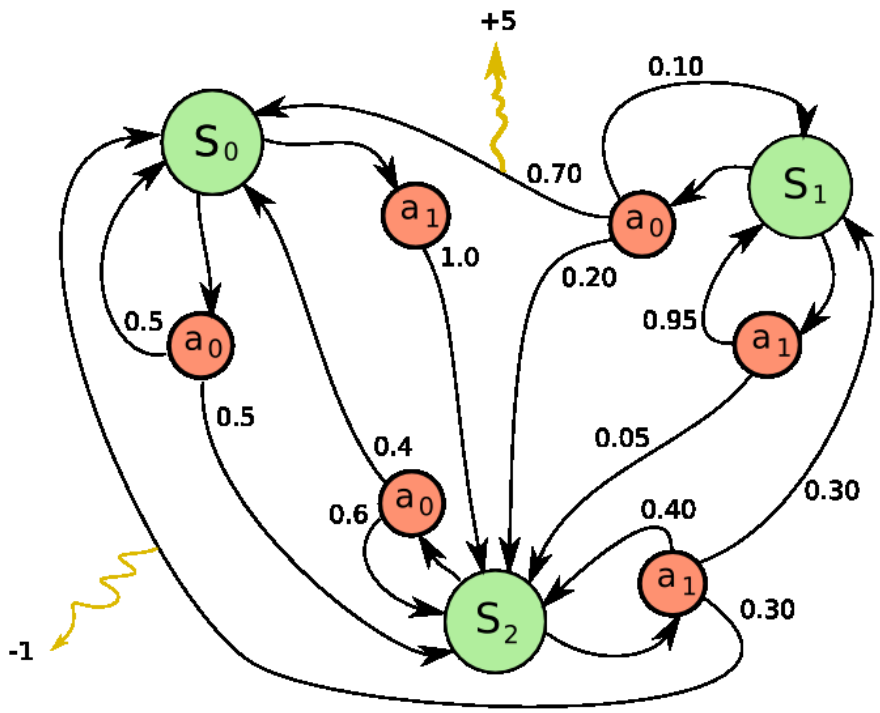
Sample efficient rich observation RL

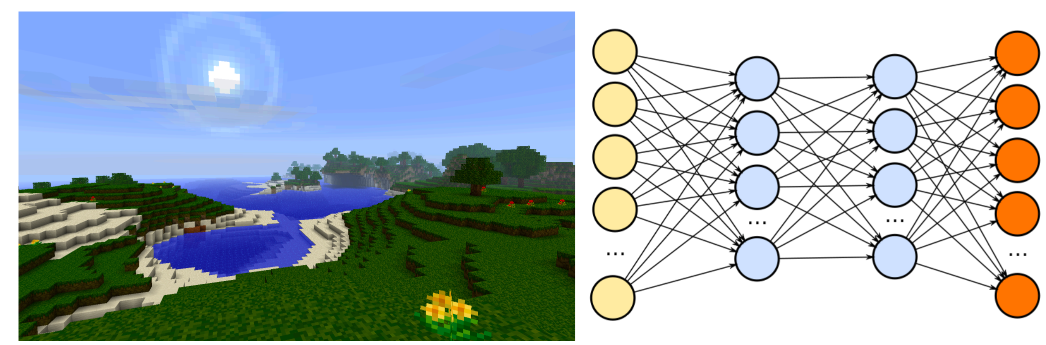
Akshay Krishnamurthy
akshaykr@microsoft.com

RL theory vs practice



Theory

Simple tabular environments
Sophisticated, efficient exploration
No generalization

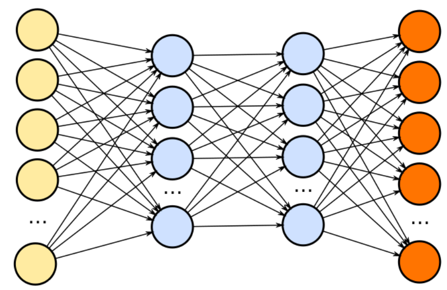
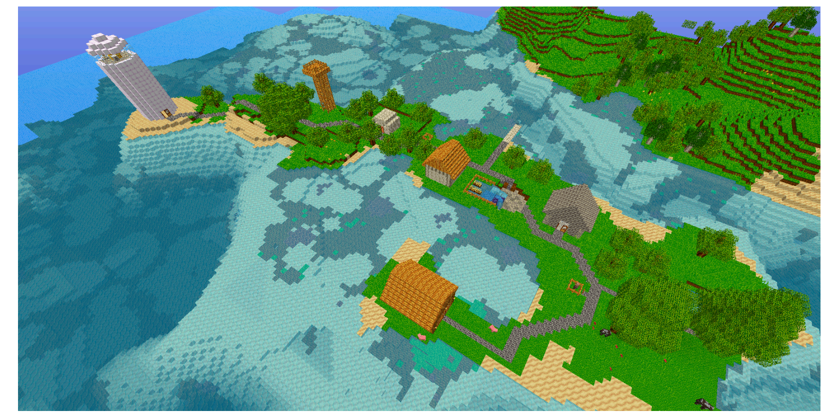


Practice

Complex rich-observation environments
Generalization via function approximation
(relatively) simple exploration

Can we design provably sample-efficient RL algorithms for rich observation environments?

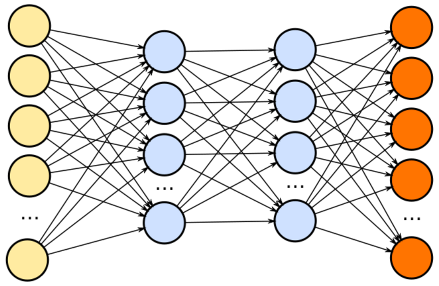
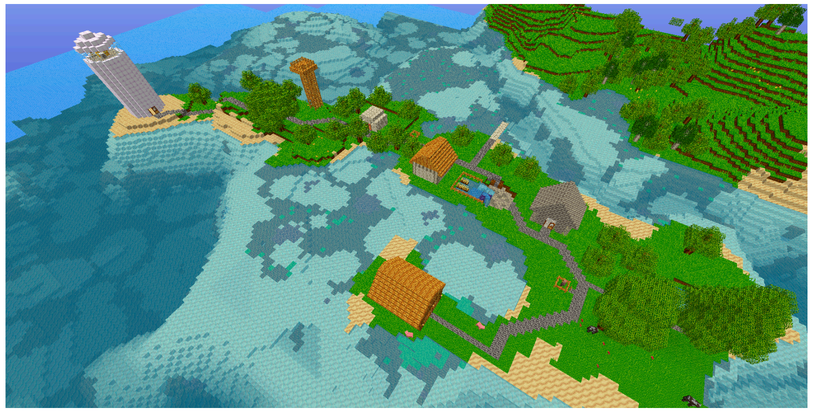
Our goal

