MDP Algorithms

Thomas Keller

University of Basel

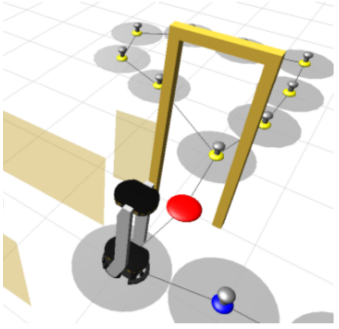
June 20, 2018

Outline of this lecture

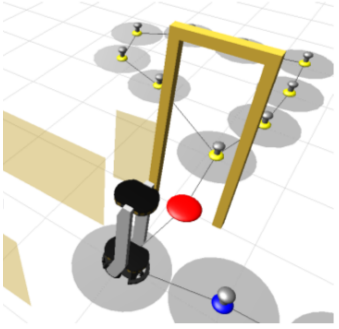
- Markov decision processes
- Planning via determinization
- Monte-Carlo methods
- Monte-Carlo Tree Search
- Heuristic Search
- Trial-based Heuristic Tree Search

Please ask questions at any time!

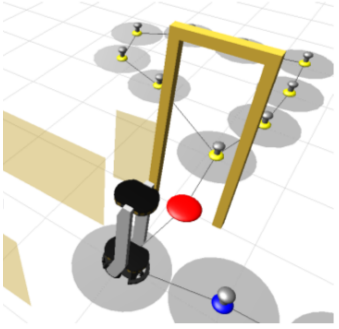
Motivation

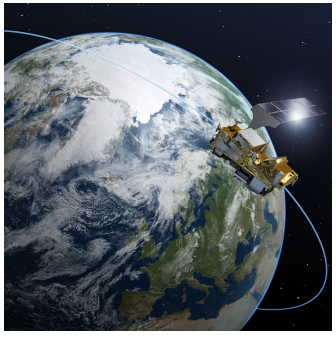


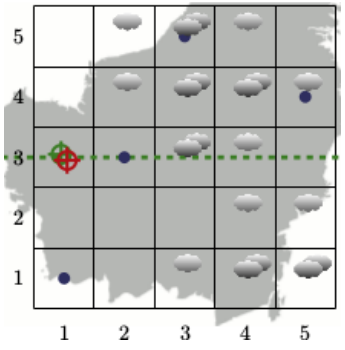
Motivation



Motivation







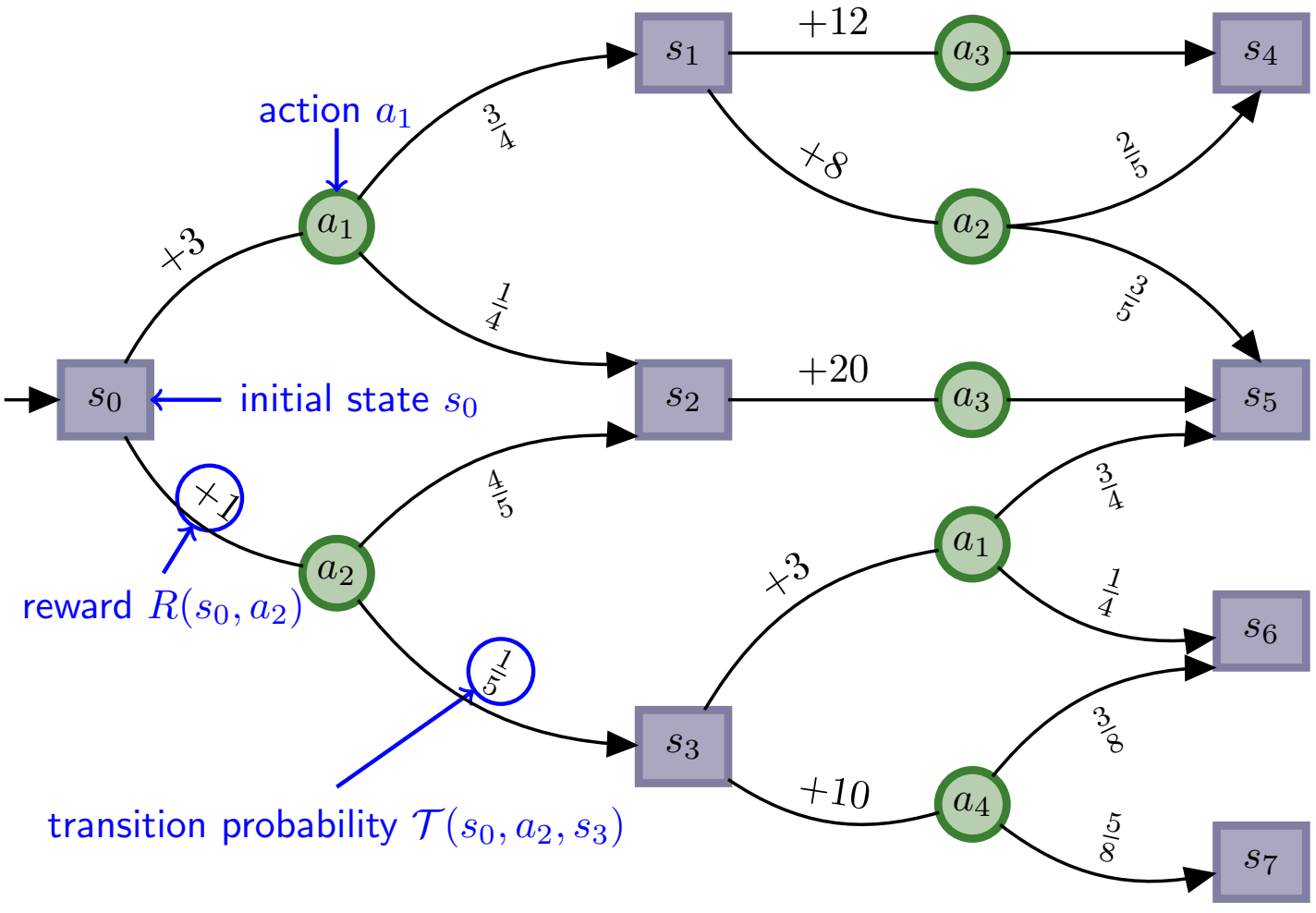
Markov decision process

Definition (Markov decision process)

A (finite-horizon) Markov decision process (MDP) is a 6-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, s_0, H \rangle$, where

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{T} : S \times A \times S \mapsto [0, 1]$ is the transition function
- $\mathcal{R} : S \times A \mapsto \mathbb{R}$ is the reward function
- $s_0 \in \mathcal{S}$ is the initial state
- $H \in \mathbb{N}$ is the horizon

Markov decision process: example



MDPs are acyclic in finite-horizon setting

Policy

Definition (Policy)

A partial policy for a Markov decision process $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, s_0, H \rangle$ is a mapping $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1] \cup \{\perp\}$.
A partial policy π is executable in $\mathcal{M}$ if $\pi(s, a) \neq \perp$ for all states s and actions a that can be reached by π from s_0.

$\pi(s, a)$ gives probability that action a is executed in state s under application of policy π

State- and Action Value

Definition (State- and Action Values)

The state-value $V^\pi(s)$ of a state $s \in \mathcal{S}$ under policy π is

$$V^\pi(s) := \begin{cases} 0 & \text{if } s \text{ is terminal} \\ Q^\pi(s, \pi(s)) & \text{otherwise,} \end{cases}$$

where $Q^\pi(s,a)$ is the action-value of s and action $a \in \mathcal{A}$ under π

$$Q^\pi(s,a) := R(s,a) + \sum_{s' \in \mathcal{S}} \left(T(s,a,s') \cdot V^\pi(s') \right)$$

for all state-action pairs (s,a).

Optimal Policy

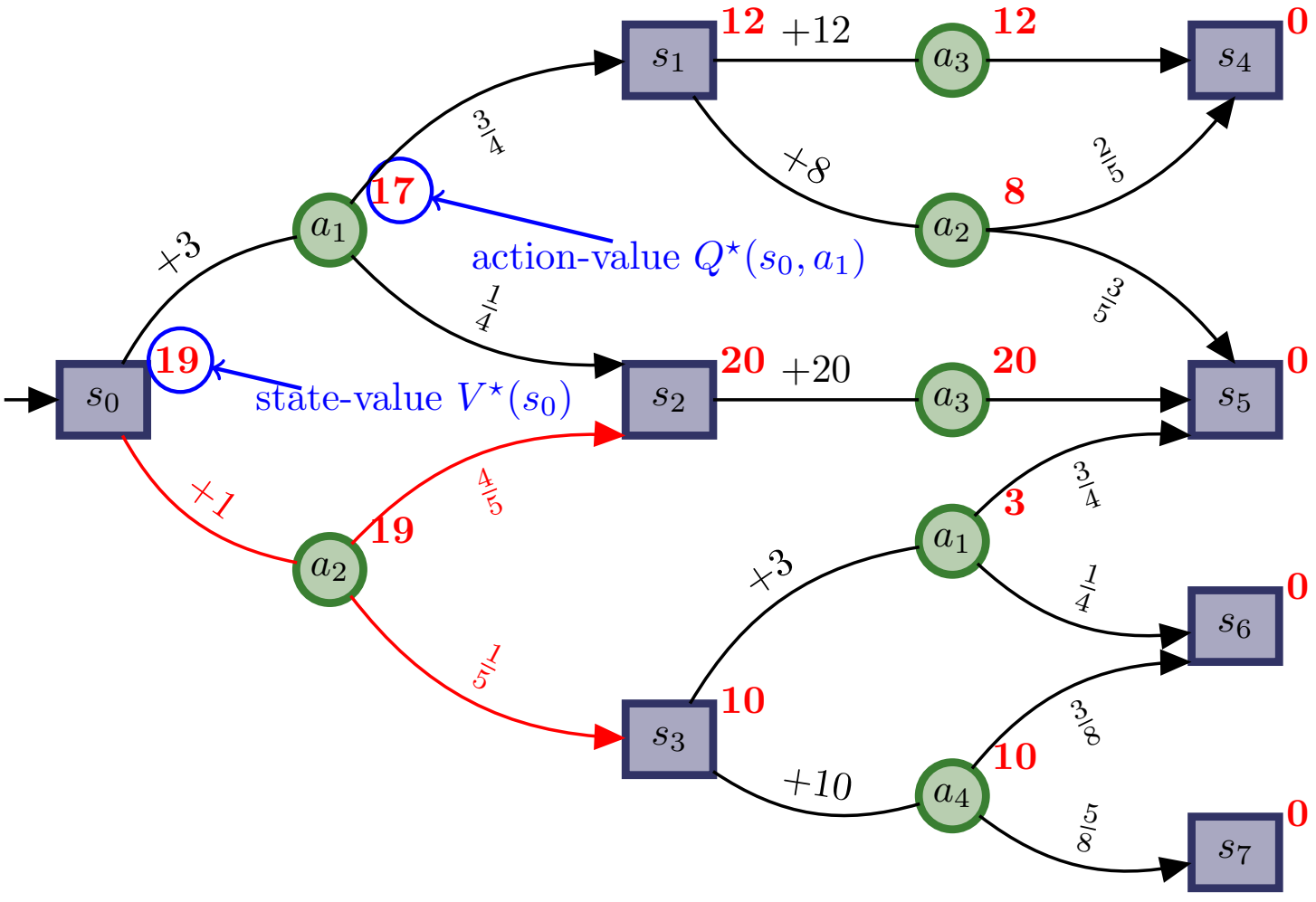
- policy π is optimal if $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ are maximal among all π (in the following, $\pi^{\star}$, $V^{\star}(s)$ and $Q^{\star}(s,a)$)
- compute $V^{\star}(s)$ and $Q^{\star}(s,a)$ as

