Bayes-Optimal Reinforcement Learning

Emma Brunskill
15-889e
Fall 2015

Sample Efficient RL

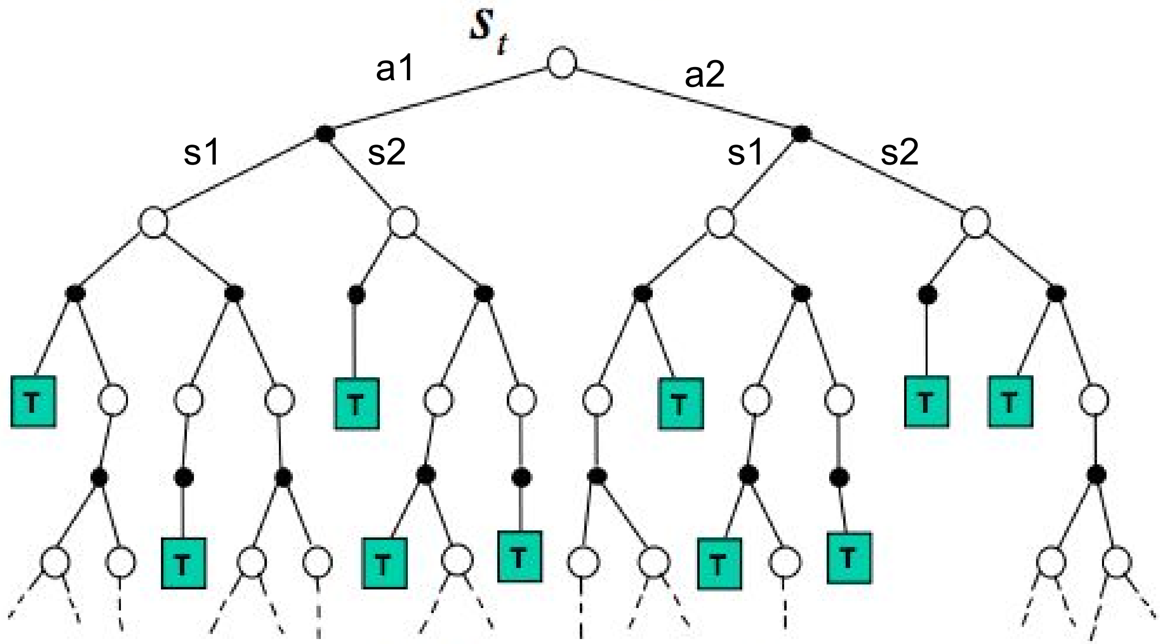
- Probably Approximately Correct
- Minimizing regret
- **Today: Bayes-optimal RL**

Overview

- Quick intro to or refresher of POMDPs
  - Definition
  - Belief state tracking
  - Online planning
- Bayes-optimal bandits
- Bayes-optimal RL

Review: MDP Forward Search w/Generative Model

- Forward search algorithms select the best action by lookahead
- They build a search tree with the current state s_t at the root
- Using a model of the MDP to look ahead

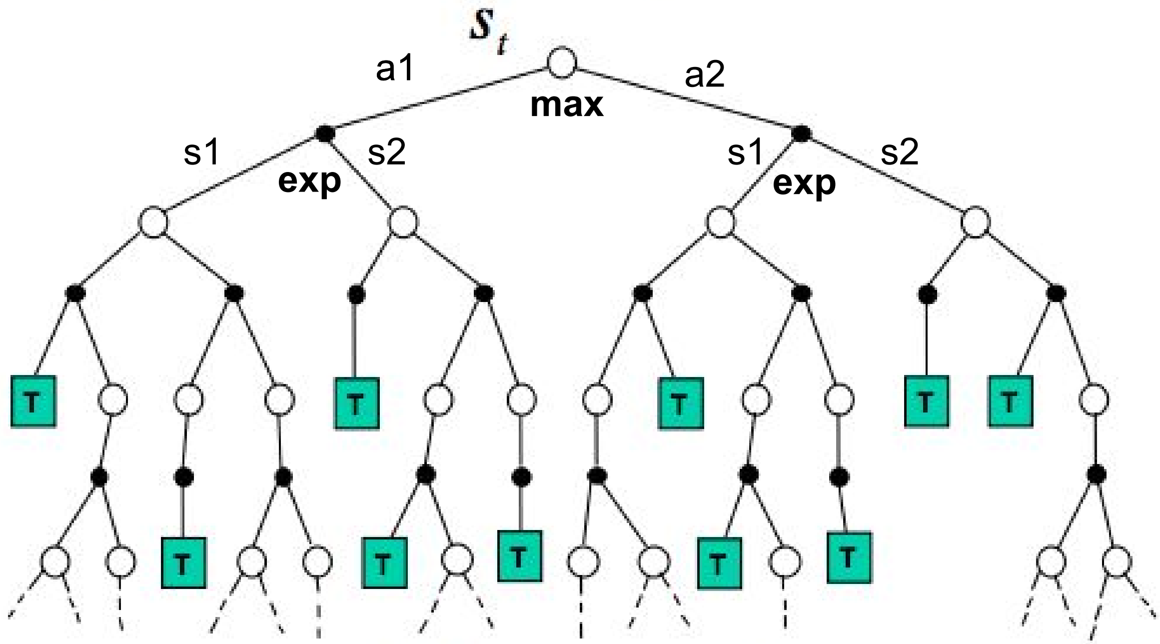


- No need to solve whole MDP, just sub-MDP starting from now

Slide modified from David Silver

Exact/Exhaustive Forward Search: $(|S||A|)^H$ Nodes

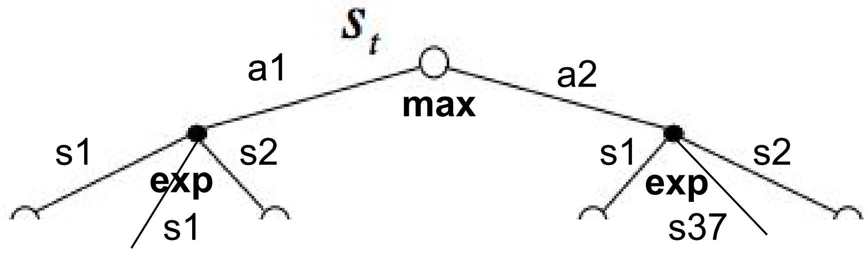
- Forward search algorithms select the best action by lookahead
- They build a search tree with the current state s_t at the root
- Using a model of the MDP to look ahead



- No need to solve whole MDP, just sub-MDP starting from now

Sparse Sampling: Don't Enumerate All Next States, Instead Sample Next States s' ~ P(s'|s,a)

- Forward search algorithms select the best action by lookahead
- They build a search tree with the current state s_t at the root
- Using a model of the MDP to look ahead