Provably efficient algorithms for rich observation reinforcement learning

1. Generalization via function approximation
2. Statistical efficiency
3. Computational efficiency

This talk

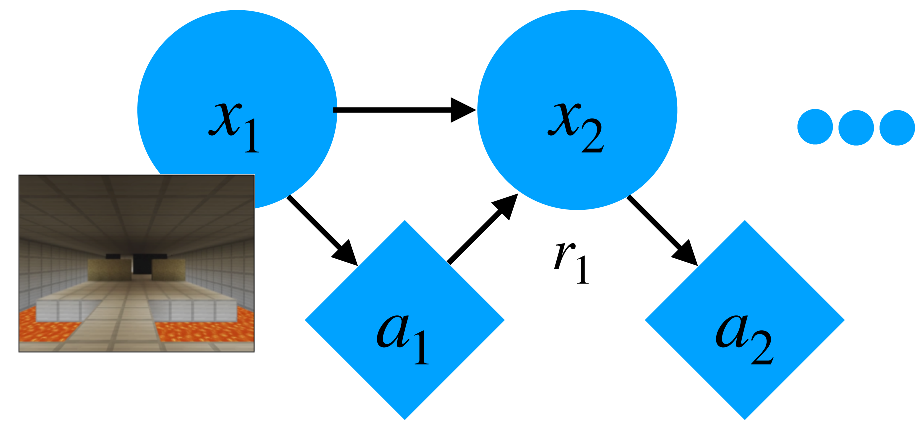
Part 1: Contextual decision processes and OLIVE

- Provably sample efficient
- Not computationally efficient
- A new complexity measure

Part 2: Block-MDPs and PCID

- Provably sample and computationally efficient
- Less general

Formal model



For h = 1, …, H:

- Observe context $x_h \in \mathcal{X}$
- Take action $a_h \in [K]$
- Receive reward $r_h \in \mathbb{R}$
- Transition to x_{h+1}

A policy $\pi : \mathcal{X} \to [K]$ has value $V(\pi) = \mathbb{E}\left[\sum_{h=1}^{H} r_h \mid a_h = \pi(x_h)\right]$

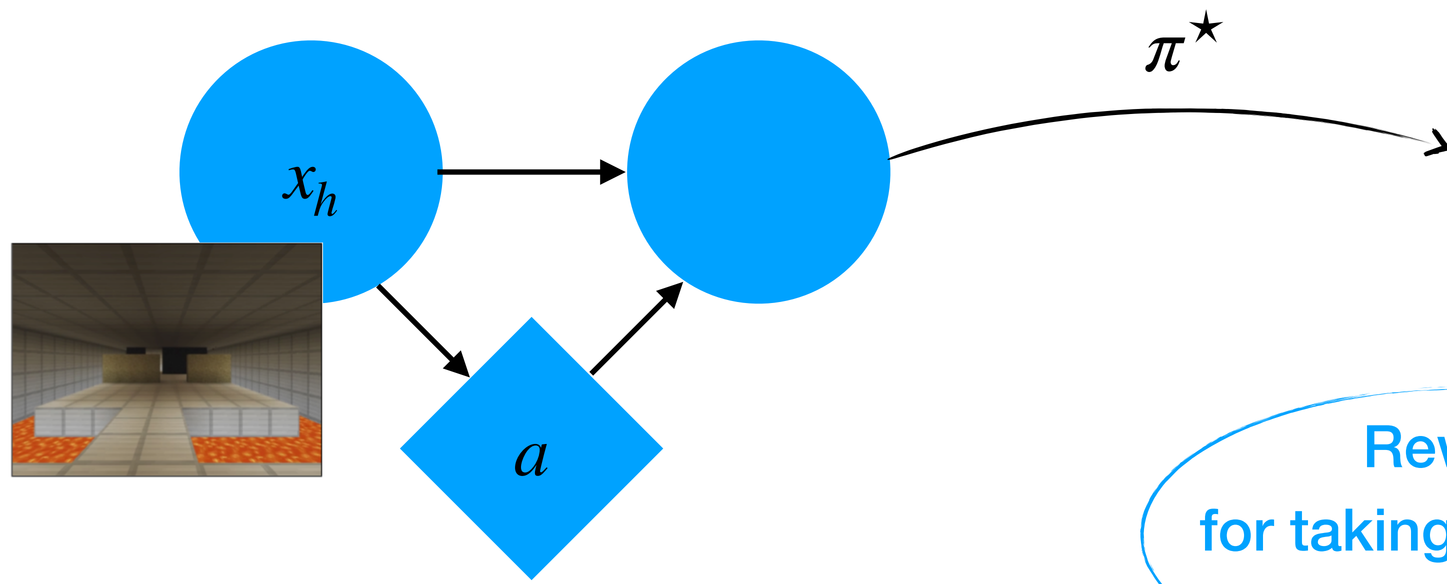
PAC Learning: Find policy $\hat{\pi}$ such that:

Optimal policy

$$V(\hat{\pi}) \geq V(\pi^\star) - \epsilon$$

Sample complexity: Number of trajectories required.

Function Approximation



Reward
for taking a then $\pi^\star$
from x_h

$$Q^\star(x_h, a) = \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'} \mid a_h = a, a_{h'} = \pi^\star(x_{h'})\right]$$

Use class $\mathscr{F}$ to approximate $Q^\star$

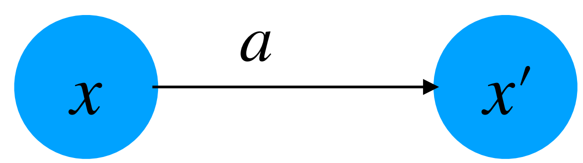
Realizability: assume $Q^\star \in \mathscr{F}$

f induces policy $\pi_f(x) = \underset{a}{argmax}\ f(x, a)$

Bellman Equations and Validity

$$Q^\star(x_h, a) = \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'} \mid a_h = a, a_{h'} = \pi^\star(x_{h'})\right]$$

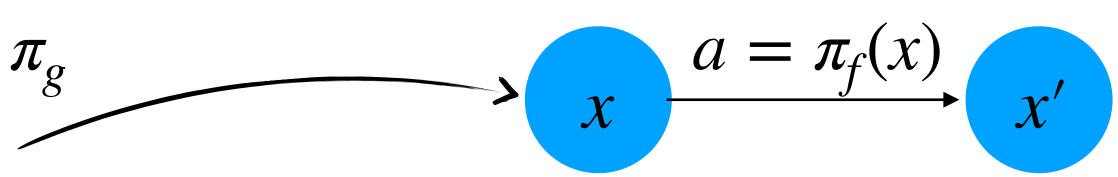
Optimality equation: $\forall x, a$



Hard to use algorithmically with large $\mathcal{X}$

$$Q^\star(x, a) = \mathbb{E}\left[r + Q^\star(x', \pi^\star(x')) \mid x, a\right]$$