$$V^{\star}(s) := \begin{cases} 0 & \text{if } s \text{ is terminal} \\ \max_{a \in \mathcal{A}} Q^{\star}(s,a) & \text{otherwise} \end{cases}$$

$$Q^{\star}(s,a) := R(s,a) + \sum_{s' \in \mathcal{S}} \left(T(s,a,s') \cdot V^{\star}(s') \right).$$

- start with terminal states and proceed backwards in DAG until initial state

Optimal solution



State- and action-values describe expected reward

Lecture goals

How can we act well despite the complexity of the problem?

- Which algorithms for probabilistic planning are there?
- Which are their strengths and weaknesses?
- What do they have in common, and what are the differences?

Anytime Planning: Plan-Execute-Monitor Cycle

- size of executable policy exponential in horizon
- compact representation of executable policy required to describe solution $\Rightarrow$ not part of this lecture
- instead: perform plan-execute-monitor cycle:
  - plan action a for the current state s_0
  - execute a
  - update s_0 and H in $\mathcal{M}$
  - repeat until $H = 0$

Anytime Planning: Plan-Execute-Monitor Cycle

Advantages and disadvantages of plan-execute-monitor:

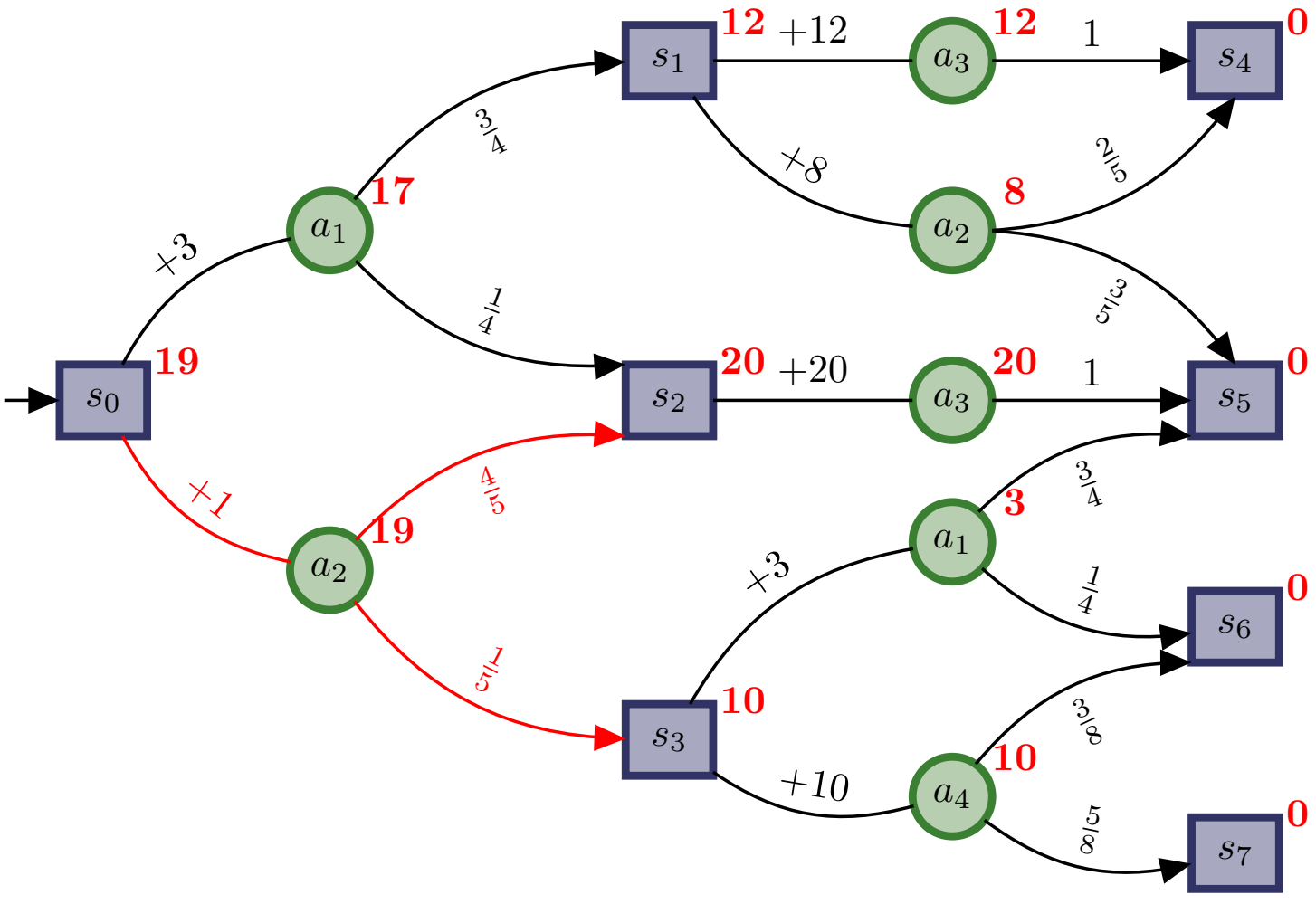
+ avoid loss of precision that often comes with compact description of executable policy
+ do not waste time with planning for states that are never reached during execution
- poor decisions can be avoided by spending more time with deliberation before execution

Idea

- replace probabilistic actions with deterministic ones
- leads to classical planning problem
- (often) determinization can be solved in practice even if MDP cannot

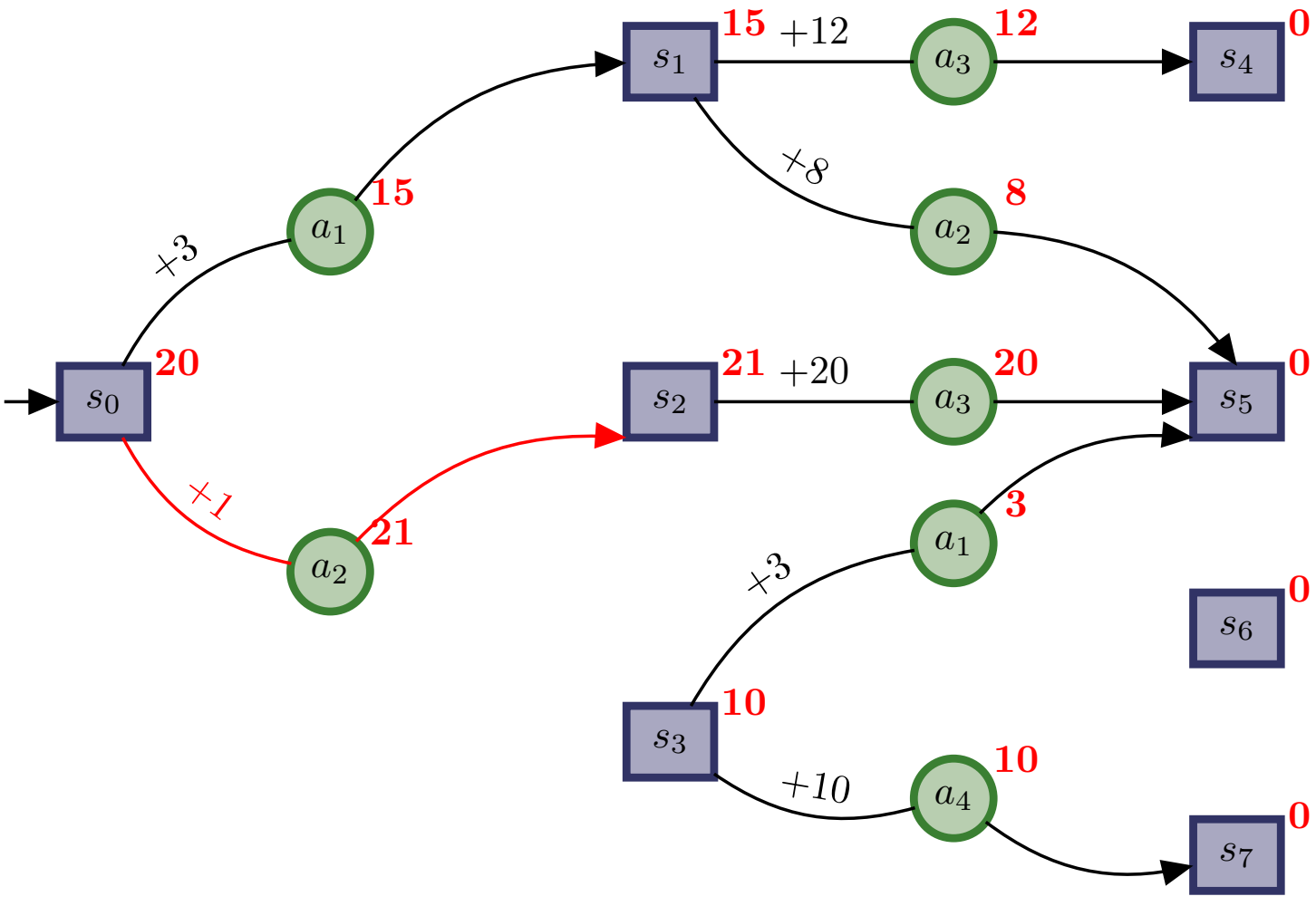
How do we come up with a determinization?

