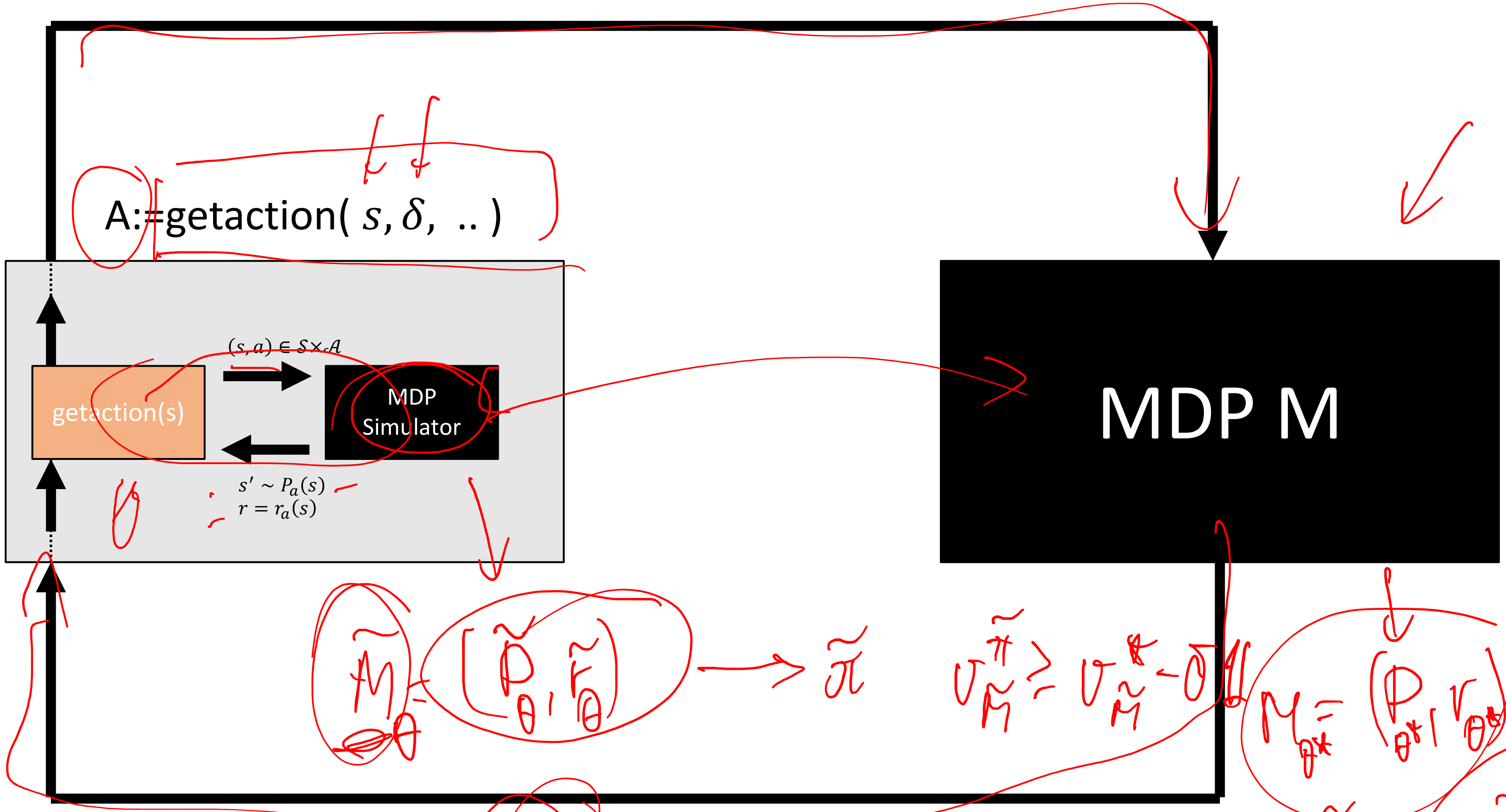
Lecture 5
Local/online planning, part 1

Motivation

- S is too large, cannot afford to run algorithms that scale with S in any ways
- How to address this?
  - Do not require π^*, only $\pi^*(s)$ at the current state
  - Being lazy is good
- No tables, but simulator

Online planning (R97, KMS02)

$A \in \mathcal{A}$



s: current state

Objective: $v^{\pi} \geq v^{*} - \delta \mathbf{1}$

Simulator access: global, local, online

def getaction(simulator, s, δ): online planning, global access simulator

(S,A) := simulator.problemsize()

F := simulator.getallfeatures() # F= $(\phi(s))_s$

...