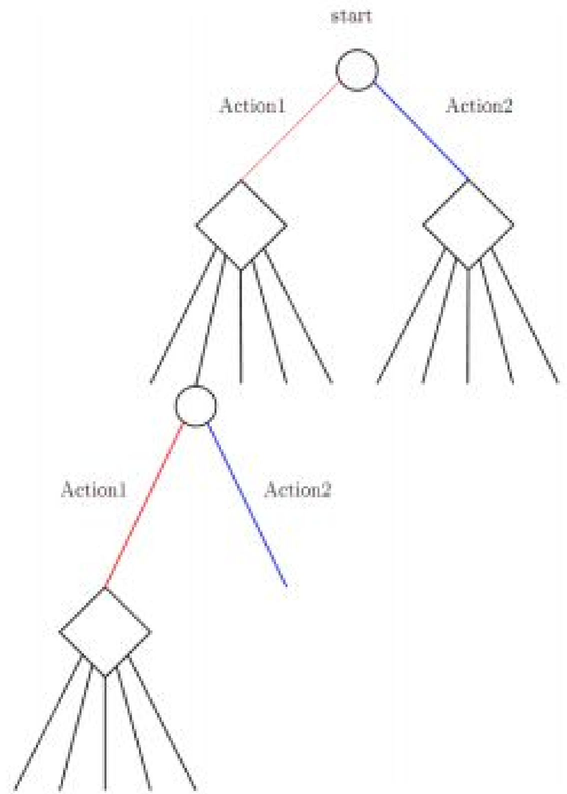
Sample n next states, s_i ~ P(s'|s,a)

Compute (1/n) Sum_i V(s_i)

Converges to expected future reward: $Sum_{s'}$ p(s'|s,a)V(s')

Limitation of Sparse Sampling

- Sparse sampling wastes time on bad parts of tree
  - Devotes equal resources to each state encountered in the tree
  - Would like to focus on most promising parts of tree
- But how to control exploration of new parts of tree vs. exploiting promising parts?



Monte Carlo Tree Search

Combine ideas of sparse sampling with an adaptive method for focusing on more promising parts of the ree

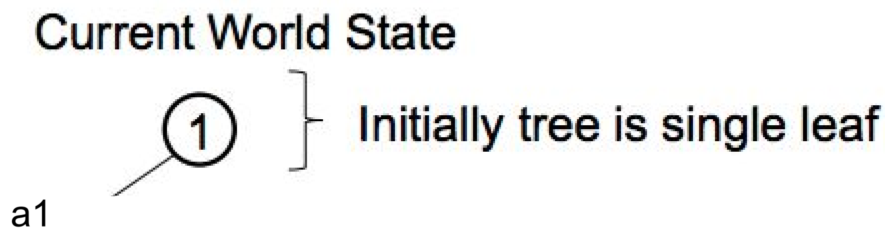
Here “more promising” means the actions that are seem likely to yield higher long term reward

Upper Confidence Tree (UCT)

[Kocsis & Szepesvari, 2006]

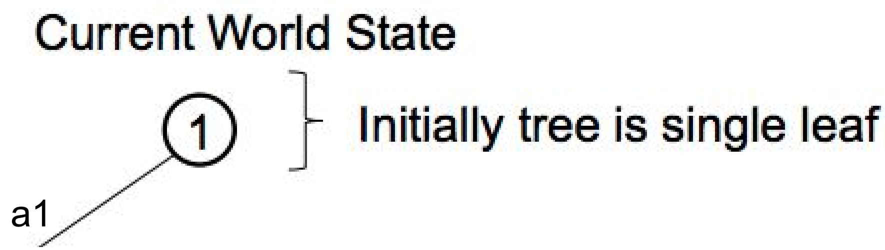
- Combine forward search and simulation search
- Instance of Monte-Carlo Tree Search
  - Repeated Monte Carlo simulation of rollout policy
  - Rollouts add one or more nodes to search tree
- UCT
  - Uses optimism under uncertainty idea
  - Some nice theoretical properties
  - Much better realtime performance than sparse sampling

Slide modified from Alan Fern

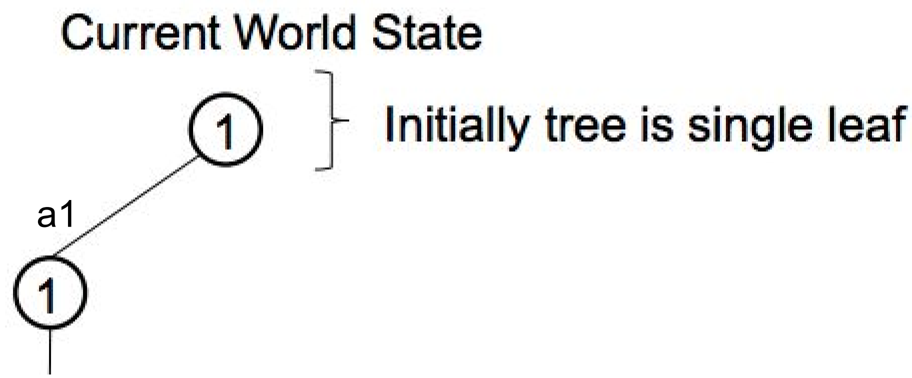


- Set desired max tree depth (e.g. H)
- Select any action a1 haven't tried from leaf state s
- Sample next reward and state s' given p(s'|s,a1) & r(s,a)
- For the remainder of tree (H - depth of leaf)
  - Use rollout policy π to simulate a trajectory
  - s',π(s'),r',s'',π(s''), …

  where next states and rewards are sampled from transition and reward model given current state s'' and π(s'') etc.
  - Sum up rewards, and this is a sample of the return of following π from the leaf state given action a1



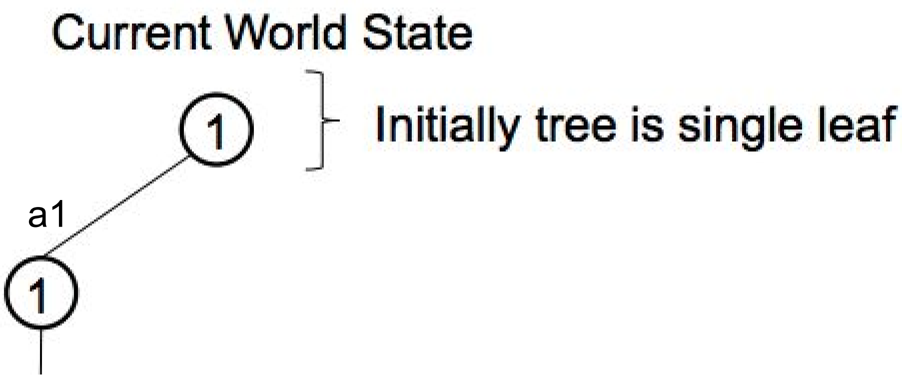
E.g. sample H more steps using π, got 0 reward for all steps except the final step where got a 1



Set expected reward for a1 for leaf node to be average of all returns from rollout policy