Determinization: Example

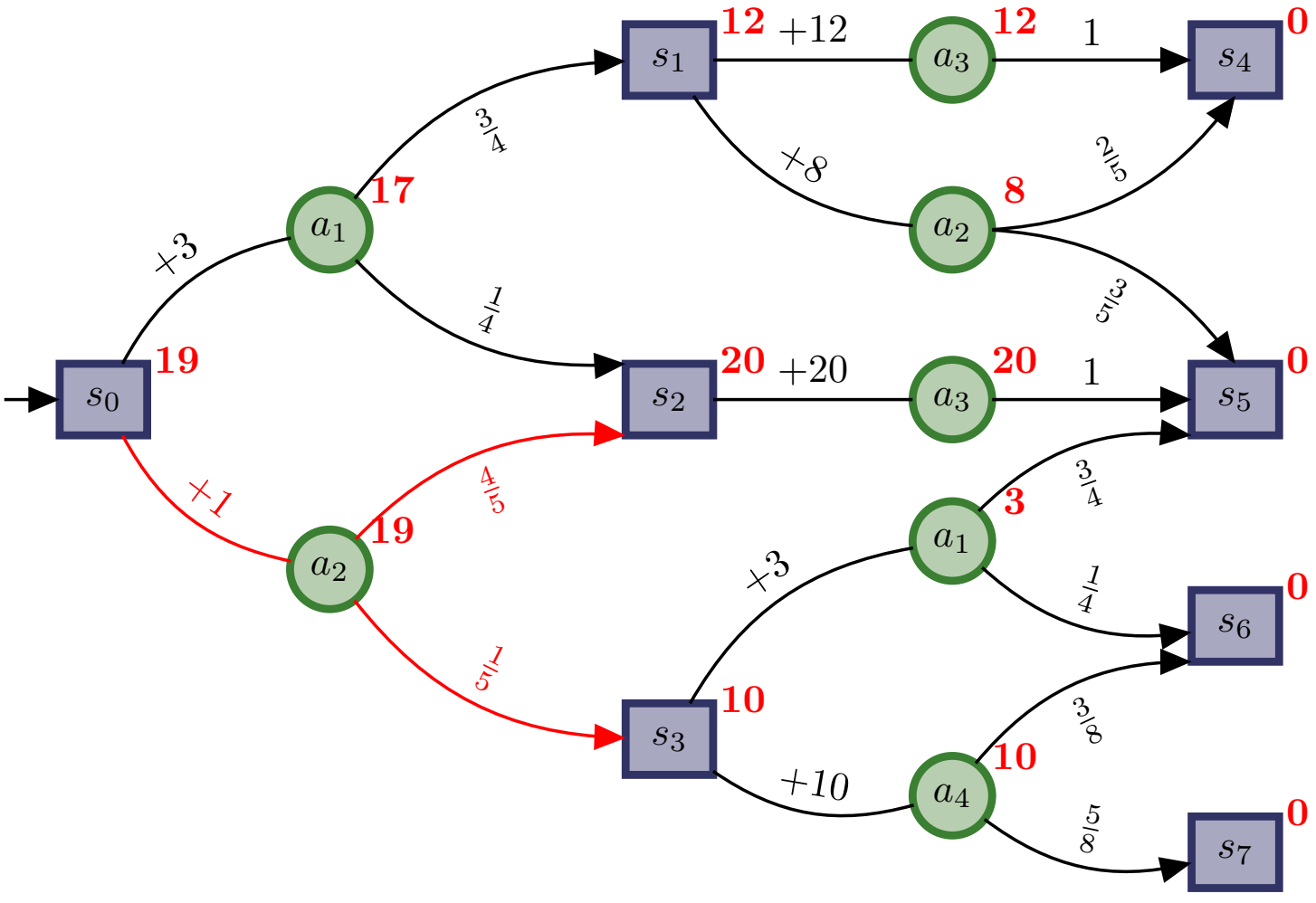


Determinization: Single-outcome determinization

Remove all outcomes but one

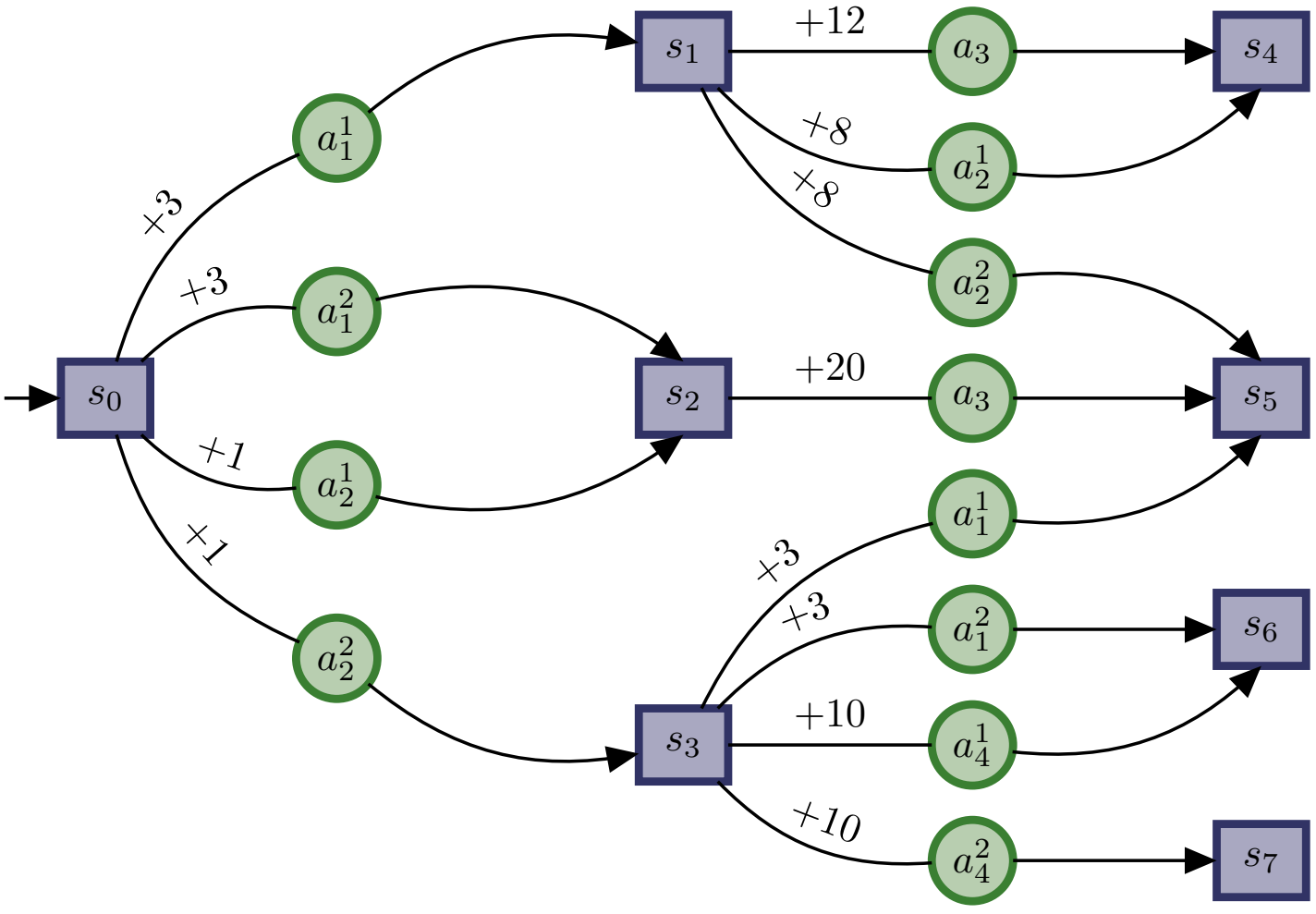


All-outcomes determinization: Example

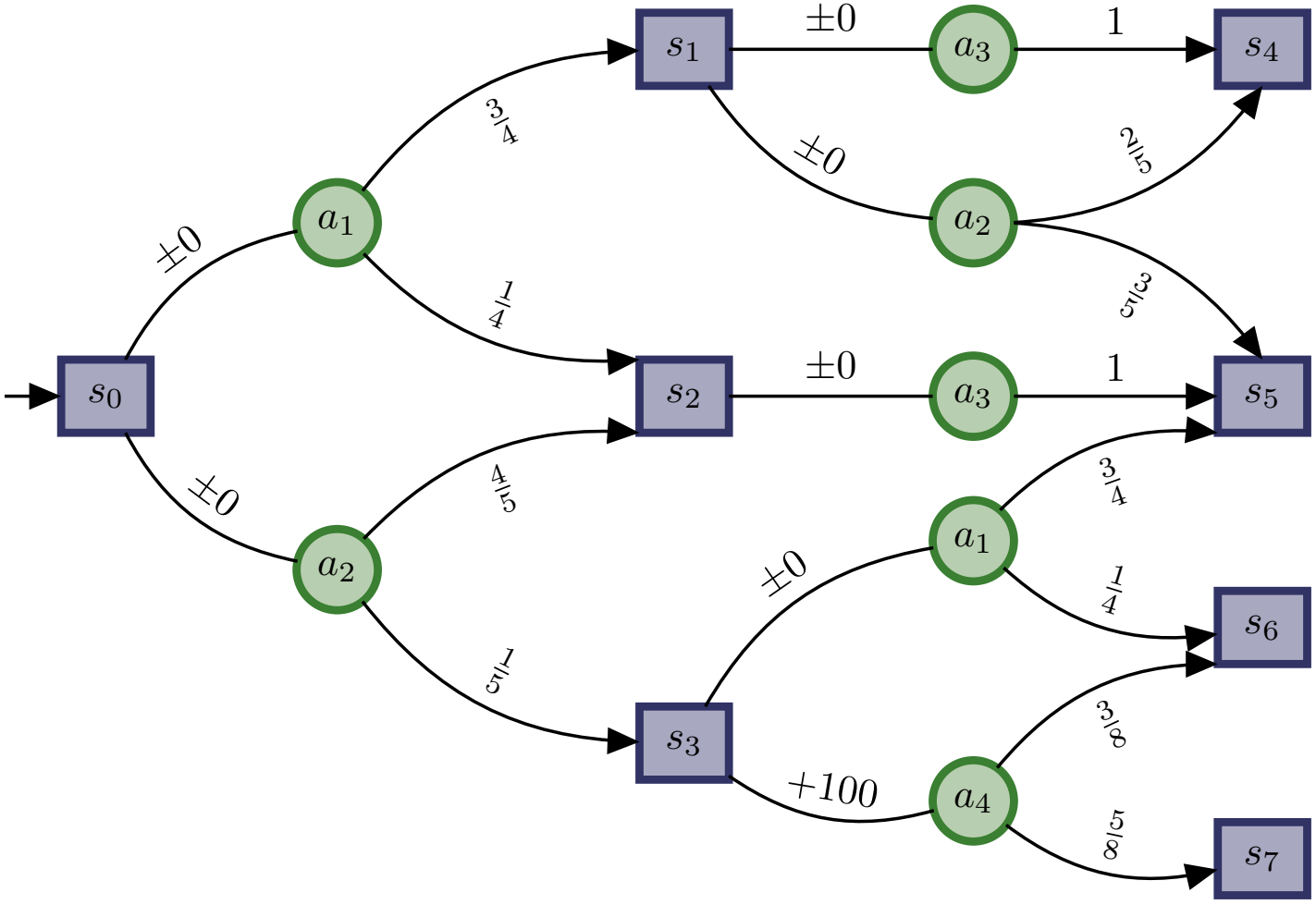