(s',r') := simulator.gen(s,a) # $s \in [S]$ arbitrary, $a \in [A]$

...

return a # $a \in [A]$ s.t. for the policy π induced, $v^\pi \geq v^* - \delta \mathbf{1}$

def getaction(simulator, s, f, δ): online planning, local access simulator

A := simulator.num_actions() # $f = \phi(s)$

...

(s',r',f') := simulator.gen(s,a) # s: state previously seen, $a \in [A]$, $f' = \phi(s')$

...

return a # $a \in [A]$ s.t. for the policy π induced, $v^\pi \geq v^* - \delta \mathbf{1}$

online access

Value iteration

$$\pi_k(s) = \operatorname{argmax}_a q_{k+1}(s,a)$$

$$q_k = \tilde{T}^k \mathbf{0}$$

$$\tilde{T}q = r + \gamma PMq$$

$$k \geq H_{\gamma,\delta(1-\gamma)/(2\gamma)}$$

```
define q(k,s):
  if k = 0 return 0 # base case
  return [ r(s,a) + gamma * sum( [P(s,a,s') * max(q(k-1,s')) for s' in S] ) for a in A ]
end
```

Cost: $O((SA)^k)$

Bellman optimality operator

$r + \gamma PM(r + \gamma PM q)(r + \gamma PM \ldots)$

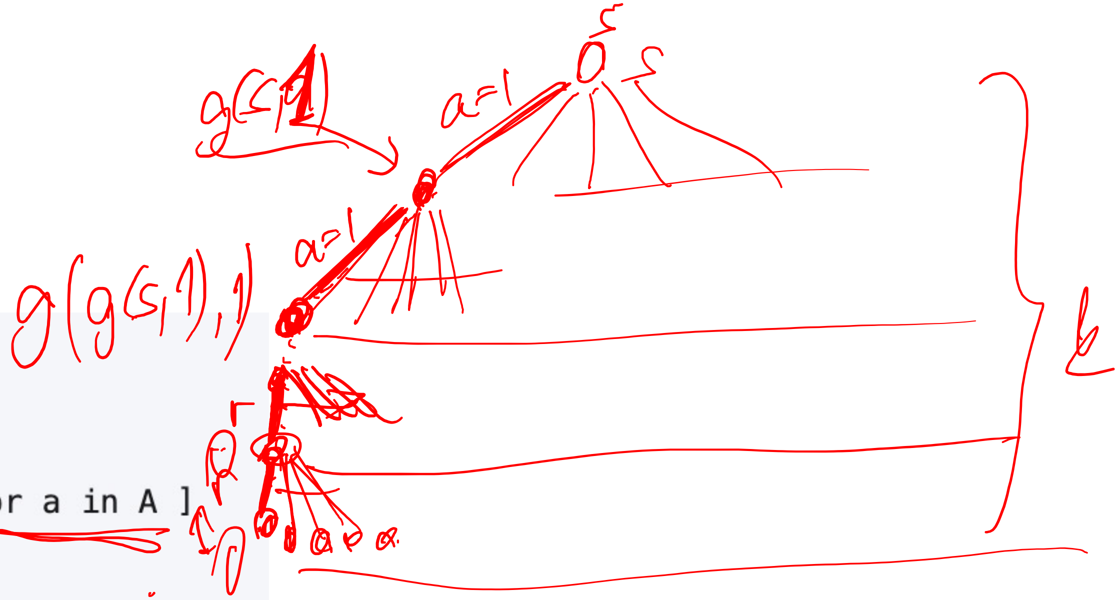
q_0, $q_1 = \tilde{T}q_0$, …, $q_k = \tilde{T}q_{k-1}$

add memoization } saves exp. cost!

Value iteration: Deterministic systems

Next state: $g(s, a)$

```
define q(k,s):
  if k = 0 return 0 # base case
   return [ r(s,a) + gamma * max(q(k-1,g(s,a))) for a in A ]
end
```