Only 1 sample and its return was 1

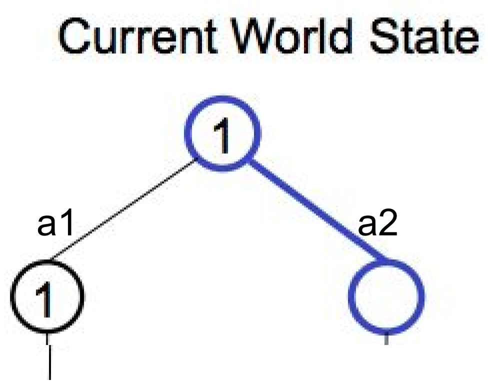
So set its expected value to 1



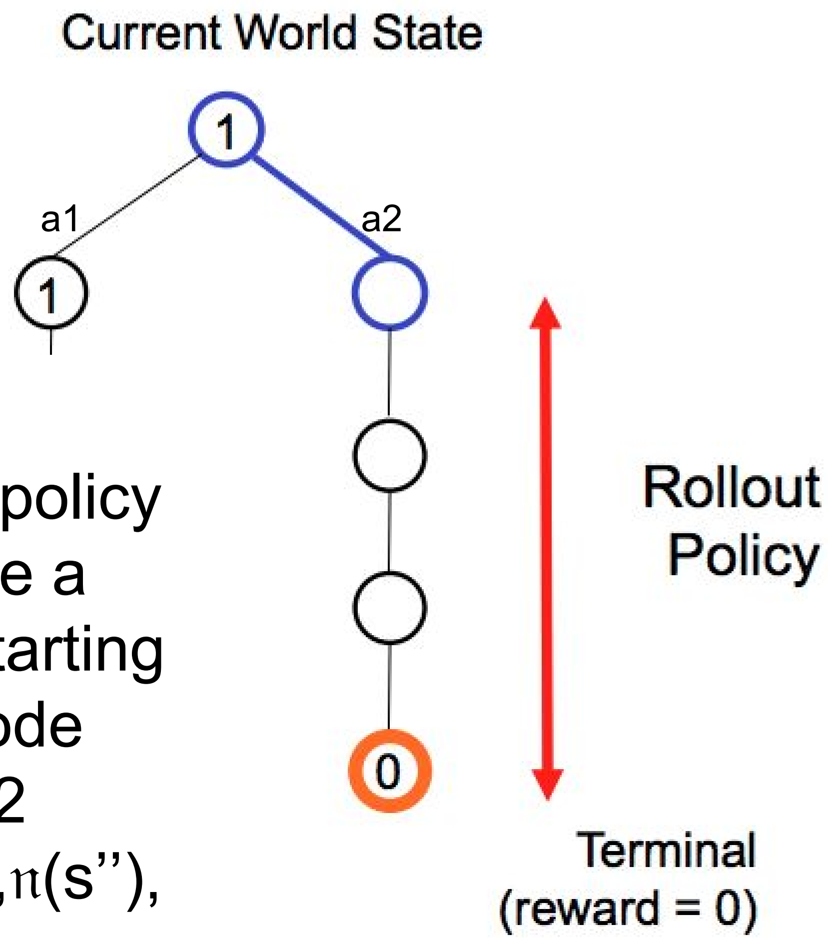
Next time reach this leaf node, check if all actions have been sampled at least once

If not, select an action that hasn't yet been expanded

Must select each action at a node at least once

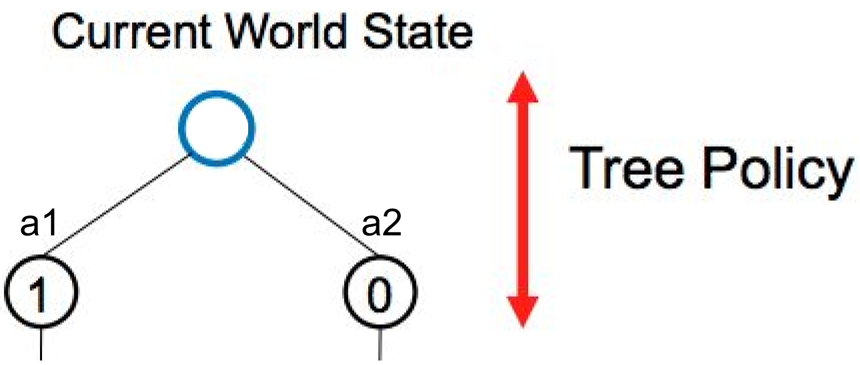


Must select each action at a node at least once



- Use rollout policy π to simulate a trajectory starting from leaf node state and a2
- s’,π(s’),r’,s’’,π(s’’), …

When all node actions tried once, select action according to tree policy



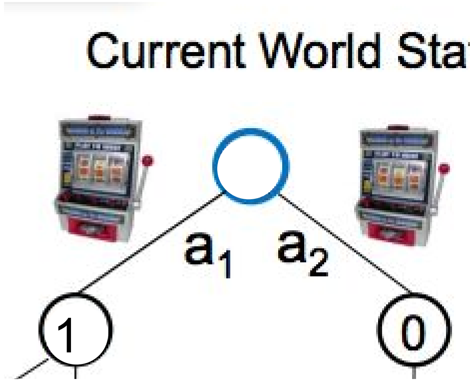
UCT Algorithm [Kocsis & Szepesvari, 2006]

- Basic UCT uses random rollout policy

- Tree policy is based on UCB: (Upper Confidence Bound)
  - Q(s,a) : average reward received in current trajectories after taking action a in state s
  - n(s,a) : number of times action a taken in s
  - n(s) : number of times state s encountered