Determinization: All-outcomes determinization

Generate one action per outcome

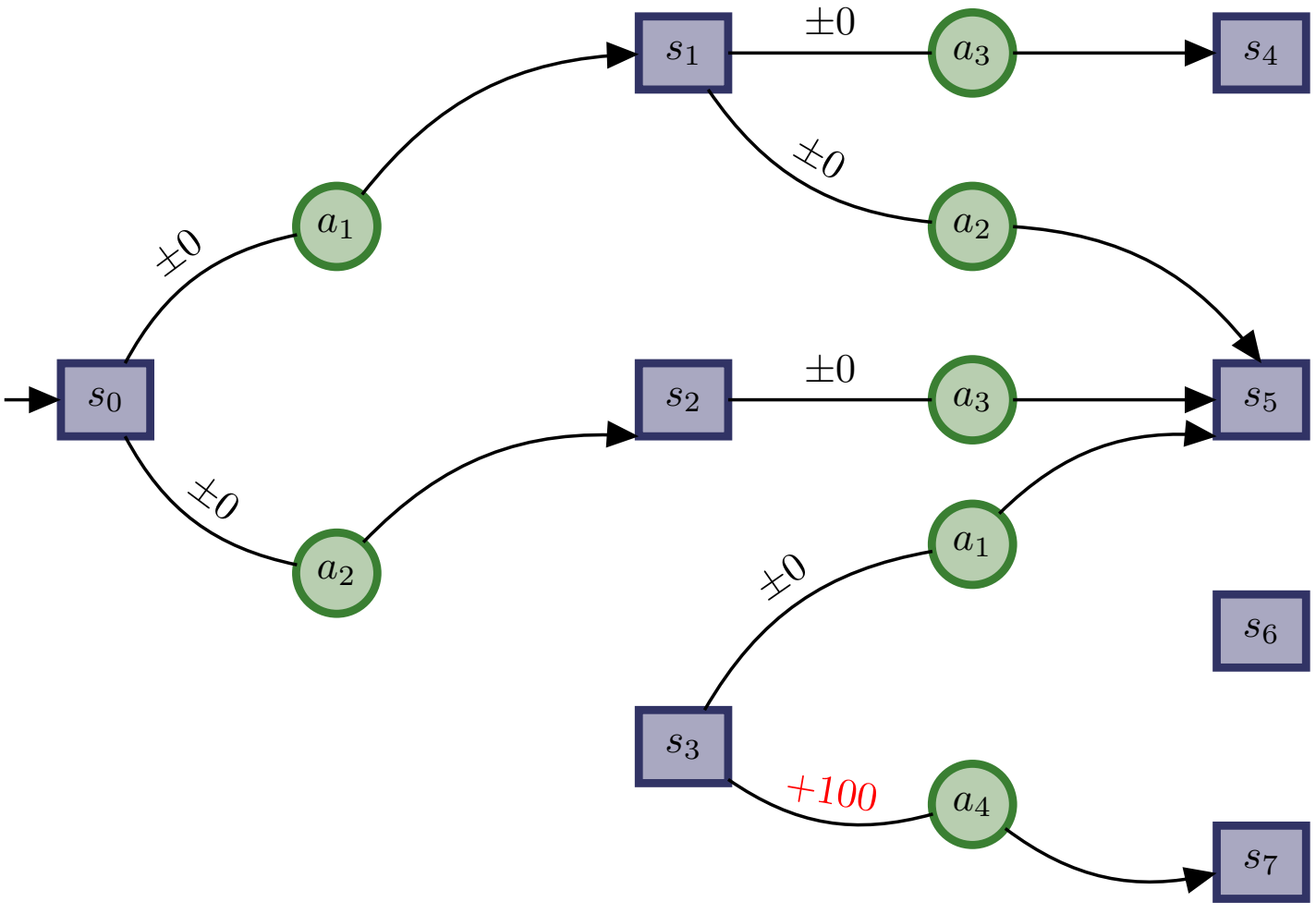


Single-outcome Determinization: Limitations



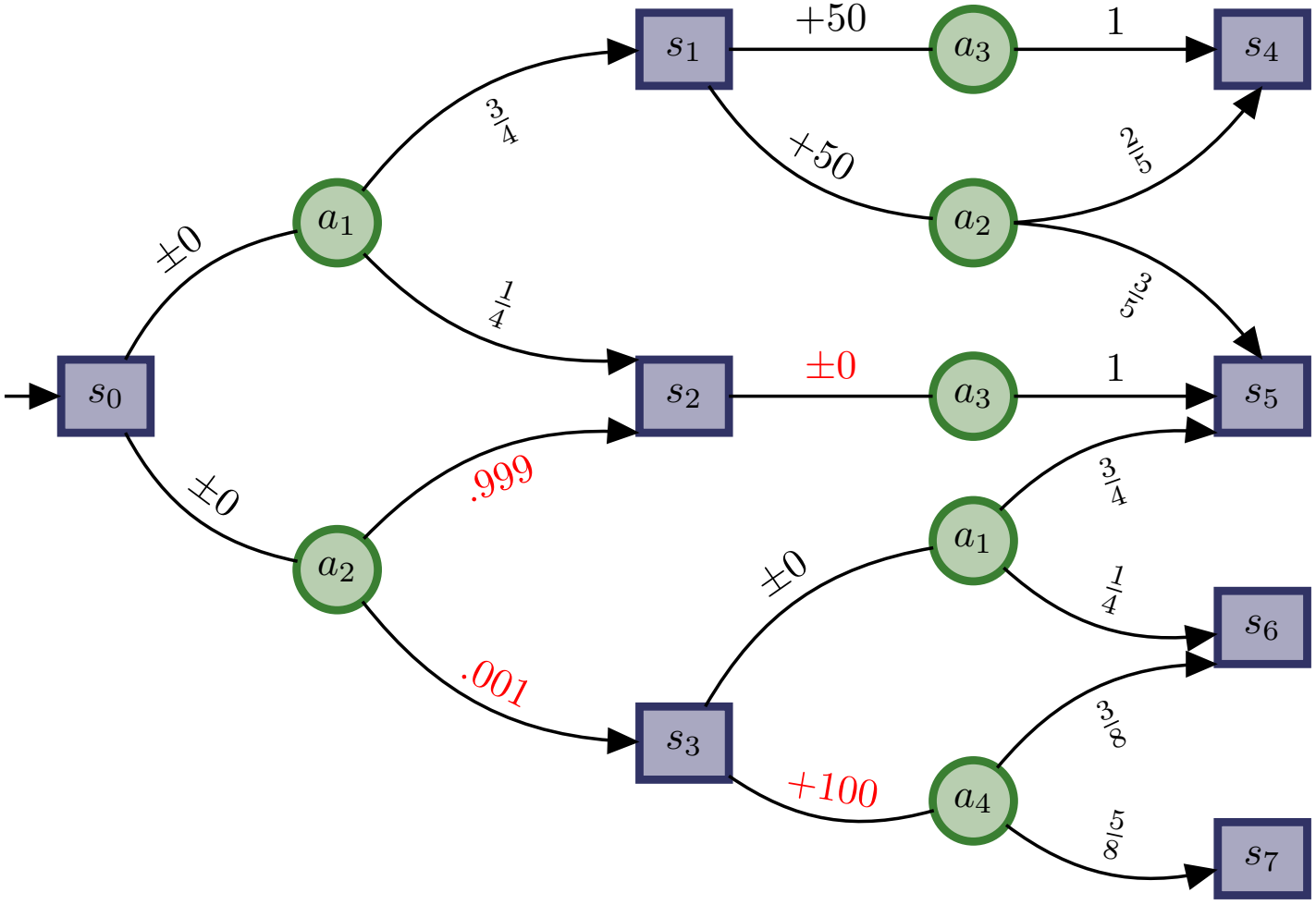
Single-outcome Determinization: Limitations