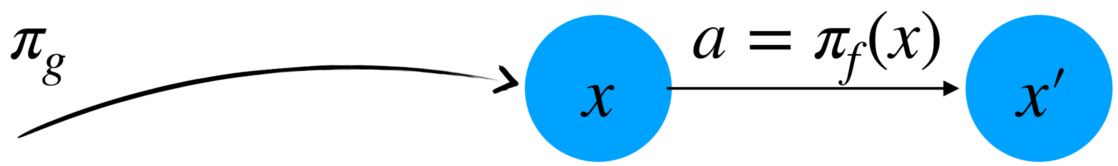
Weaker condition: f is valid if $\forall g \in \mathcal{F}, \forall h \in [H]$



$\mathcal{E}^h(f, g)$

$$\boxed{\mathbb{E}\left[f(x_h, a_h) - r - f(x_{h+1}, \pi_f(x_{h+1})) \mid a_{1:h-1} \sim \pi_g, a_h \sim \pi_f\right]} = 0$$

On Validity



$\mathcal{E}^h(f, g)$

$$\boxed{\mathbb{E}\left[f(x_h, a_h) - r - f(x_{h+1}, \pi_f(x_{h+1})) \mid a_{1:h-1} \sim \pi_g, a_h \sim \pi_f\right]} = 0$$

$Q^\star$ is always valid!

If f is valid, easy to estimate $\pi_f's$ value:

$$f \text{ valid } \Rightarrow V(\pi_f) = \mathbb{E}\left[f(x_1, \pi_f(x_1))\right]$$

For fixed (g, h), can check for all f with importance weighting

Idea: Find valid fs, then use predictions to optimize for policy
Key issue: How to choose roll-in distribution g?
Answer: Optimism!

OLIVE

Optimistic guess for $V^\star$

Repeat:

1. Pick $\hat{f} \in \mathscr{F}$ to maximize $\mathbb{E}\left[f(x_1, \pi_f(x_1))\right] = V^f(\pi_f)$

2. Test if $\hat{\pi} = \pi_{\hat{f}}$ is good: $V(\hat{\pi}) \geq V^{\hat{f}}(\hat{\pi})$?

   Check our optimistic guess

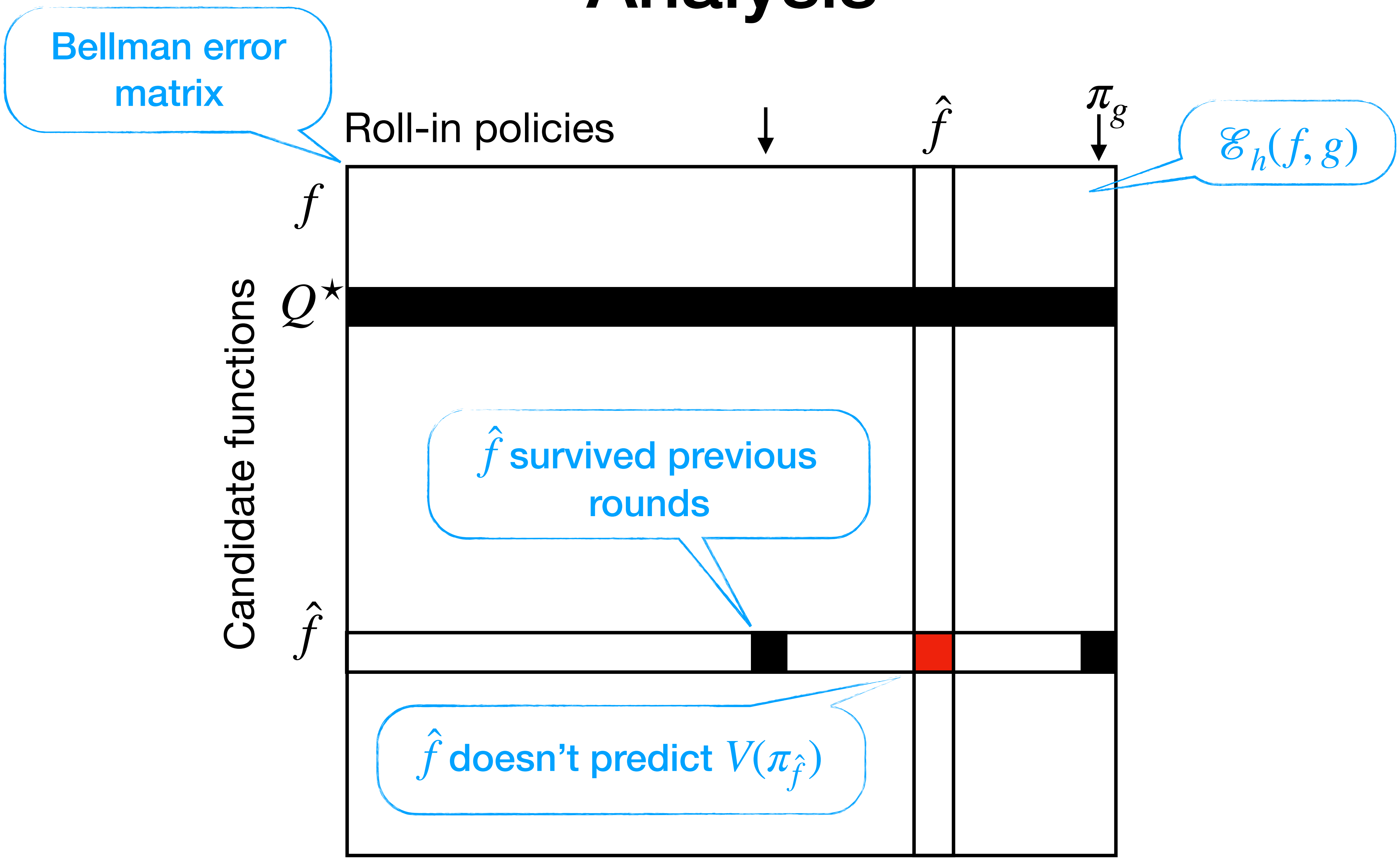
3. If it is, terminate and output $\hat{\pi}$

4. Otherwise, eliminate all f for which $\mathscr{E}^h(f, \hat{f}) \neq 0$ at some h

   Refine the search space

First observation: Each iteration requires few samples
Key issue: How many iterations?