$$\pi_{UCT}(s) = \arg\max_a Q(s,a) + c\sqrt{\frac{\ln n(s)}{n(s,a)}}$$

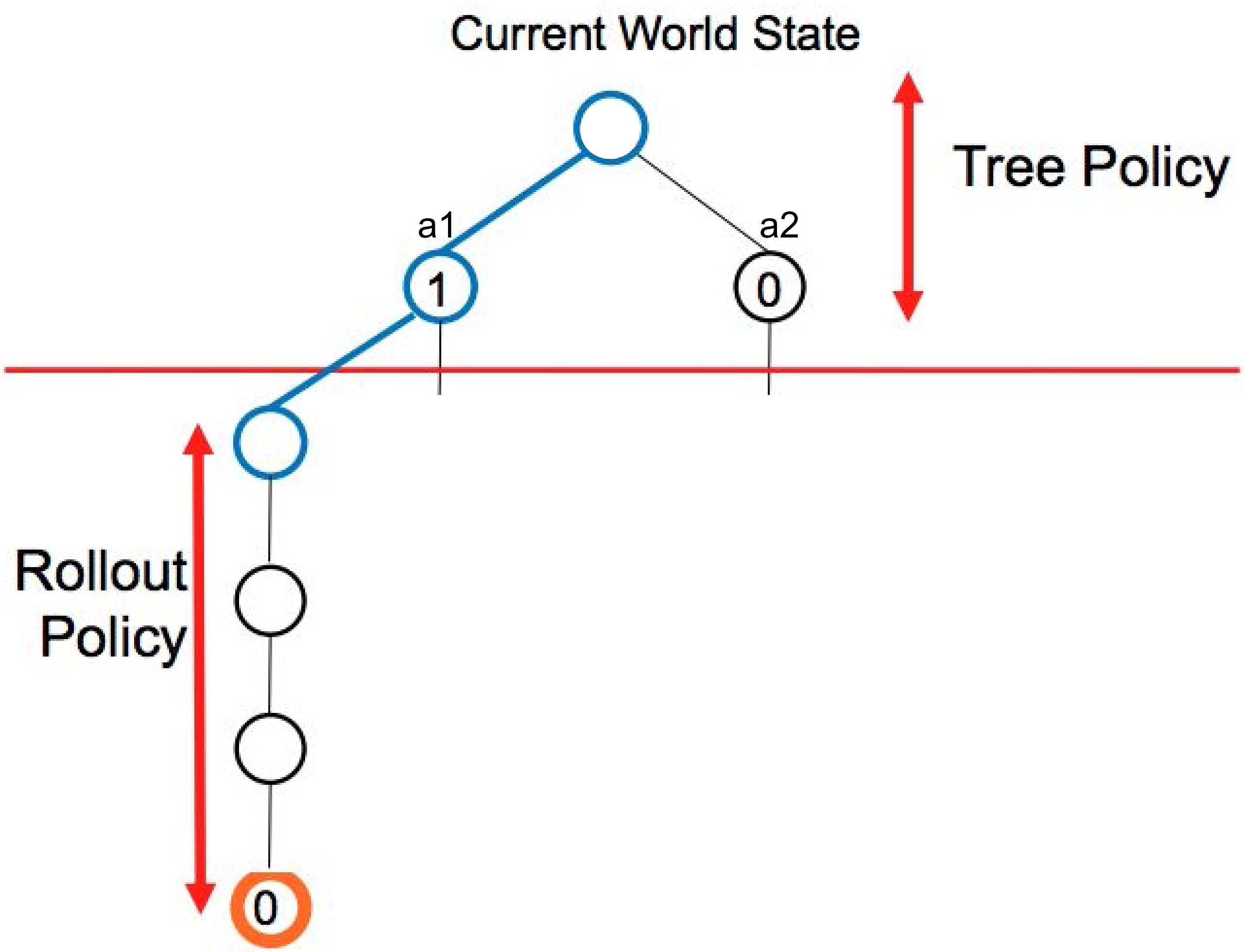
Theoretical constant that must be selected empirically in practice



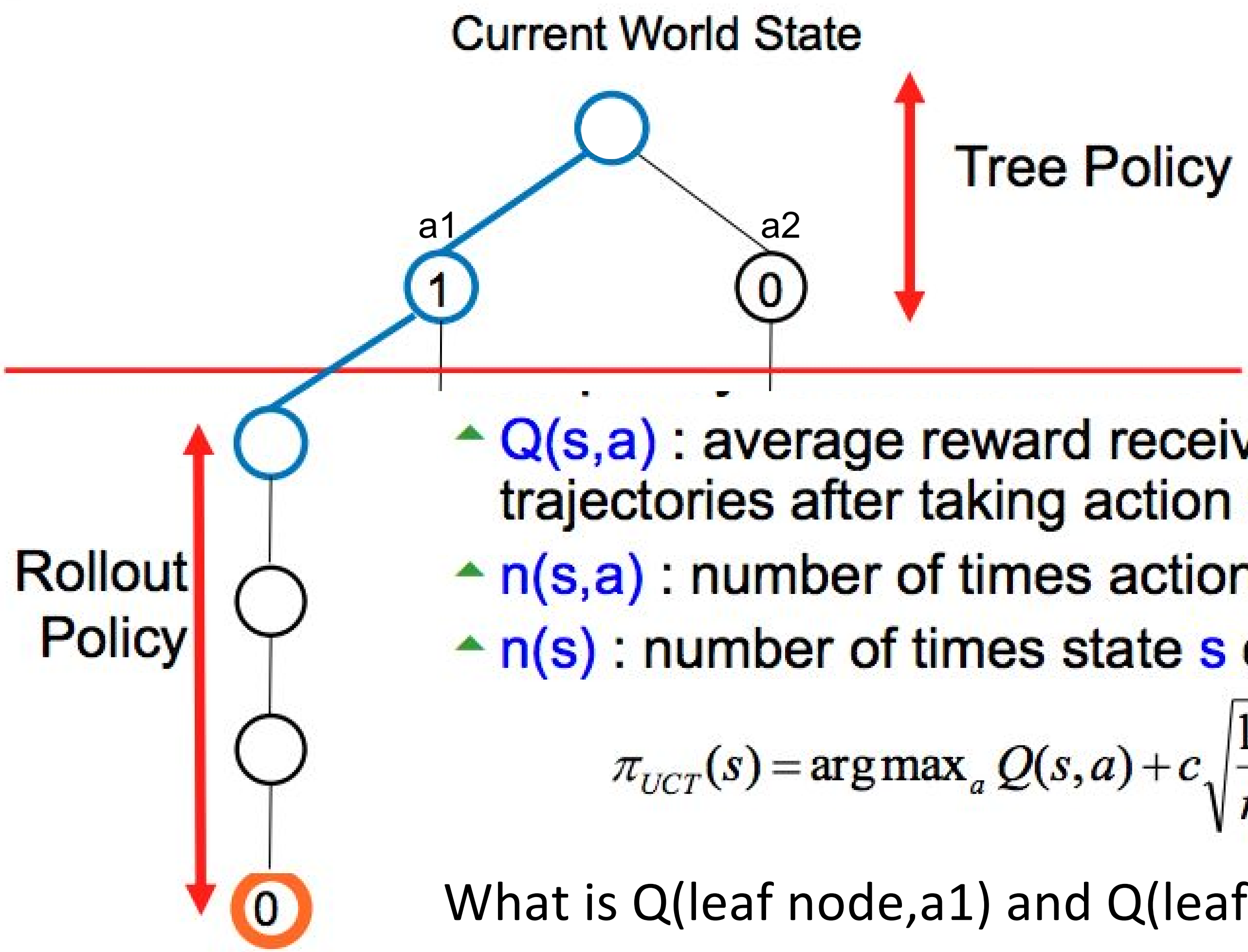
$$\pi_{UCT}(s) = \arg\max_a Q(s,a) + c\sqrt{\frac{\ln n(s)}{n(s,a)}}$$

- Which action would we choose?

When all node actions tried once, select action according to tree policy



When all node actions tried once, select action according to tree policy



- Q(s,a) : average reward received in current trajectories after taking action a in state s
- n(s,a) : number of times action a taken in s
- n(s) : number of times state s encountered

$$\pi_{UCT}(s) = \arg\max_a Q(s,a) + c\sqrt{\frac{\ln n(s)}{n(s,a)}}$$

What is Q(leaf node,a1) and Q(leaf node, a2)?

if c=5 which action will be chosen next time at the leaf node?