Important parts of the MDP can become unreachable



All-outcomes Determinization: Limitations

All-outcomes determinization is too optimistic



Determinizations in Probabilistic Planning

in combination with classical planner in plan-execute-monitor cycle approach

- well-suited if uncertainty has certain form (e.g., actions can fail or succeed)
- well-suited if information on probabilities noisy (e.g., path planning for robots in uncertain terrain)

domain-independent implementation: FF-Replan (Yoon, Fern & Givan)

Determinizations in Probabilistic Planning

as subsolver of a more complex system:

- FPG (Buffet and Aberdeen) learns a policy utilizing FF-Replan
- RFF (Teichteil-Königsbuch, Infantes & Kuter) extends determinization-based plan to policy
- PROST-2011 (Keller & Eyerich) and PROST-2014 (Keller & Geißer) use a determinization-based iterative deepening search as heuristic

Monte-Carlo Tree Search: Brief History

- Starting in the 1930s: first researchers experiment with Monte-Carlo methods
- 1998: Ginsberg's GIB player, based on Hindsight Optimization, achieves strong performance playing Bridge
- 2002: Kearns et al. propose Sparse Sampling
- 2002: Auer et al. present UCB1 action selection for multi-armed bandits
- 2006: Coulom coins the term Monte-Carlo Tree Search (MCTS)
- 2006: Kocsis and Szepesvári combine UCB1 and MCTS into the most famous MCTS variant, UCT
- 2007-2016: Constant progress in MCTS-based Go player lead to AlphaGo's defeat of 9-dan Go player Lee Sedol

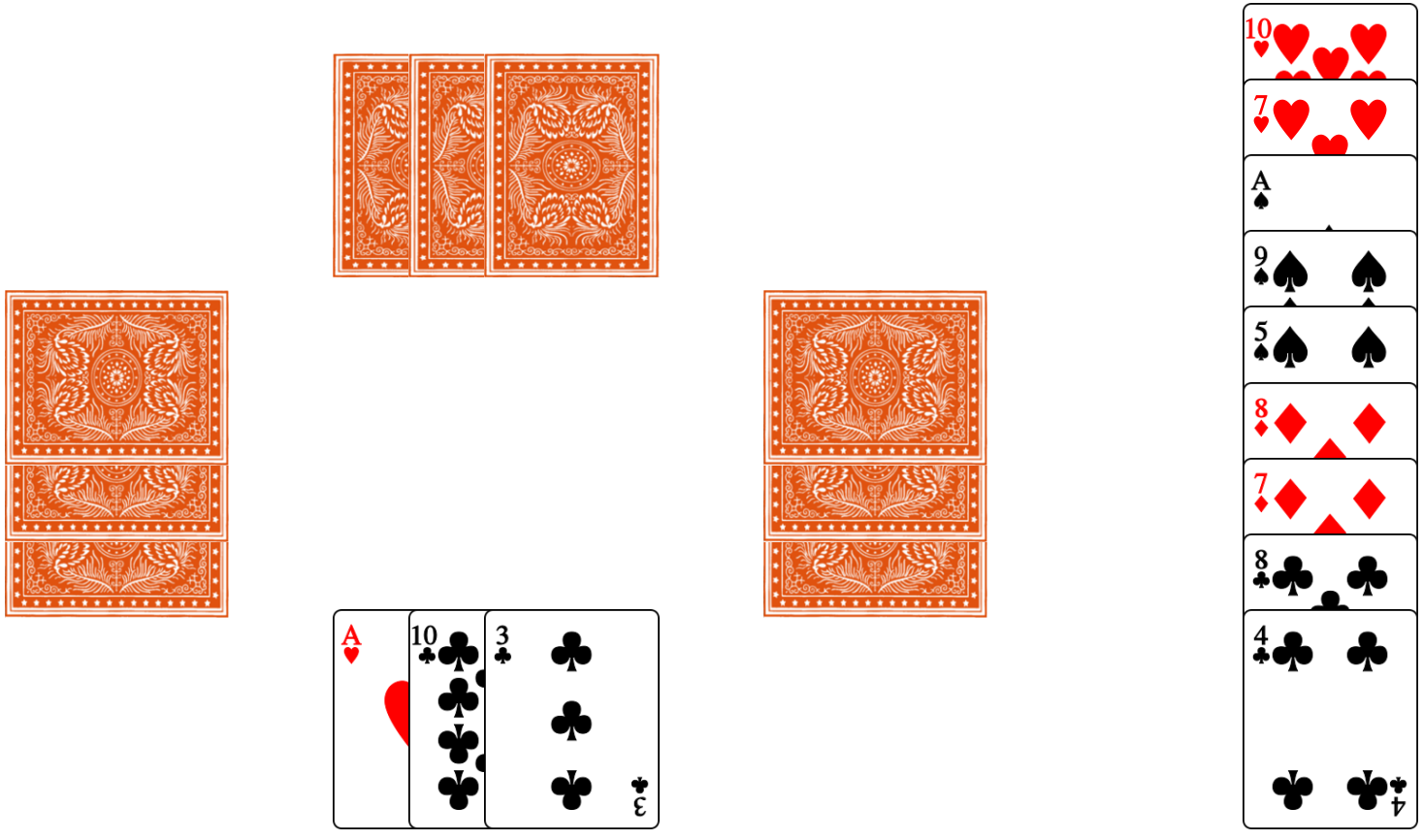
Monte-Carlo Methods: Idea

- subsume a broad family of algorithms
- decisions are based on random samples
- results of samples are aggregated by computing the average
- apart from these points, algorithms differ significantly

Hindsight Optimization: Idea

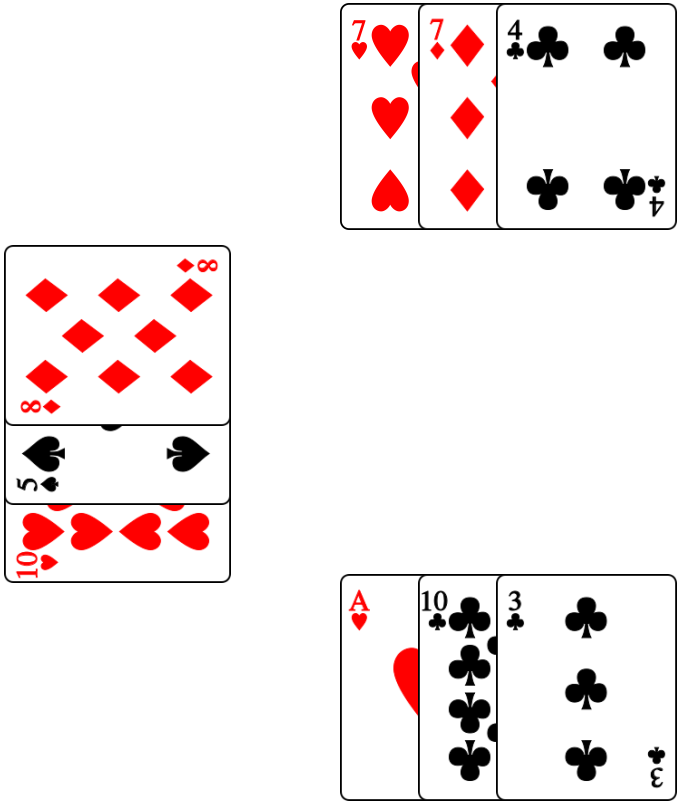
- perform samples as long as resources (deliberation time, memory) allow:
  - sample a determinization of the MDP
  - solve the determinization for each applicable action
  - update average reward $\hat{Q}^{HOP}(s_0, a)$ for each action a with action-value estimate of a in the sample
- execute the action with the highest average reward

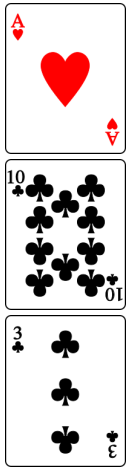
Hindsight Optimization: Example



South to play, three tricks to win, trump suit ♣

Hindsight Optimization: Example





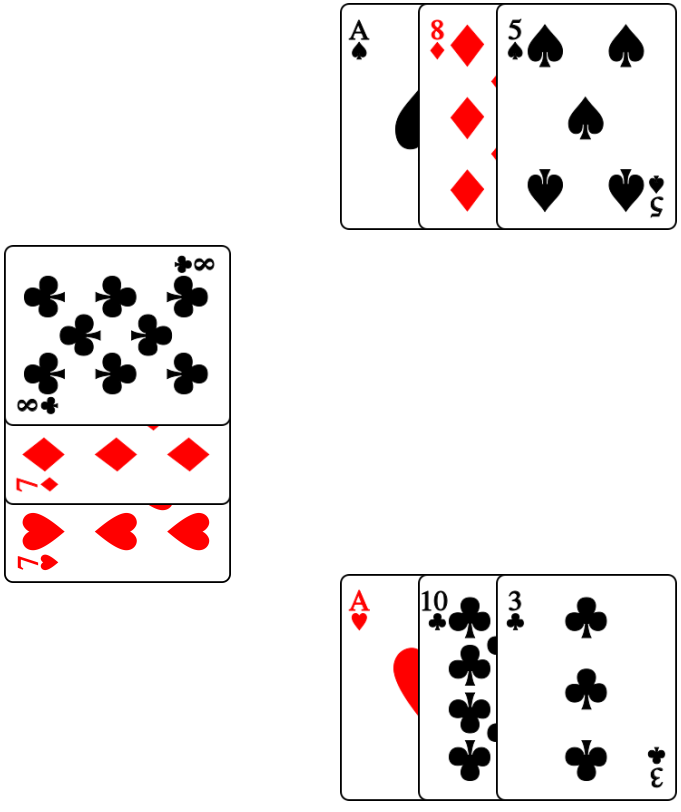
0% (0/1)

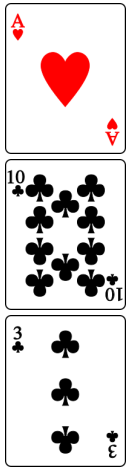
100% (1/1)

0% (0/1)