Analysis



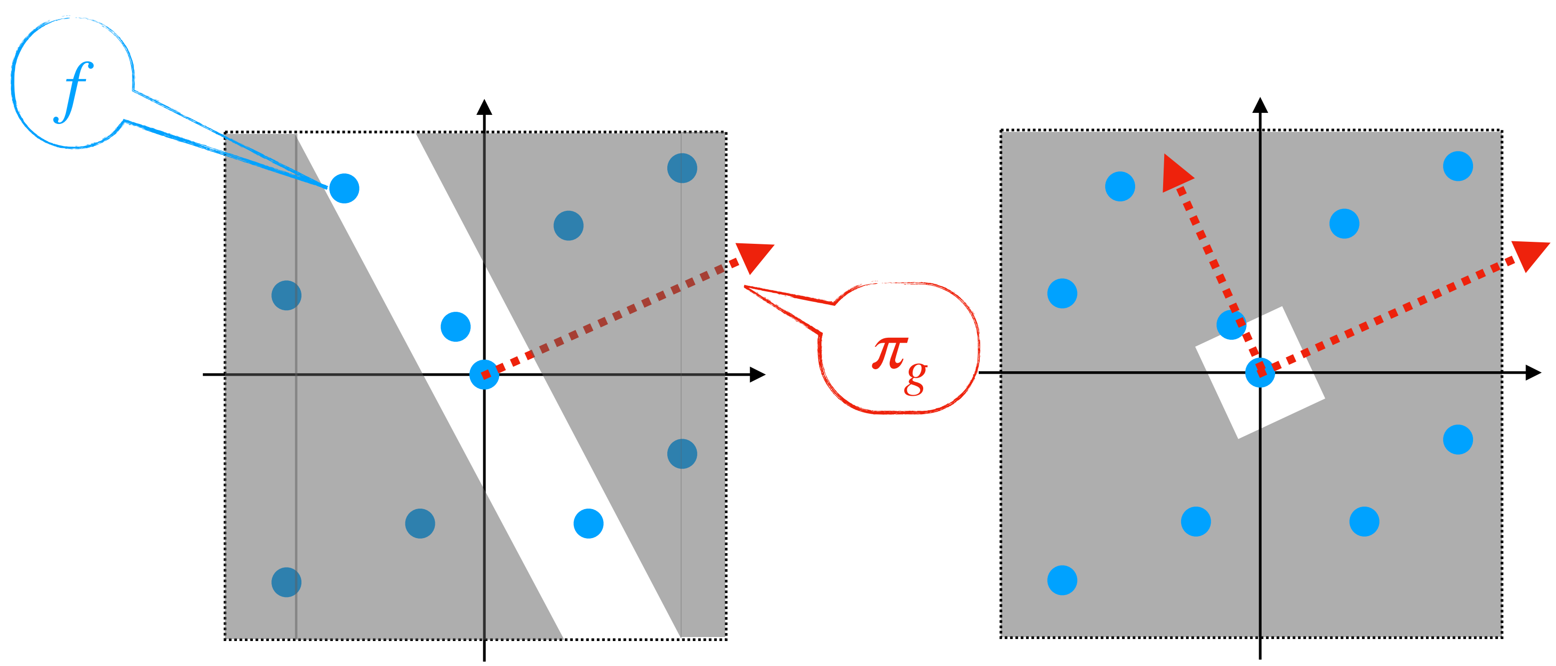
$\hat{f}$ column is linearly independent of previous $\Rightarrow$ # of iterations $\leq$ matrix rank

Main result

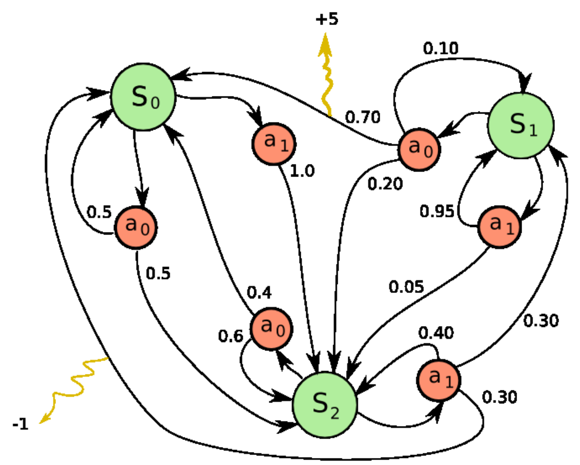
Theorem: Sample complexity of OLIVE is $poly(\textbf{rank}, K, H, \log|\mathcal{F}|)$

K = Number of actions, H = Time horizon

- Handling statistical errors more complicated
  - Uses geometric argument with ellipsoid volumes
- We call the complexity measure the **Bellman Rank**

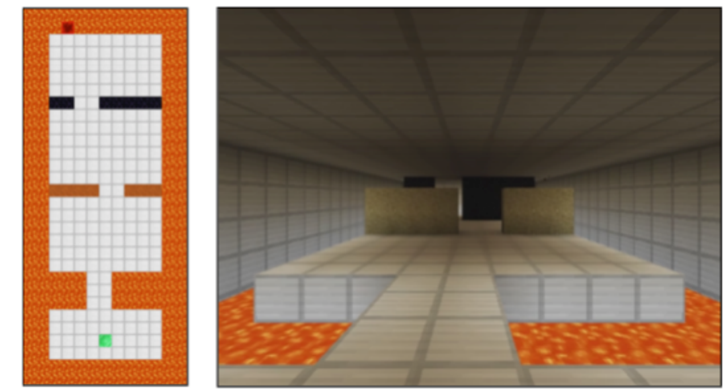


When is Bellman rank small?

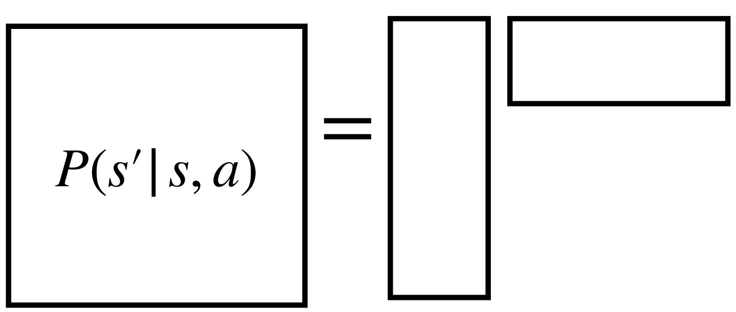


Finite/Tabular MDPs

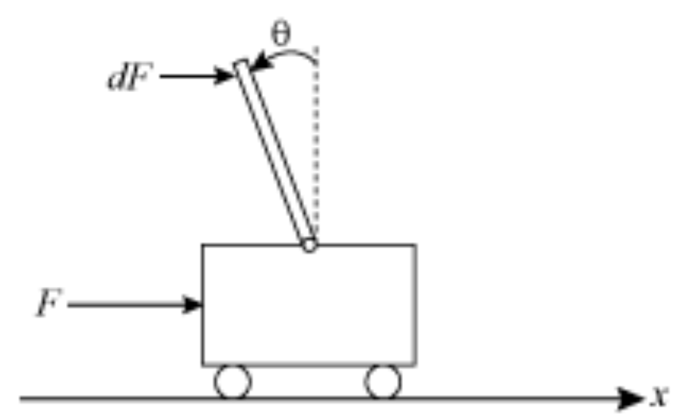
More in part 2!