UCT

- Eventually converges to the optimal value of Q(s,a) for the root state
- At that point, or when run out of computation time, choose best action at root node
- Take action, get next state from environment
- Repeat UCT planning at new state
- Empirically often does extremely well (e.g. the game of Go)

MCTS / UCT for POMDP Planning

- States → histories / belief states
- Sample observations instead of states
- Rollout policy needs to be based on histories/belief states
- Other than that, can apply directly
- Further optimizations possible, see “Monte-Carlo Planning in Large POMDPs” Silver & Veness NIPS 2010
  - Even computing belief updates may be too expensive in some domains, so use sampling

Algorithm 1 Partially Observable Monte-Carlo Planning

procedure SEARCH(h)
  repeat
    if $h = empty$ **then**
      $s \sim \mathcal{I}$
    else
      $s \sim B(h)$
    end if
    SIMULATE($s, h, 0$)
  until TIMEOUT()
  return $\underset{b}{\text{argmax}}\ V(hb)$
end procedure

procedure ROLLOUT($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  $a \sim \pi_{rollout}(h, \cdot)$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  return $\mathrm{r} + \gamma.$ROLLOUT($s', hao, depth{+}1$)
end procedure

procedure SIMULATE($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  if $h \notin T$ **then**
    for all $a \in \mathcal{A}$ **do**
      $T(ha) \leftarrow (N_{init}(ha), V_{init}(ha), \emptyset)$
    end for
    return ROLLOUT($s, h, depth$)
  end if
  $a \leftarrow \underset{b}{\text{argmax}}\ V(hb) + c\sqrt{\frac{\log N(h)}{N(hb)}}$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  $R \leftarrow r + \gamma.$SIMULATE($s', hao, depth + 1$)
  $B(h) \leftarrow B(h) \cup \{s\}$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$
  return R
end procedure

Figure from Silver & Veness NIPS 2010

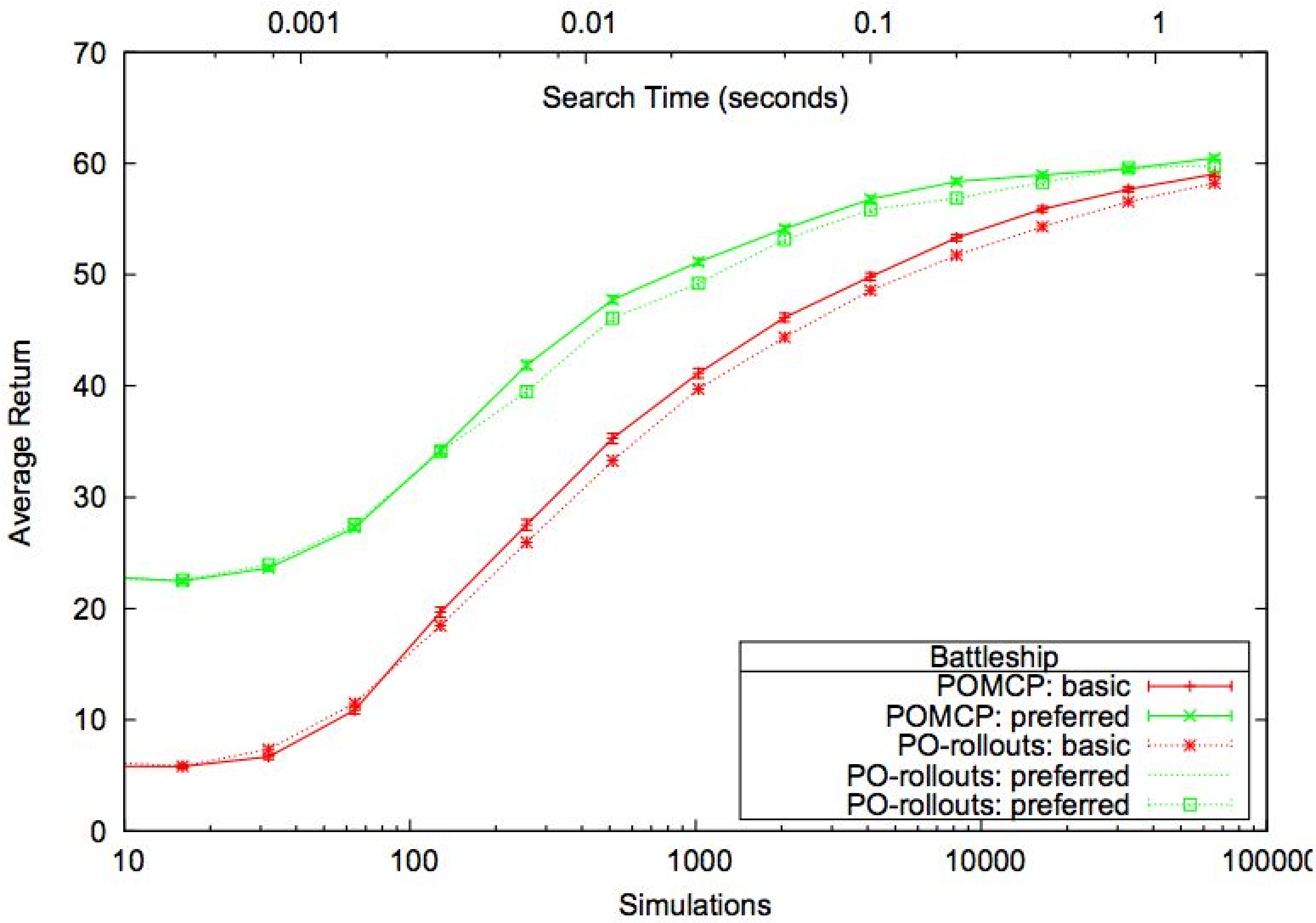


Figure from Silver & Veness
NIPS 2010

Overview

- Quick intro to or refresher of POMDPs
  - Definition
  - Belief state tracking
  - Online planning
- **Bayes-optimal bandits**
- Bayes-optimal RL

Bayes-Optimality

- Know get to act for H steps (could be infinite)
- Take actions to **precisely maximize expected reward over H steps given initial uncertainty over model** (bandit, MDP) parameters
- Reasons directly about value of information
    - If explored more, could that change decisions made?

