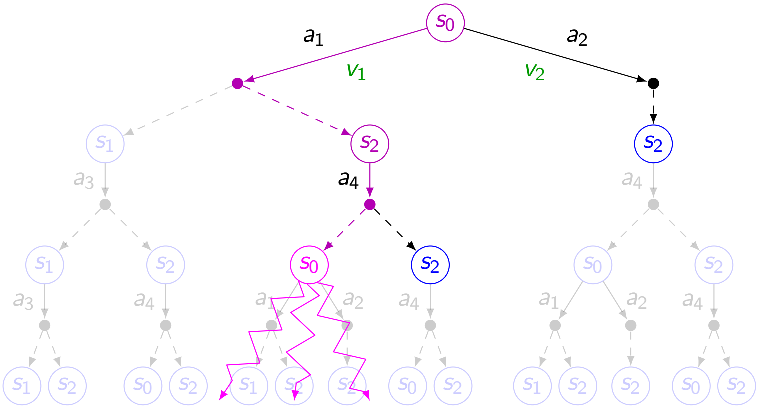
- After a given number of iterations n, MCTS outputs the best action
- The probability of choosing a suboptimal action converges to zero
- v_i converges to the real value of a_i at a speed of $(\log n)/n$

Convergence of MCTS with simulation



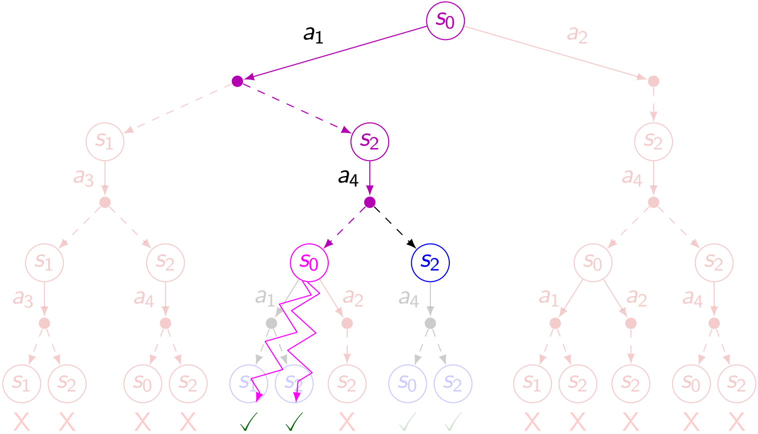
- Unlike (Kocsis & Szepesvári, 2006), MCTS is often implemented with a simulation phase used to initialise value estimates
- This changes the reward distributions of all UCB instances

Convergence of MCTS with simulation



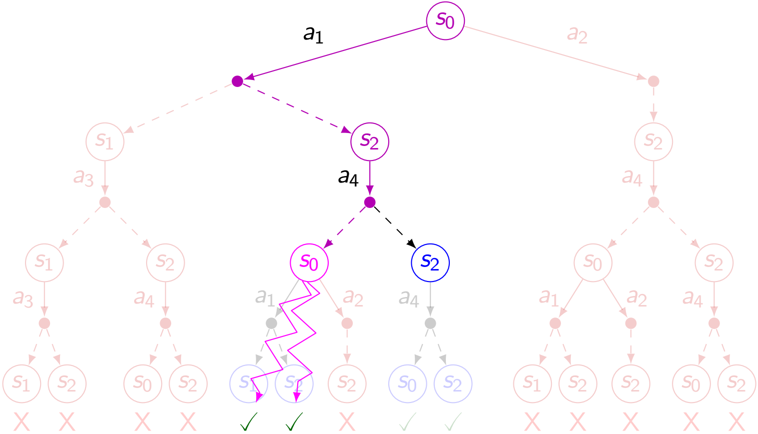
- Unlike (Kocsis & Szepesvári, 2006), MCTS is often implemented with a simulation phase used to initialise value estimates
- This changes the reward distributions of all UCB instances
- We show that the convergence properties of MCTS are maintained for all simulations: any strategy can be used to draw samples