Block MDPs
(rich-obs MDPs)



Low rank dynamics
(Linear MDP)



LQR control

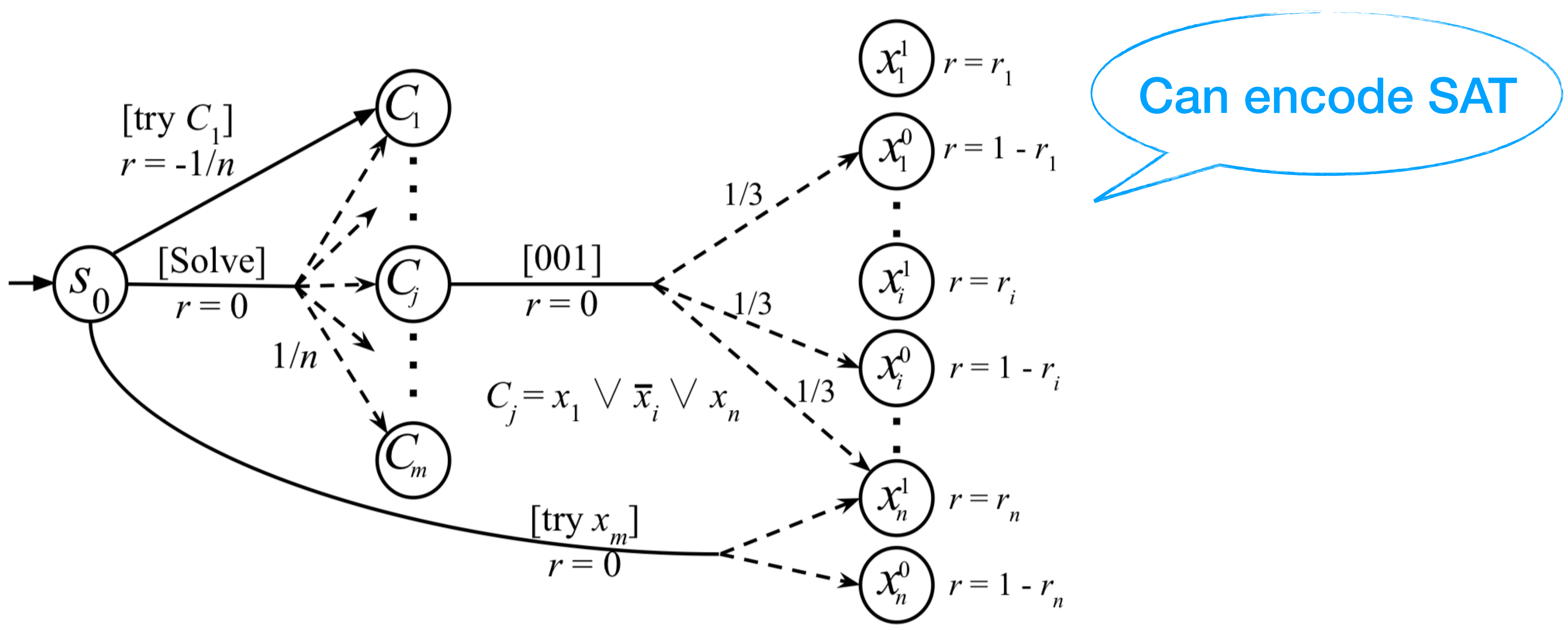
Also

$Q^\star$ preserving abstraction

Reactive PSRs

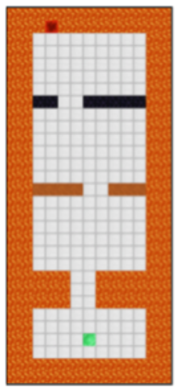
Summary for OLIVE

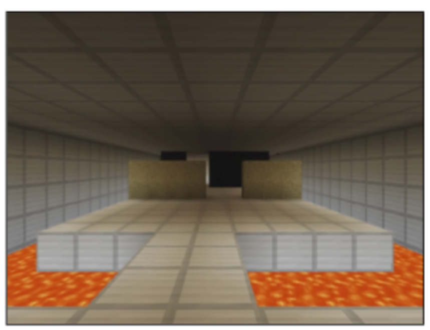
- New complexity measure: Bellman rank
- New algorithm: OLIVE with PAC analysis
  - Extensions: robustness, infinite function classes, etc.
- Recent progress:
  - Model-based version [SJKAL Colt 19]
  - $\sqrt{T}$ regret [DPWZ arXiv 19]
- **But** not computationally efficient!
  - NP-hard even in tabular case [DJKALS NeurIPS 18]