Bayes Optimal Bandits

- We have viewed bandits as *one-step* decision-making problems
- Can also view as *sequential* decision-making problems
- At each step there is an *information state* $\tilde{s}$
  - $\tilde{s}$ is a statistic of the history, $\tilde{s}_t = f(h_t)$
  - summarising all information accumulated so far
- Each action a causes a transition to a new information state $\tilde{s}'$ (by adding information), with probability $\tilde{\mathcal{P}}^a_{\tilde{s},\tilde{s}'}$
- This defines MDP $\tilde{\mathcal{M}}$ in augmented information state space

$$\tilde{\mathcal{M}} = \langle \tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{P}}, \mathcal{R}, \gamma \rangle$$

Bernoulli Bandits

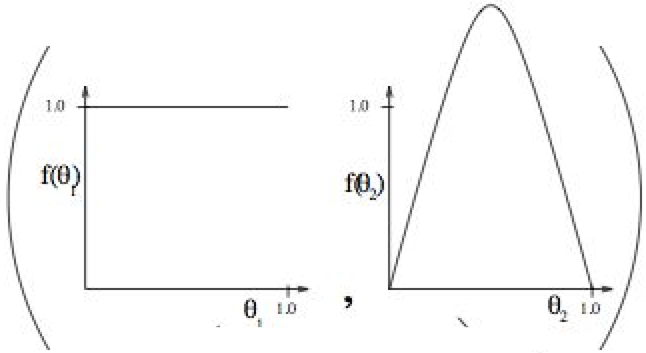
- Consider a Bernoulli bandit, such that $\mathcal{R}^a = \mathcal{B}(\mu_a)$
- e.g. Win or lose a game with probability μ_a
- Want to find which arm has the highest μ_a
- The information state is $\tilde{s} = \langle \alpha, \beta \rangle$
  - α_a counts the pulls of arm a where reward was 0
  - β_a counts the pulls of arm a where reward was 1

Solving Information State Bandits

- Challenge: Number of information states can be infinite
- But can treat as a (really large) MDP planning problem

Bayes-Adaptive Bernoulli Bandits

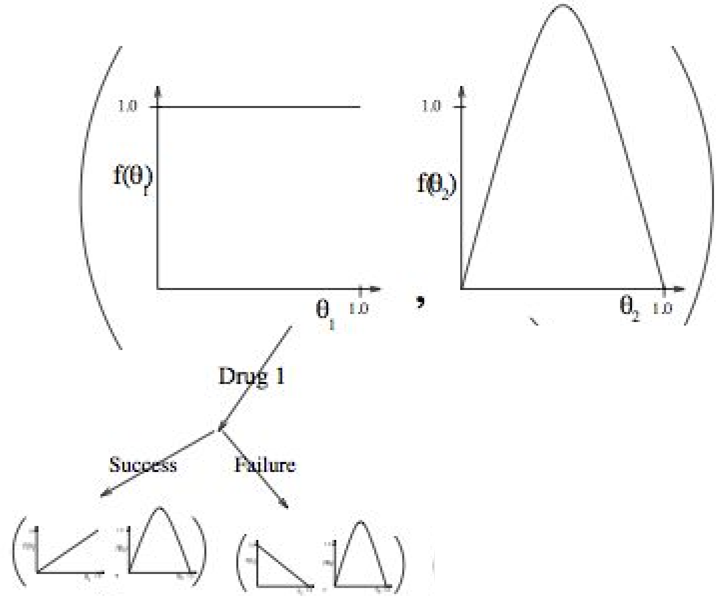
- Start with $Beta(\alpha_a, \beta_a)$ prior over reward function $\mathcal{R}^a$
- Each time a is selected, update posterior for $\mathcal{R}^a$
  - $Beta(\alpha_a + 1, \beta_a)$ if $r = 0$
  - $Beta(\alpha_a, \beta_a + 1)$ if $r = 1$



Arm 1 (Drug 1) Arm 2 (Drug 2)

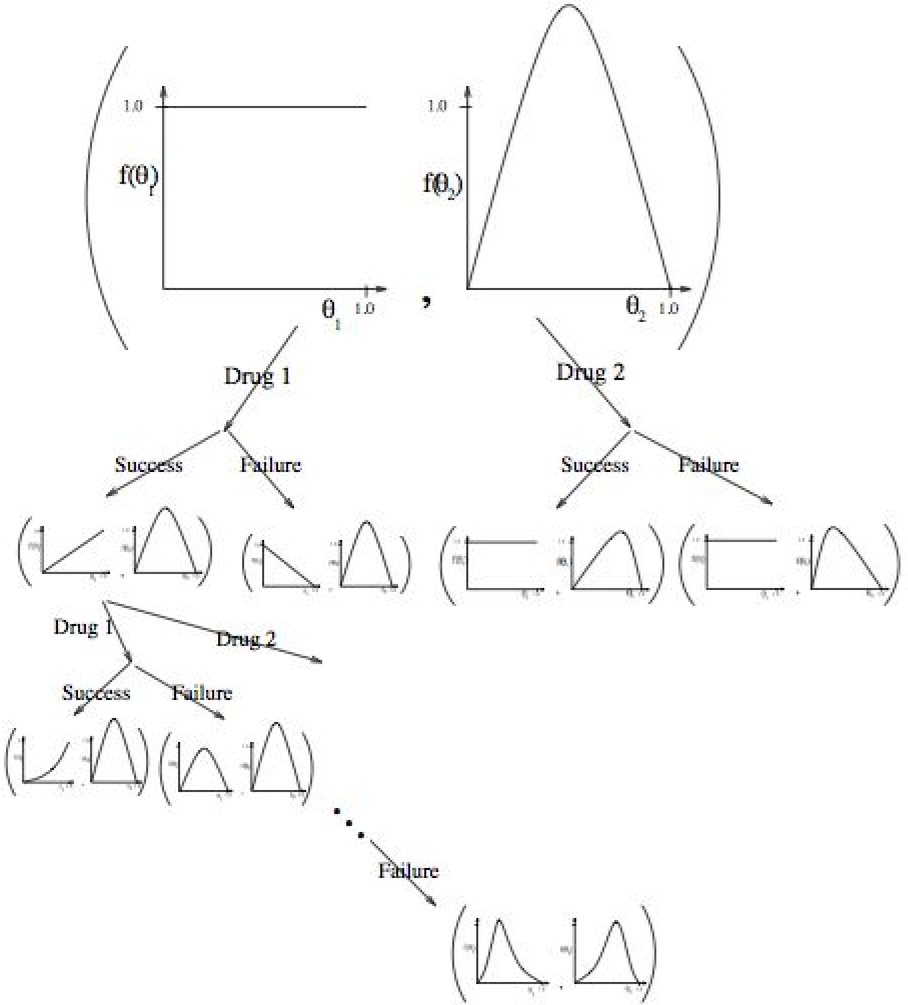
Bayes-Adaptive Bernoulli Bandits

- Start with $Beta(\alpha_a, \beta_a)$ prior over reward function $\mathcal{R}^a$
- Each time a is selected, update posterior for $\mathcal{R}^a$
  - $Beta(\alpha_a + 1, \beta_a)$ if $r = 0$
  - $Beta(\alpha_a, \beta_a + 1)$ if $r = 1$
- This defines transition function $\tilde{\mathcal{P}}$ for the Bayes-adaptive MDP
- Information state $\langle \alpha, \beta \rangle$ corresponds to reward model $Beta(\alpha, \beta)$
- Each state transition corresponds to a Bayesian model update