South to play, three tricks to win, trump suit ♣

Hindsight Optimization: Example





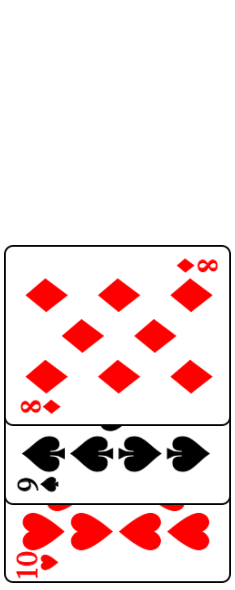
50% (1/2)

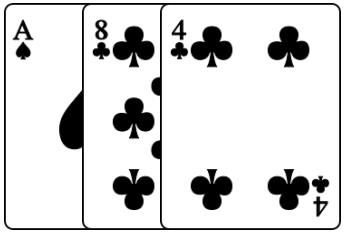
100% (2/2)

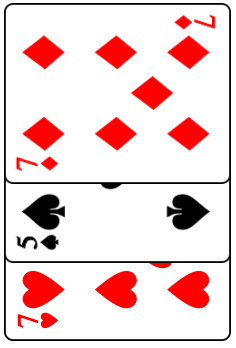
0% (0/2)

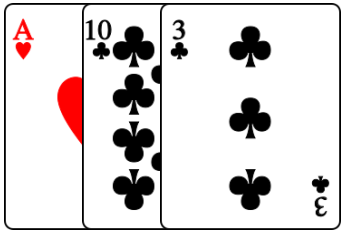
South to play, three tricks to win, trump suit ♣

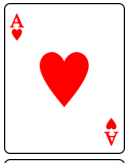
Hindsight Optimization: Example



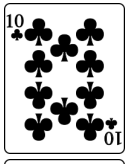




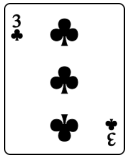




67% (2/3)



100% (3/3)



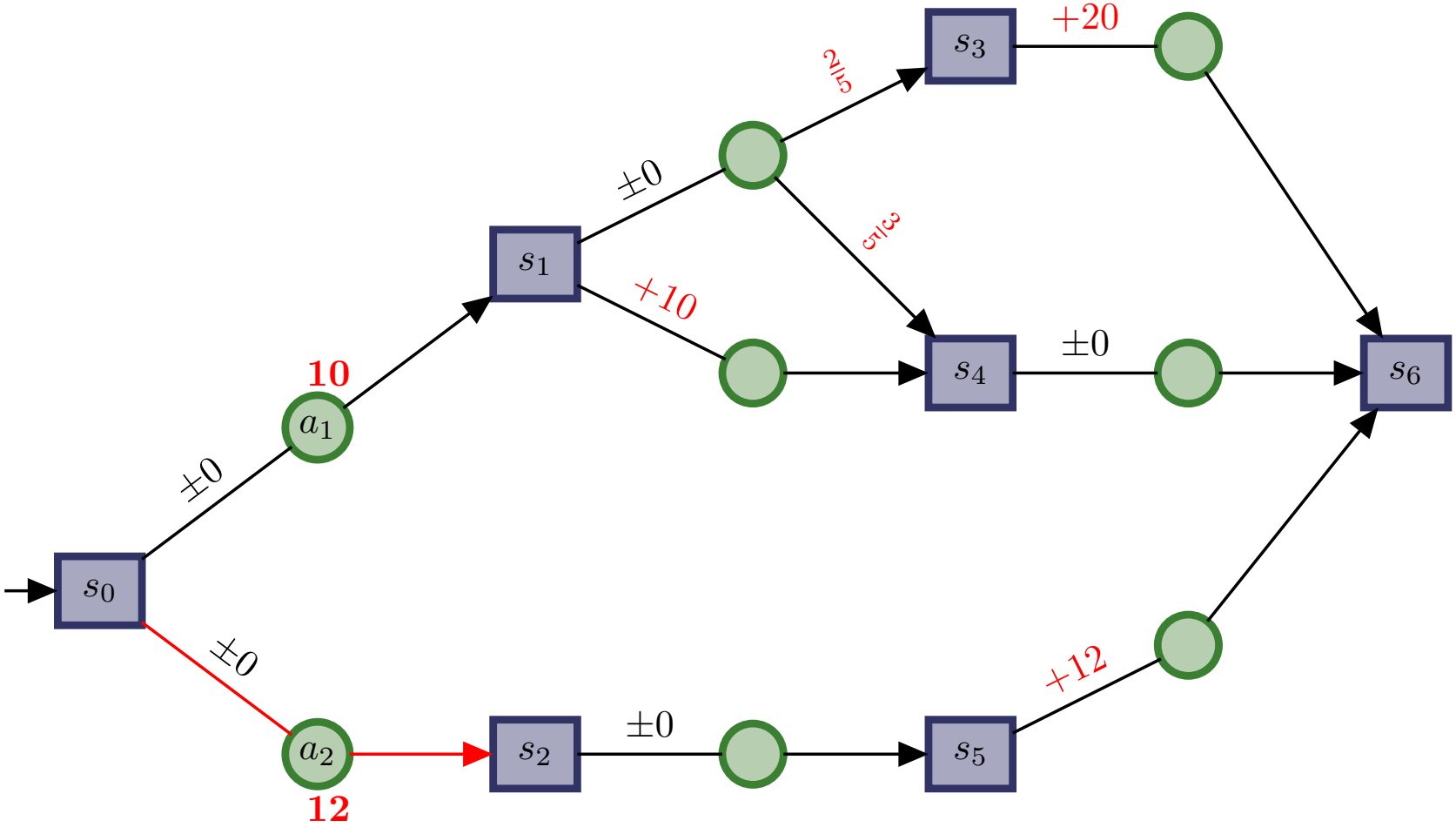
33% (1/3)

South to play, three tricks to win, trump suit ♣

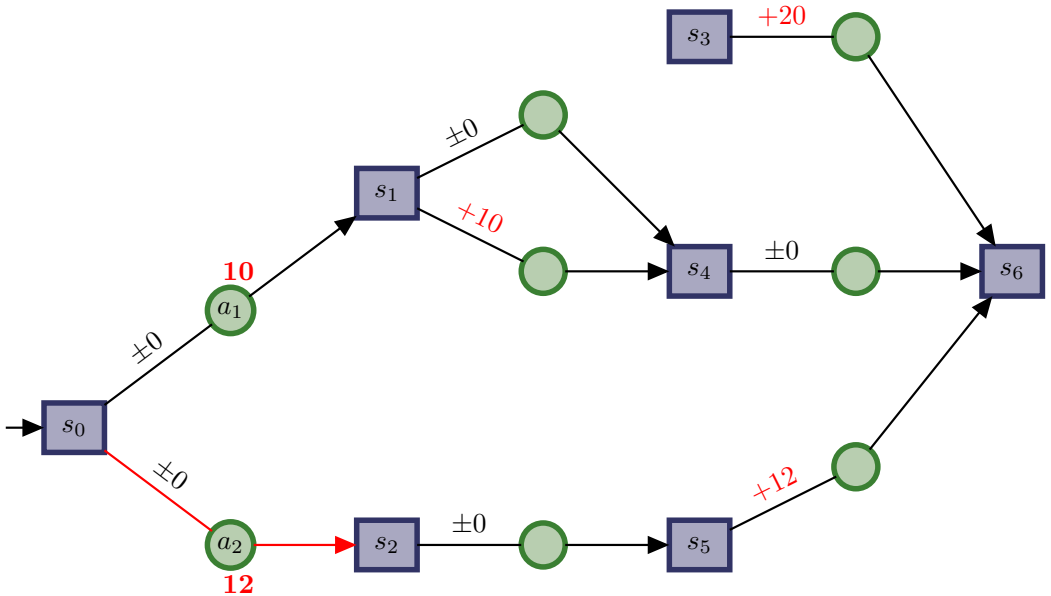
Hindsight Optimization: Evaluation

- HOP well-suited for some problems
- must be possible to solve sampled MDP efficiently:
  - domain-dependent knowledge (various games and MDPs)
  - classical planner (FF-Hindsight)
  - LP solver (Issakkimuthu et al., ICAPS 2015)
- What about optimality with unbounded resources?