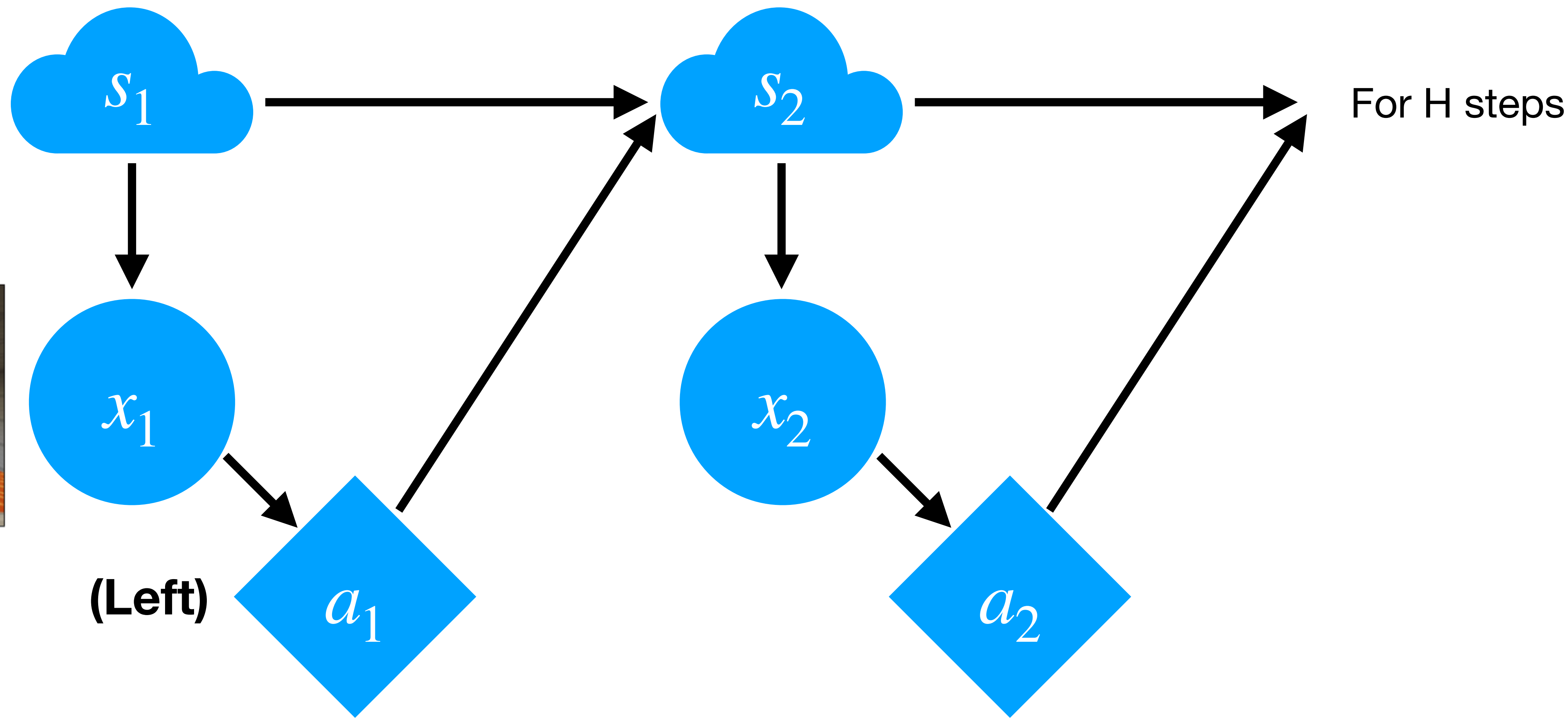


State decoding in Block MDPs

Block MDPs

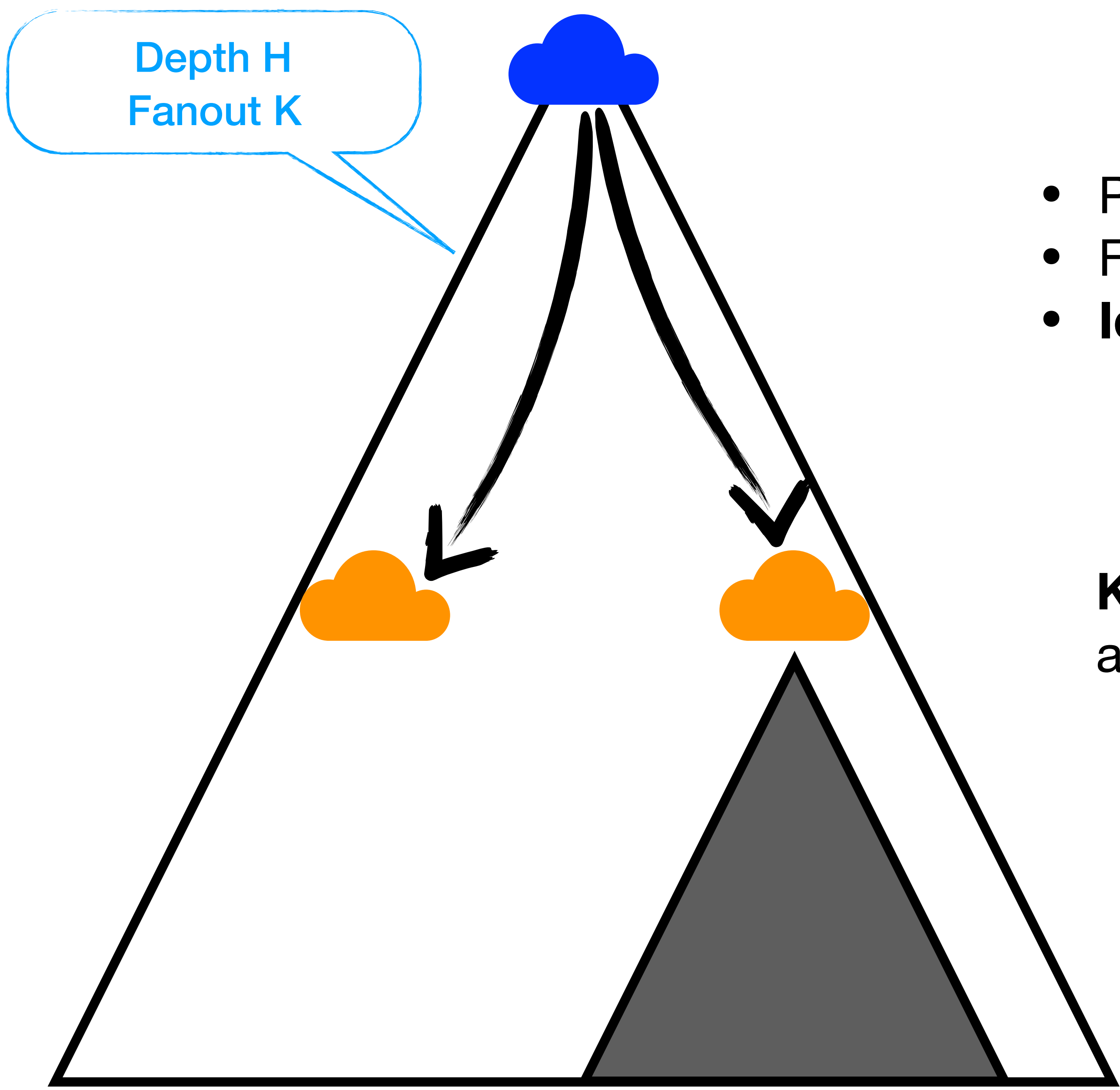






Agent only observes rich context (visual signal)
Environment summarized by small hidden state space (agent location)
State can be decoded from observation
Contexts and transitions are stochastic
Goal: Reward free exploration, find policies that cover the state space