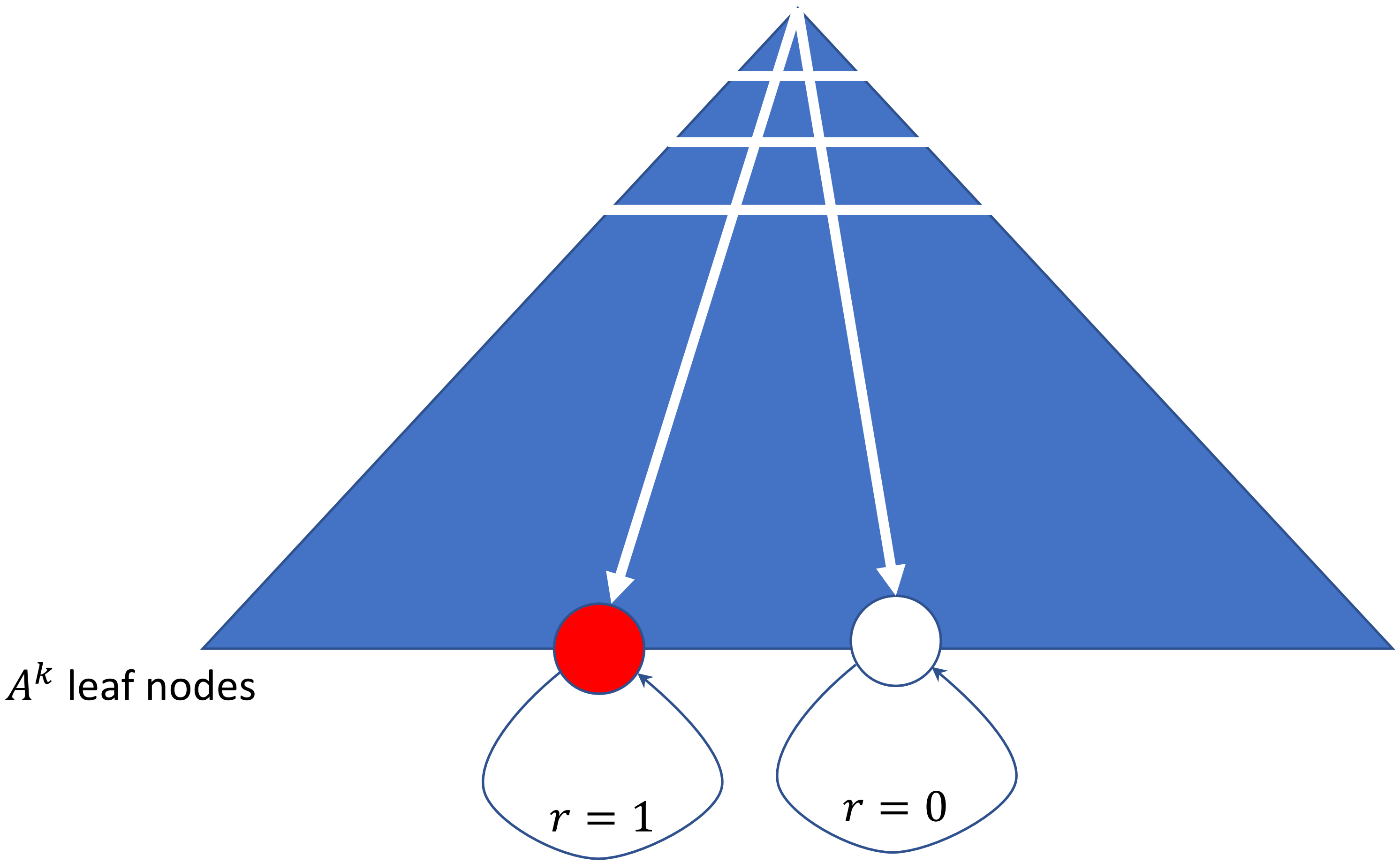


Cost: $O(A^k)$ – independent of S

Theorem (local planning lower bound): Take any local planner p that is δ-sound with $\delta < 1$ for discounted MDPs with rewards in $[0, 1]$. Then there exist some MDPs on which p uses at least $\Omega(\mathrm{A}^k)$ queries at some state with

$$k = \left\lceil \frac{\ln(1/(\delta(1-\gamma)))}{\ln(1/\gamma)} \right\rceil,$$

where A is the number of actions in the MDP.


A^k leaf nodes
$r = 1$
$r = 0$

Questions from slack

Farzane Aminmansour 1 hour ago

The definition of the MDP simulator implies that there is a default assumption that the simulator is a forward model of the MDP. It is mentioned that given a transition (s,a,r, s'), like a successor model, we queried the simulator with input (s,a) and it will output (r, s'). I am curious about if we had a backwards model for planning instead of a forward one wherewith input (s', a), the simulator would have outputted (s, r)?

In particular, how would $P_a(s)$ change in backwards models? It seems that in a backwards simulator, this distribution would be inherently tied to the policy. Imagine a situation where both s_1 and s_2 lead to s' when taking action a. If a policy visits state s_1 more frequently than s_2, then the backwards model will make $p(s_1 | s', a)$ higher than $p(s_2 | s', a)$. How would this affect all the theoretical guarantees in local planning?

+2

Discussion

Computational complexity

- How do we account for compute cost?
- What is computation?
  - Turing model/bit model
  - RAM model/computation over the reals
  - Random bits?
  - Biological computation? Liquid computers? ??
  - Other models? What do we expect of a model of computation?
  - Implications of choices
    - Input size depends on model
    - Cost depends on model
    - Which model is a better fit to “reality”?

https://eccc.weizmann.ac.il//static/books/A_Simple_Introduction_to_Computable_Analysis_Fragments_of_a_Book/