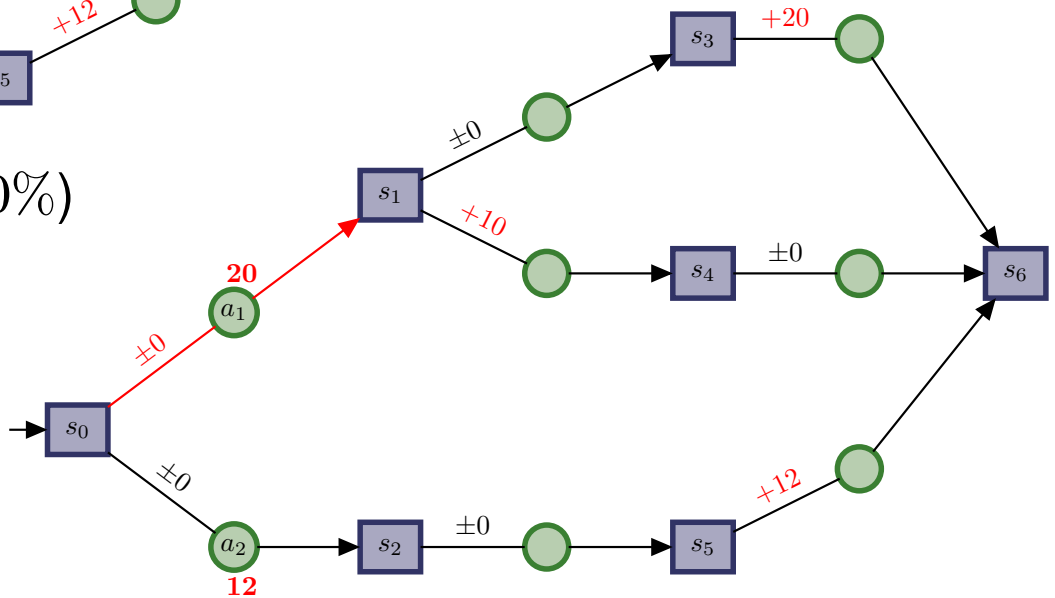
Hindsight Optimization: Optimality



Hindsight Optimization: Optimality

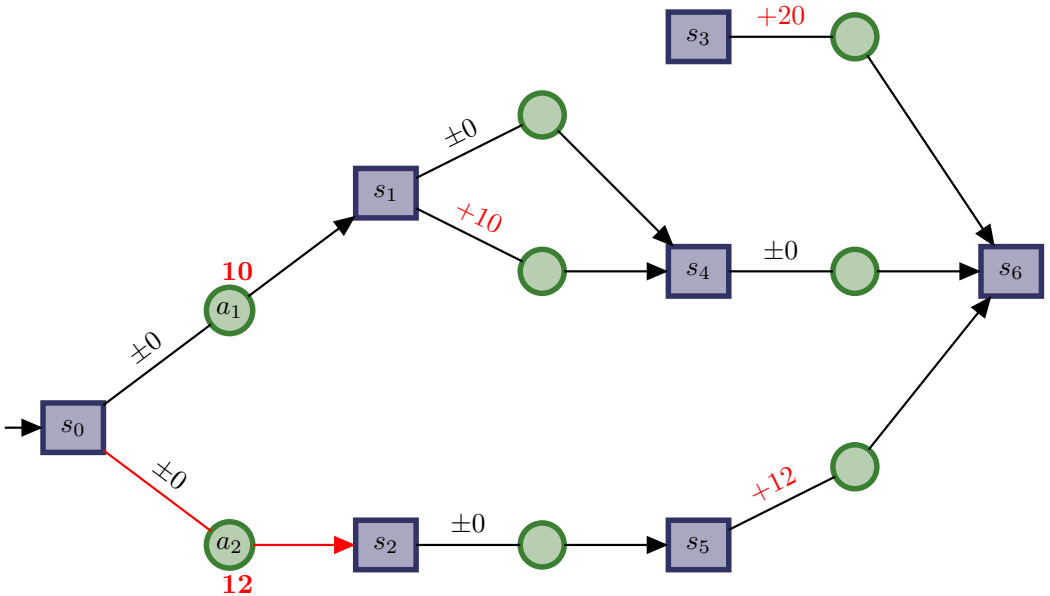


(sample probability: 60%)



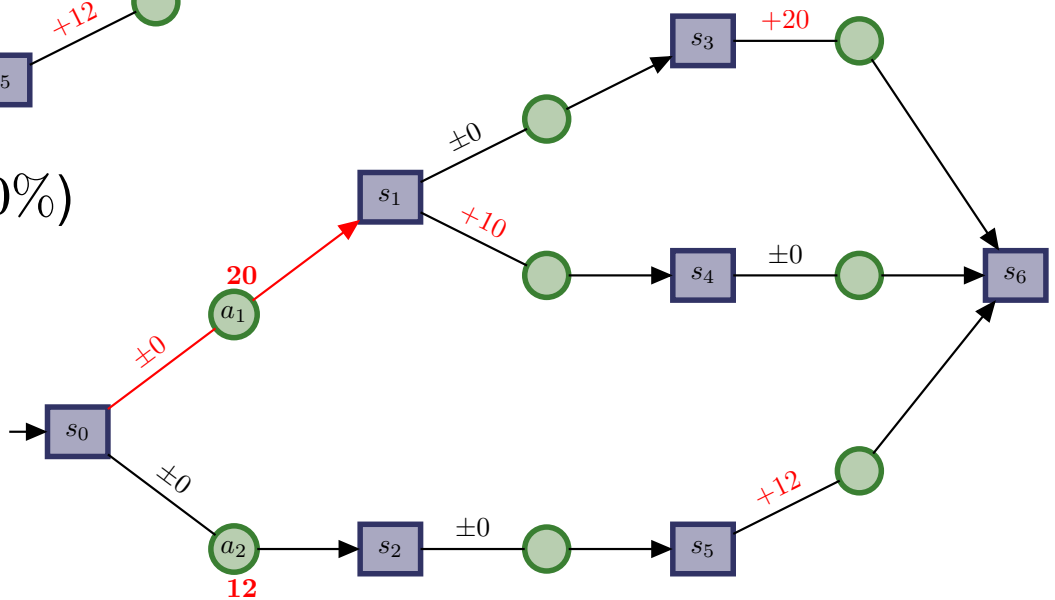
(sample probability: 40%)

Hindsight Optimization: Optimality



(sample probability: 60%)

$$\hat{Q}^{HOP}(s_0, a_1) = 14$$
$$\hat{Q}^{HOP}(s_0, a_2) = 12$$



(sample probability: 40%)

Hindsight Optimization: Evaluation

- HOP well-suited for some problems,
- must be possible to solve sampled MDP efficiently:
  - domain-dependent knowledge (various games and MDPs)
  - classical planner (FF-Hindsight, Yoon et. al, AAAI 2008)
  - LP solver (Issakkimuthu et al., ICAPS 2015)
- What about optimality in the limit?
  $\Rightarrow$ in many problems not optimal due to assumption of clairvoyance

Hindsight Optimization: Clairvoyance

Idea of Hindsight Optimization (Repetition):

- perform samples as long as resources (deliberation time, memory) allow:
  - sample a determinization of the MDP
  - solve the determinization for each applicable action
  - update average reward $\hat{Q}^{HOP}(s_0, a)$ for each action a with action-value estimate of a in the sample
- execute the action with the highest average reward

Hindsight Optimization: Clairvoyance

Idea of Hindsight Optimization (Repetition):

- perform samples as long as resources (deliberation time, memory) allow:
  - sample a determinization of the MDP
  - solve the determinization for each applicable action
  - update average reward $\hat{Q}^{HOP}(s_0, a)$ for each action a with action-value estimate of a in the sample
- execute the action with the highest average reward

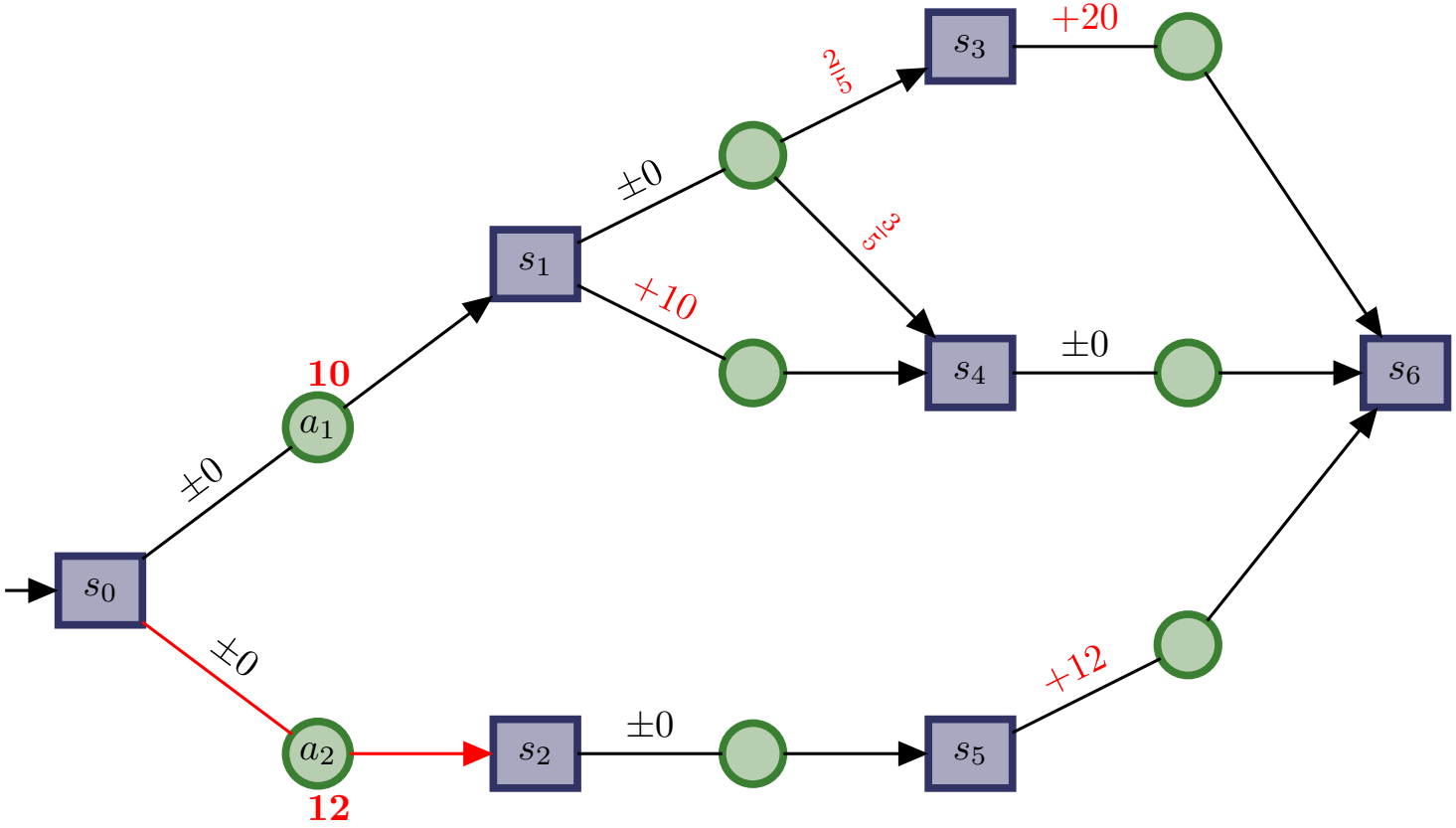
Policy Simulation: Idea

- separate computation of policy and its evaluation
- by simulating the execution of a policy
- any base policy can be used