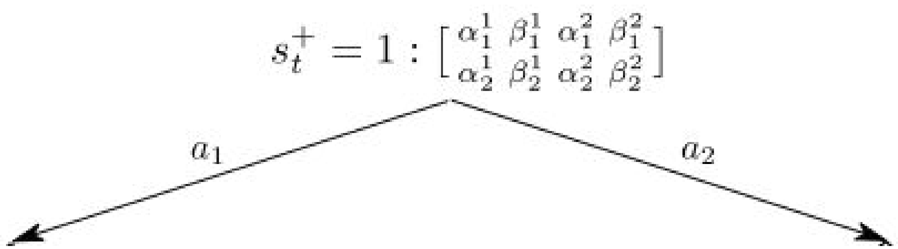


Bayes-Adaptive Bernoulli Bandits

- Start with $Beta(\alpha_a, \beta_a)$ prior over reward function $\mathcal{R}^a$
- Each time a is selected, update posterior for $\mathcal{R}^a$
  - $Beta(\alpha_a + 1, \beta_a)$ if $r = 0$
  - $Beta(\alpha_a, \beta_a + 1)$ if $r = 1$
- This defines transition function $\tilde{\mathcal{P}}$ for the Bayes-adaptive MDP
- Information state $\langle \alpha, \beta \rangle$ corresponds to reward model $Beta(\alpha, \beta)$
- Each state transition corresponds to a Bayesian model update