MCTS under advice



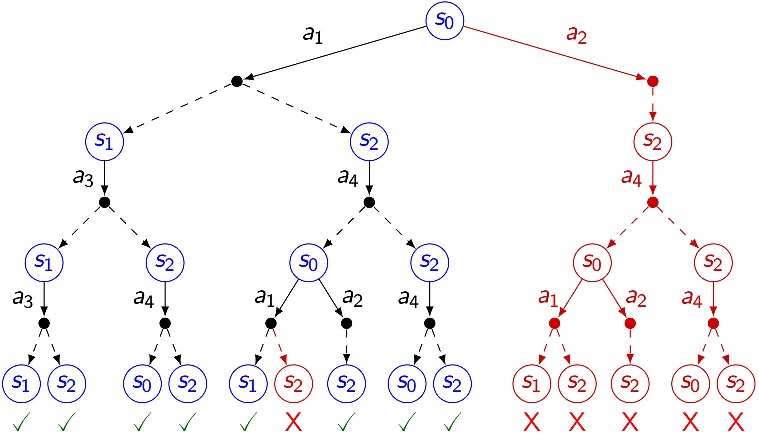
- Select actions in the unfolding pruned by a selection advice φ
- Simulation is restricted according to a simulation advice ψ

MCTS under advice



- Select actions in the unfolding pruned by a selection advice φ
- Simulation is restricted according to a simulation advice ψ
- We show that the convergence properties are maintained:
  - for a selection advice that satisfies some assumptions,
  - for all simulation advice.

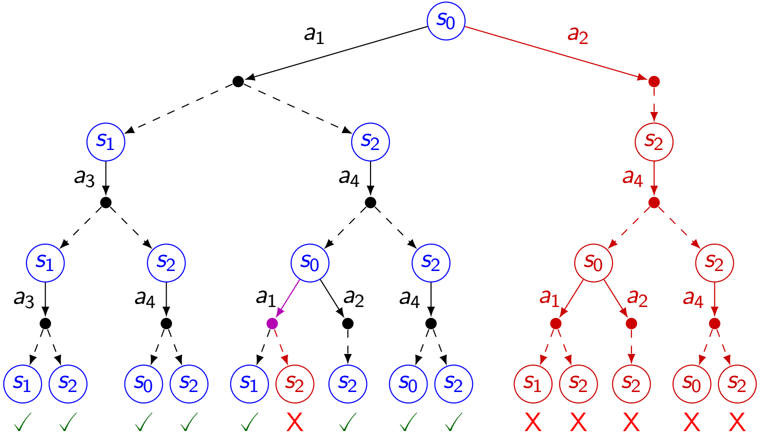
Assumptions on the selection advice



The selection advice must

- be strongly enforceable: can be enforced by controller if the MDP is seen as a game $\rightsquigarrow$ does not partially prune stochastic transitions

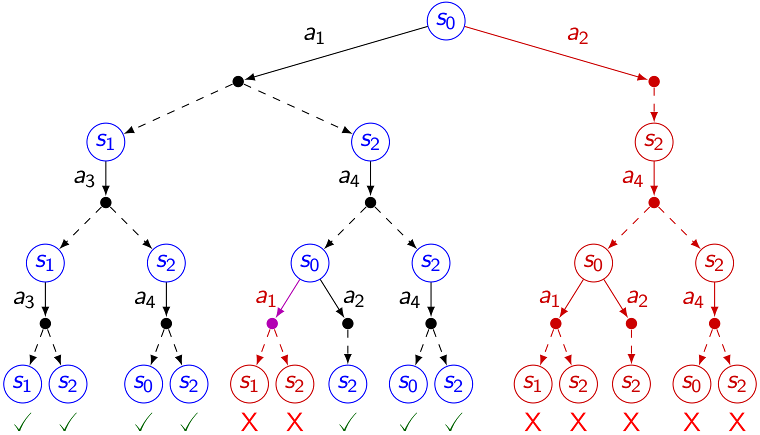
Assumptions on the selection advice



The selection advice must

- be strongly enforceable: can be enforced by controller if the MDP is seen as a game $\rightsquigarrow$ does not partially prune stochastic transitions