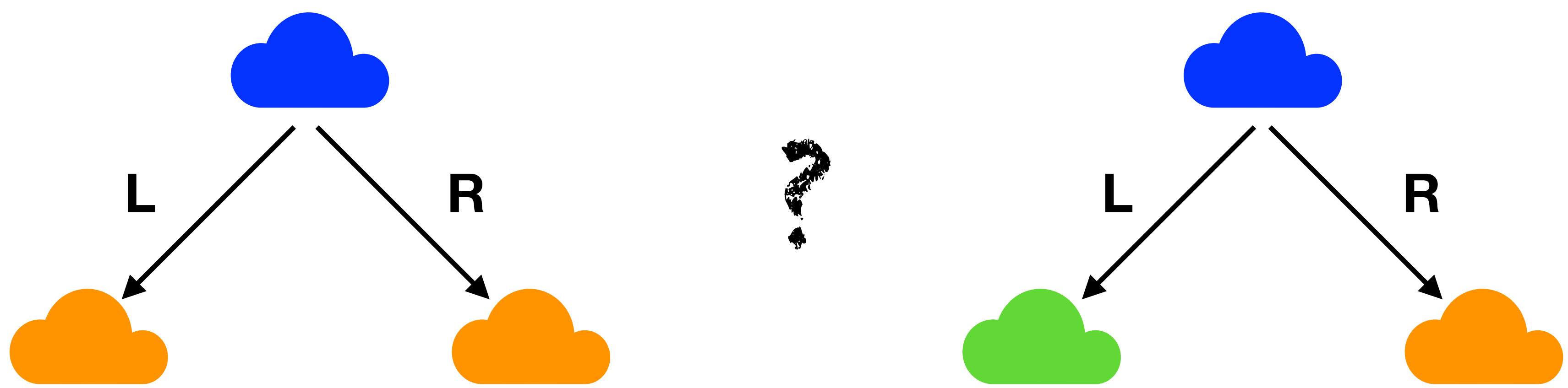
Deterministic case: Intuition



- Process is a large search tree
- Few hidden states => highly redundant!
- **Idea:** prune redundant paths

Key question: How to tell if two paths arrive at same state?

Deterministic case: Intuition



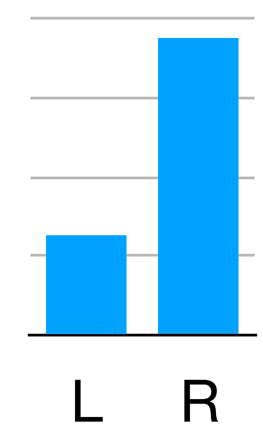
Idea: Try to predict previous action from current context

For each action a:

    From start, try a, observe next context x

    Add example x with label a to dataset

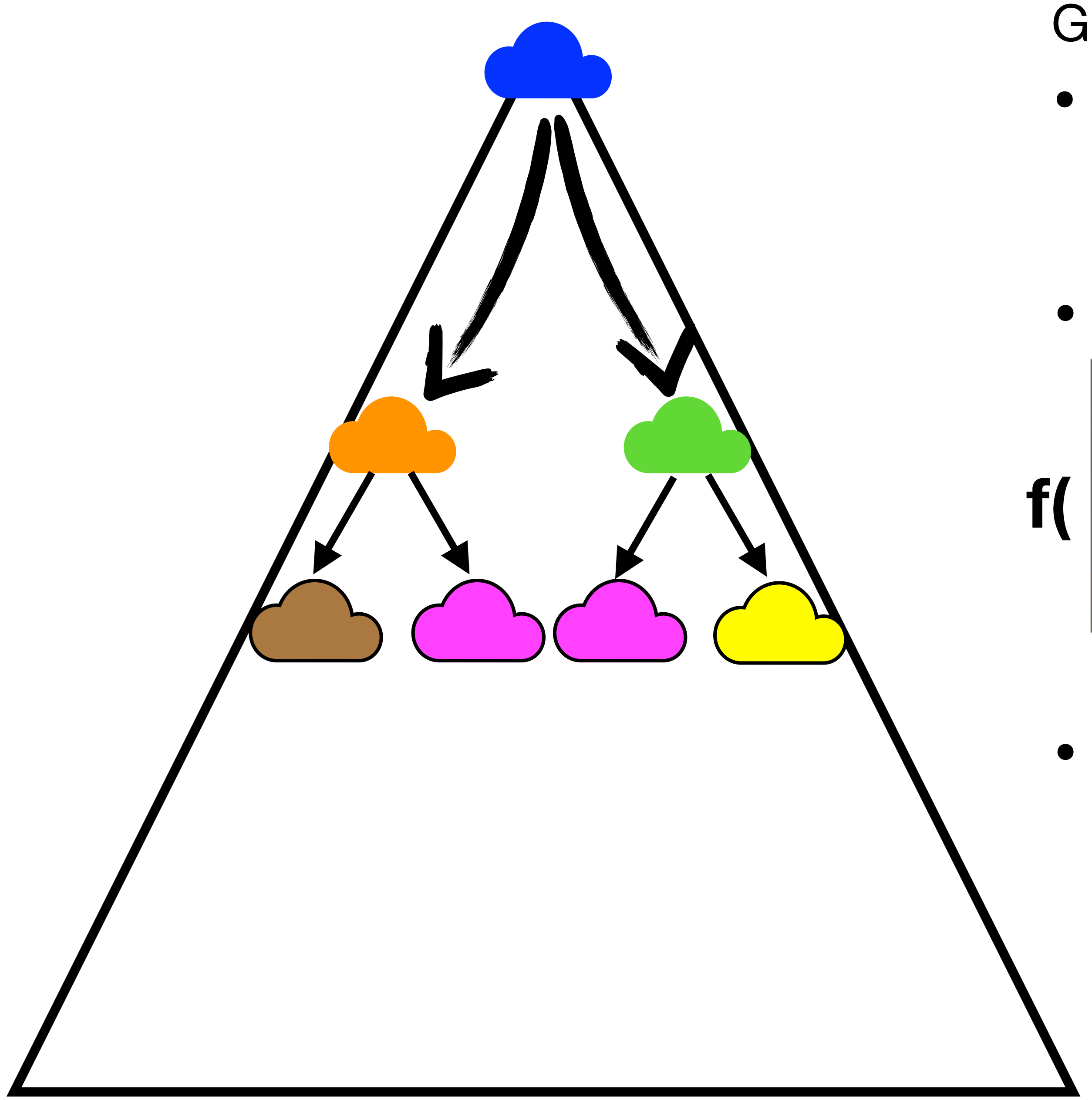
Train classifier on collected dataset

f( **) =** 

If classifier has trivial performance, can prune paths!

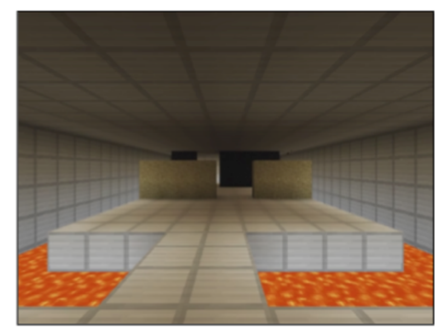
Classifier can also decode hidden state

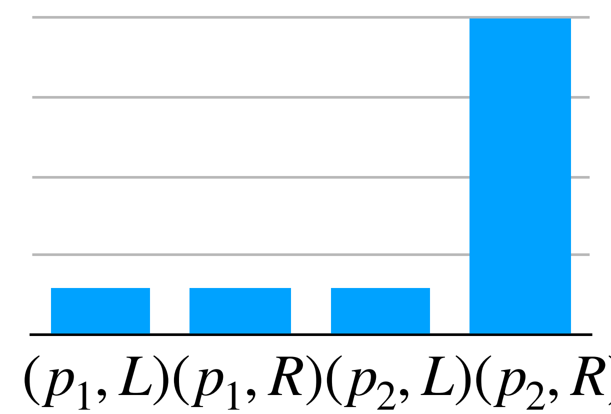
Deterministic case: Algorithm



Given paths for level $h - 1$

- For each path p:
  - follow p, take a uniformly and observe x.
  - Add example x with label (p, a) to dataset
- Train classifier