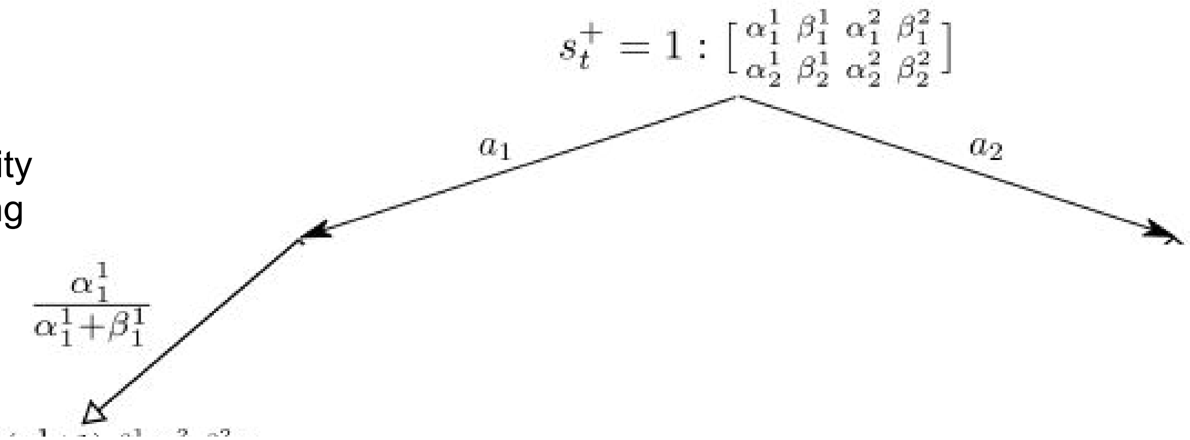


Bayes-Adaptive Bernoulli Bandits



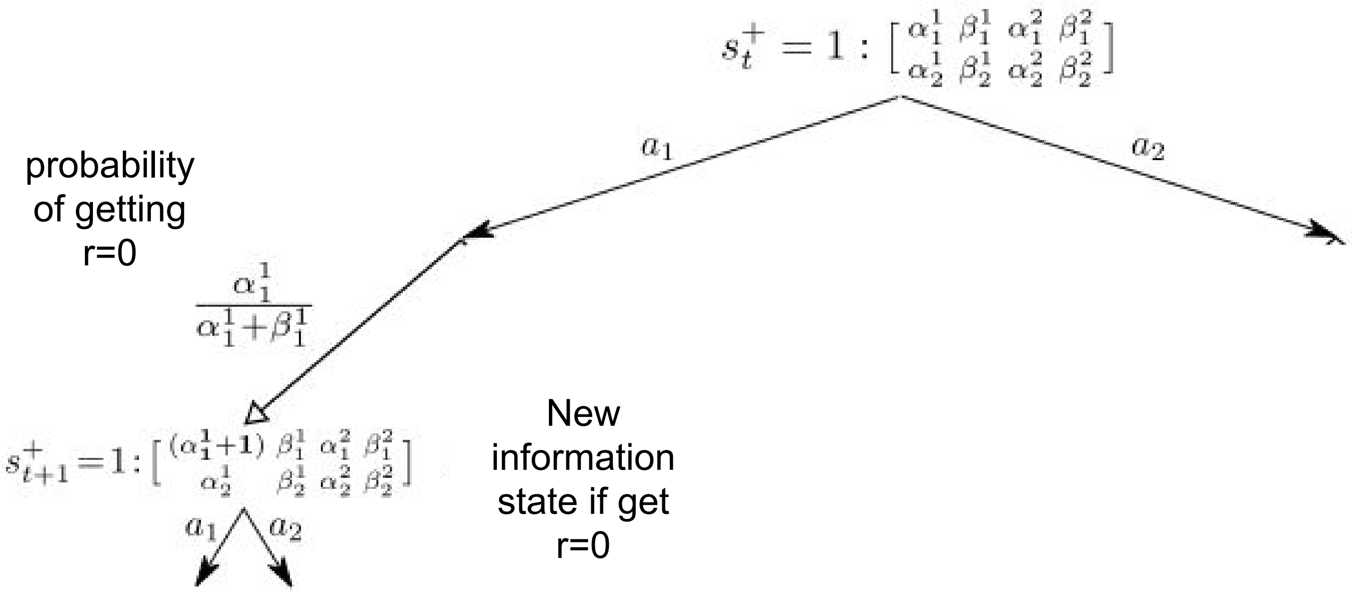
Bayes-Adaptive Bernoulli Bandits

probability
of getting
r=0

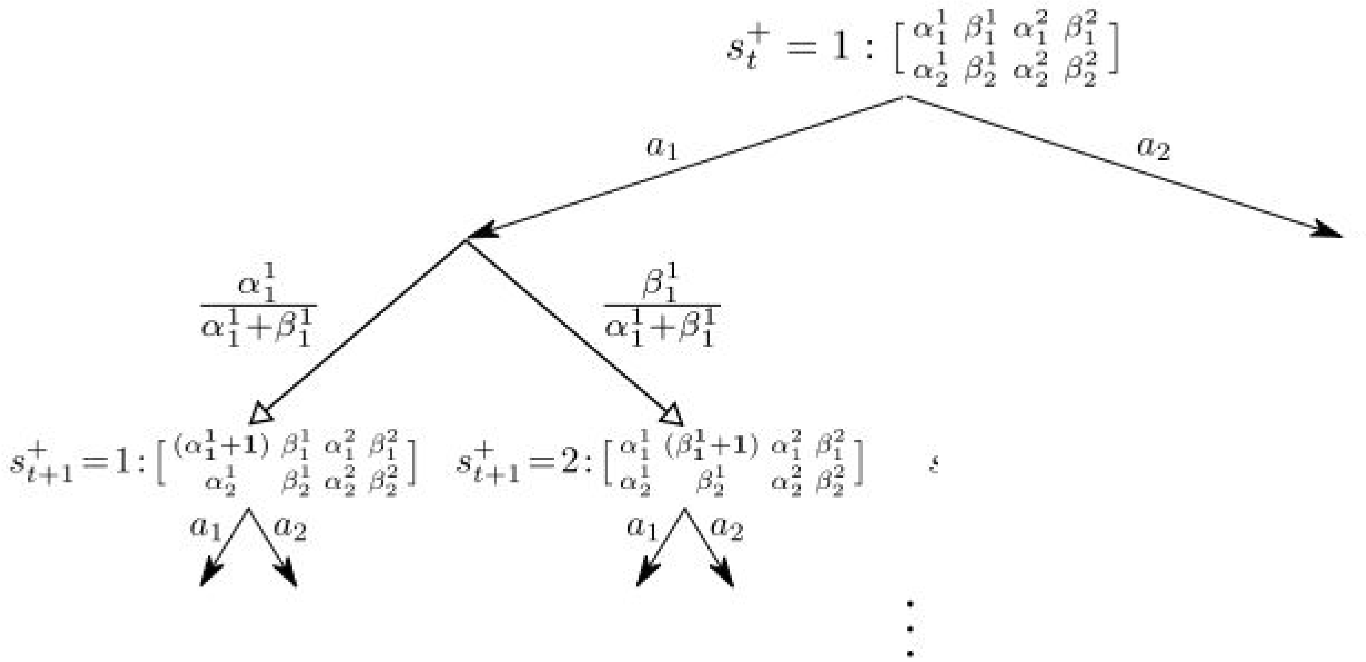


Slide modified from David Silver

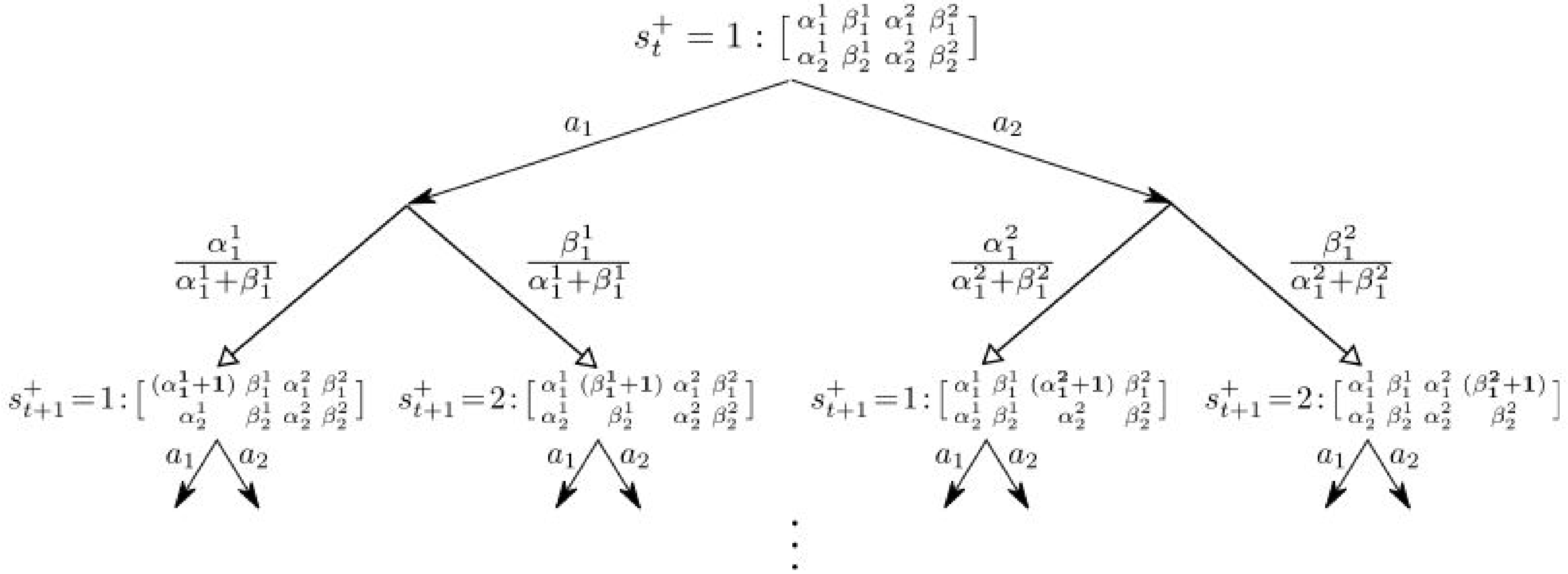
Bayes-Adaptive Bernoulli Bandits



Bayes-Adaptive Bernoulli Bandits



Bayes-Adaptive Bernoulli Bandits



Gittins Index

- Bayes-adaptive MDP can be solved by dynamic programming
- The solution is known as the *Gittins index*
- Exact solution to Bayes-adaptive MDP is typically intractable
  - Information state space is too large
- Recent idea: apply simulation-based search (Guez et al. 2012)
  - Forward search in information state space
  - Using simulations from current information state

Overview

- Quick intro to or refresher of POMDPs
  - Definition
  - Belief state tracking
  - Online planning
- Bayes-optimal bandits
- **Bayes-optimal RL**

Information State MDPs

- MDPs can be augmented to include information state
- Now the augmented state is $\langle s, \tilde{s} \rangle$
  - where s is original state within MDP
  - and $\tilde{s}$ is a statistic of the history (accumulated information)
- Each action a causes a transition
  - to a new state s' with probability $\mathcal{P}^a_{s,s'}$
  - to a new information state $\tilde{s}'$
- Defines MDP $\tilde{\mathcal{M}}$ in augmented information state space

$$\tilde{\mathcal{M}} = \langle \tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{P}}, \mathcal{R}, \gamma \rangle$$

Alternative View: Learning as Planning

- Reinforcement learning as a POMDP planning problem
- Hidden state is the parameters of the MDP: reward model and transition model
- Want to maximize expected discounted sum of rewards given belief state (over these parameters)

Challenge

- Parameter space is real-valued
- → Infinite/continuous state space
- POMDP planning over a continuous set of states…
- Though hidden state is static
  - Assume MDP parameters don't change
- Learning as planning is elegant but often intractable

Finite Set of Models? Finite State POMDP

- Consider if in 1 of M MDPs
- Don't know which one, but only finite number
- Then can model as a finite state POMDP (Poupart et al. 2006, Brunskill 2012)
- Tractable to compute an epsilon-optimal policy!

Infinite Set of Possible MDP Models? Use MCTS Planning

- Model as a POMDP
- Use MCTS planning to solve
- Plus some additional insights-- see Guez et al. 2013

Summary

- Understand how to implement MCTS (very very useful tool for planning and learning in practice)
- Be able to define Bayes-optimal RL (what is the objective being solved)
- Know challenges with solving Bayes-optimal RL (why is it computationally expensive?)
- Know of at least one algorithm could use to compute an approximately Bayes-optimal soln