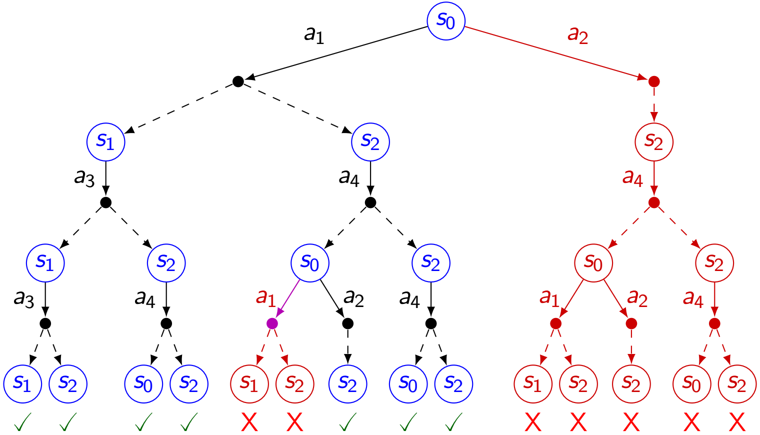
Assumptions on the selection advice



The selection advice must

- be strongly enforceable: can be enforced by controller if the MDP is seen as a game $\rightsquigarrow$ does not partially prune stochastic transitions
- satisfy an optimality assumption: does not prune all optimal actions

Assumptions on the selection advice

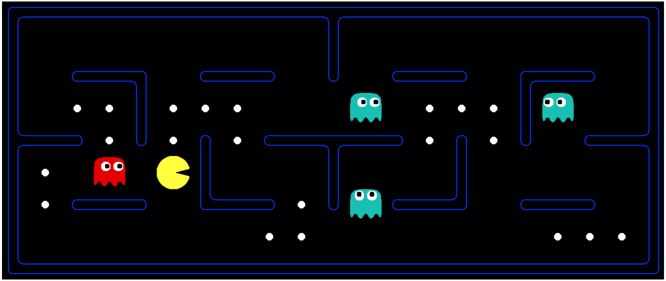


The selection advice must

- be strongly enforceable: can be enforced by controller if the MDP is seen as a game $\rightsquigarrow$ does not partially prune stochastic transitions
- satisfy an optimality assumption: does not prune all optimal actions

Experimental results

Experimental results



9 × 21 maze, 4 random ghosts

Algorithm	win	loss	no result after 300 steps	food
MCTS	17	59	24	67
MCTS+Selection advice	25	54	21	71
MCTS+Simulation advice	71	29	0	88
MCTS+both advice	85	15	0	94
Human	44	56	0	75

Conclusion

Contributions

- How to inject domain knowledge in MCTS?
  - symbolic advice for selection and simulation
- How to preserve the convergence guarantees of MCTS?
  - strongly enforceable advice with an optimality assumption
- How to implement them?
  - symbolic solutions using SAT and QBF solvers
- Does it work on large MDPs?
  - good results with safety advice on the Pac-Man domain

Conclusion

Contributions

- How to inject domain knowledge in MCTS?
  - symbolic advice for selection and simulation
- How to preserve the convergence guarantees of MCTS?
  - strongly enforceable advice with an optimality assumption
- How to implement them?
  - symbolic solutions using SAT and QBF solvers
- Does it work on large MDPs?
  - good results with safety advice on the Pac-Man domain

Future works

- Compiler LTL $\rightarrow$ symbolic advice
- Study interactions with reinforcement learning techniques (and neural networks)
- Weighted advice

Thank you