Optimistic Policy Simulation

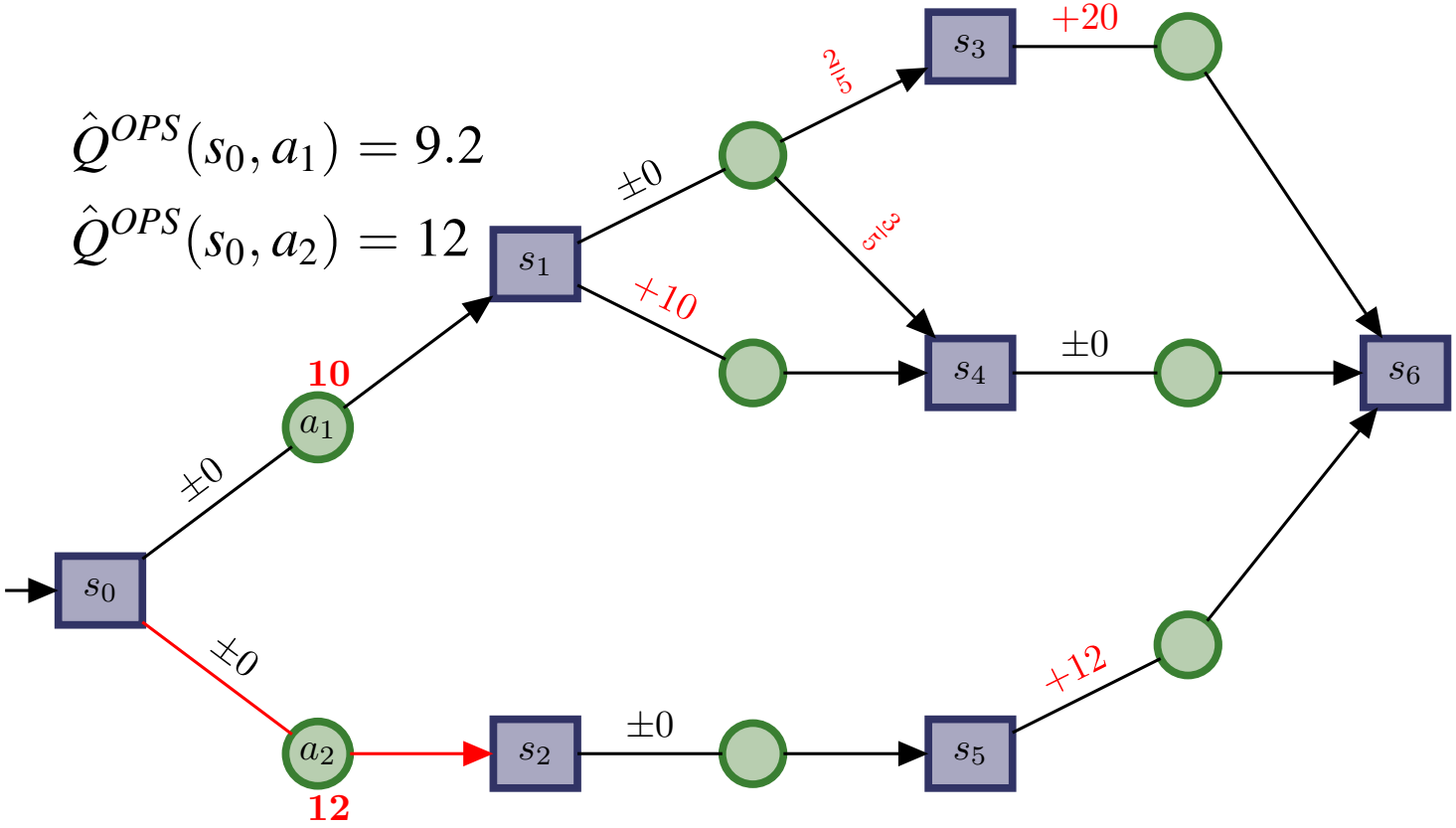
- perform samples as long as resources (deliberation time, memory) allow:
  - sample a determinization of the MDP
  - compute a policy by solving the determinization for each applicable action
  - simulate the policy
  - update average reward $\hat{Q}^{OPS}(s_0, a)$ for each action a with reward in the simulation
- execute the action with the highest average reward

Optimistic Policy Simulation: Example



- $s_1 \rightarrow s_3$ in sample
- $s_1 \rightarrow s_4$ in sample
- $s_1 \rightarrow s_3$ simulation
- $s_1 \rightarrow s_4$ in in simulation

Optimistic Policy Simulation: Example



- $s_1 \rightarrow s_3$ in sample
- $s_1 \rightarrow s_4$ in sample
- $s_1 \rightarrow s_3$ simulation
- $s_1 \rightarrow s_4$ in in simulation

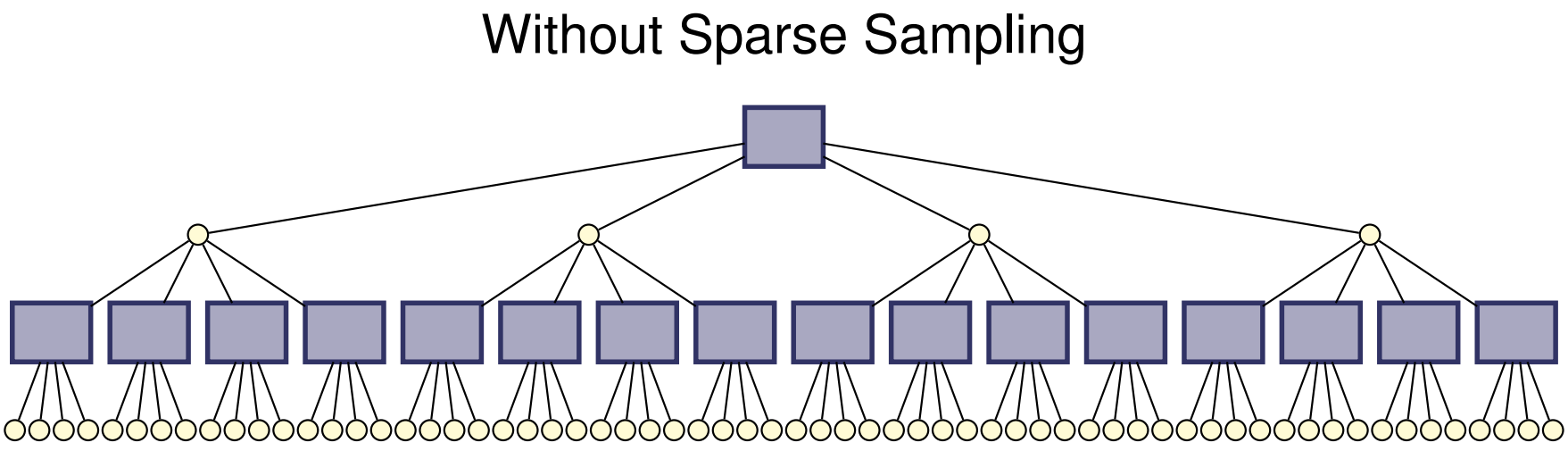
Optimistic Policy Simulation: Evaluation

- Problem: suboptimal base policy is static
- no mechansim to overcome weakness of base policy
- $\Rightarrow$ repeated suboptimal decisions in simulation affect Policy Simulation

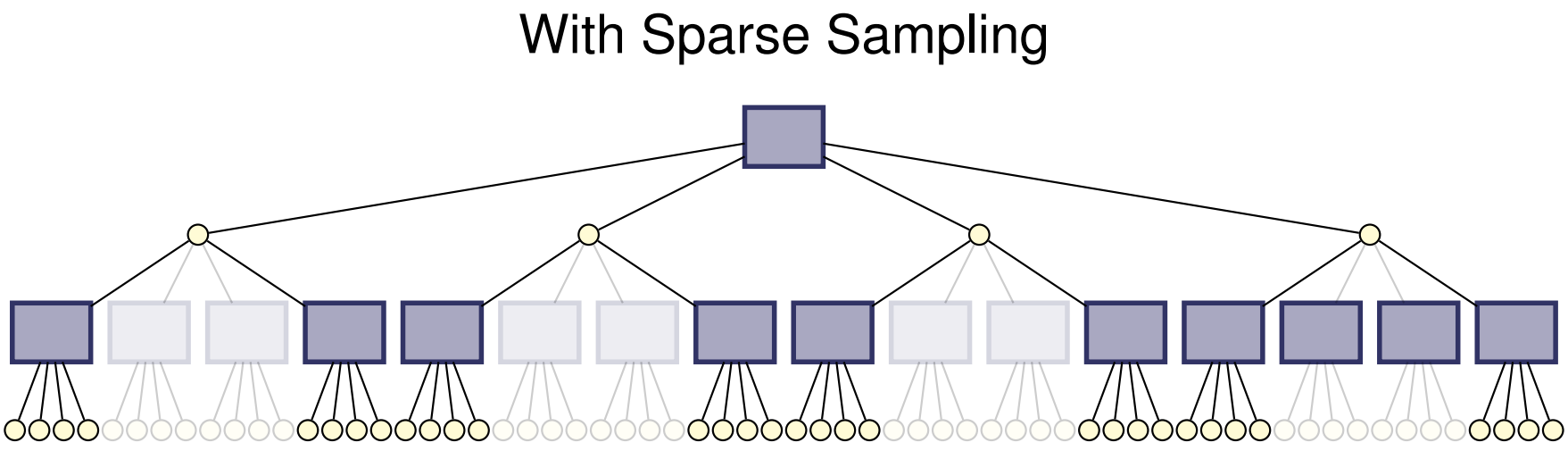
Sparse Sampling: Idea

- rather than restrict the simulated actions (as in policy simulation), restrict the simulated outcomes
- search tree creation: sample a constant number of outcomes according to their probability in each state and ignore the rest
- near-optimal: utility of resulting policy close to utility of optimal policy
- runtime independent from the number of states

Sparse Sampling: Search Tree



Sparse Sampling: Search Tree



Sparse Sampling: Problems

- independent from number of states, but still exponential in horizon
- constant that gives the number of outcomes large for good bounds on near-optimality
- search time difficult to predict
- tree is symmetric $\Rightarrow$ resources are wasted in non-promising parts of the tree

Summary

- presented several algorithms for probabilistic planning:
  - Determinizations
  - Hindsight Optimization
  - Policy Sampling
  - Sparse Sampling
- there are applications for all where they perform well
- but all are suboptimal even if provided with unlimited resources