f( **)** **=**



- Prune redundant paths

labels

predictions

0.97			
	0.48	0.48	
	0.48	0.48	
			0.97

Stochastic Case: Algorithm

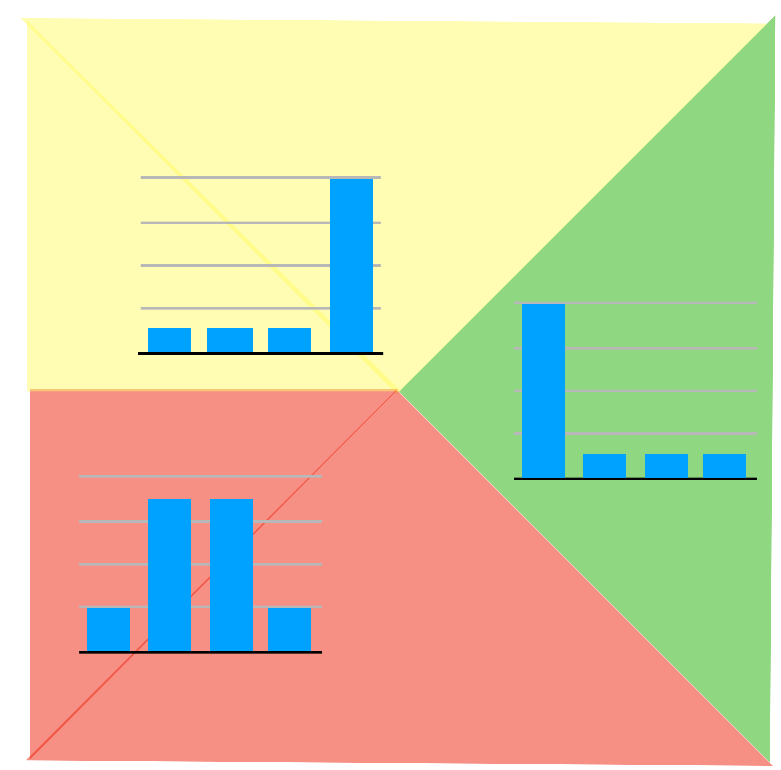
Given ~~paths~~ policies and decoder for level $h-1$

- For each ~~path~~ policy p:
  - follow p, take a uniformly and observe x. use decoder to predict s
  - Add example x with label $\cancel{(p,a)}$ (s,a) to dataset
- Train classifier
- ~~Prune redundant paths~~
- Cluster classifier outputs to get next decoder
- Fit transition model on hidden states
- Compute policies to visit all hidden states

Changes

- Replace paths with policies
- Classifier from previous level decodes hidden state to give supervision
- Next decoder obtained by clustering
- Then use model-based planning

f(data) =



Guarantees

Statistical efficiency

Theorem: PCID can find policy cover with poly(M,K,H) samples in polynomial time, with H calls to supervised learning black box.

M = Number of hidden states, K = Number of actions, H = Time horizon

Computational efficiency

Rich observations

Assumptions

- Supervised learner expressive enough: essentially can decode s from x
- Latent states reachable and identifiable

Identifiability

Define backward probability

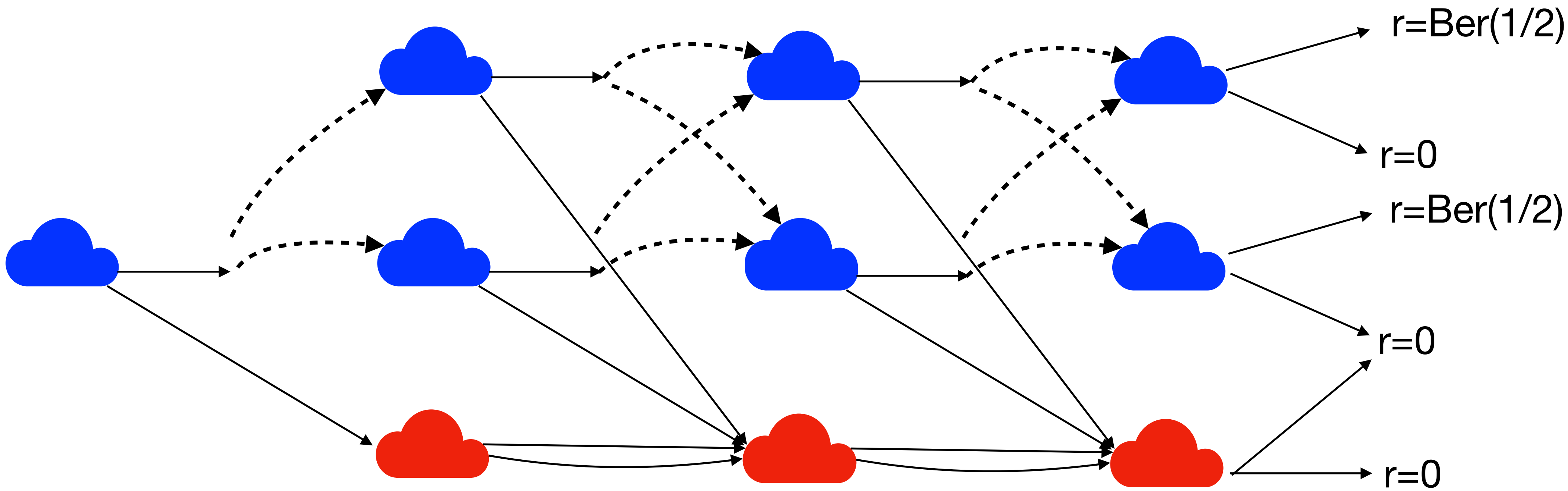
$$b_\nu(s, a \mid s') = \frac{p(s' \mid s, a)\nu(s, a)}{\sum_{\tilde{s},\tilde{a}} p(s' \mid \tilde{s}, \tilde{a})\nu(\tilde{s}, \tilde{a})}$$

Probability of coming from (s, a) given we are in s' now, and marginal is ν.

Margin: $$\gamma = \min_{s',s''} \|b_{unif}(s') - b_{unif}(s'')\|_1$$

- Our classifier is exactly learning $b(s')$: margin enables clustering!
- Sample complexity is actually $poly(M, K, H, 1/\mu_{\min}, 1/\gamma)$.
- Margin is always constant for deterministic latent transitions

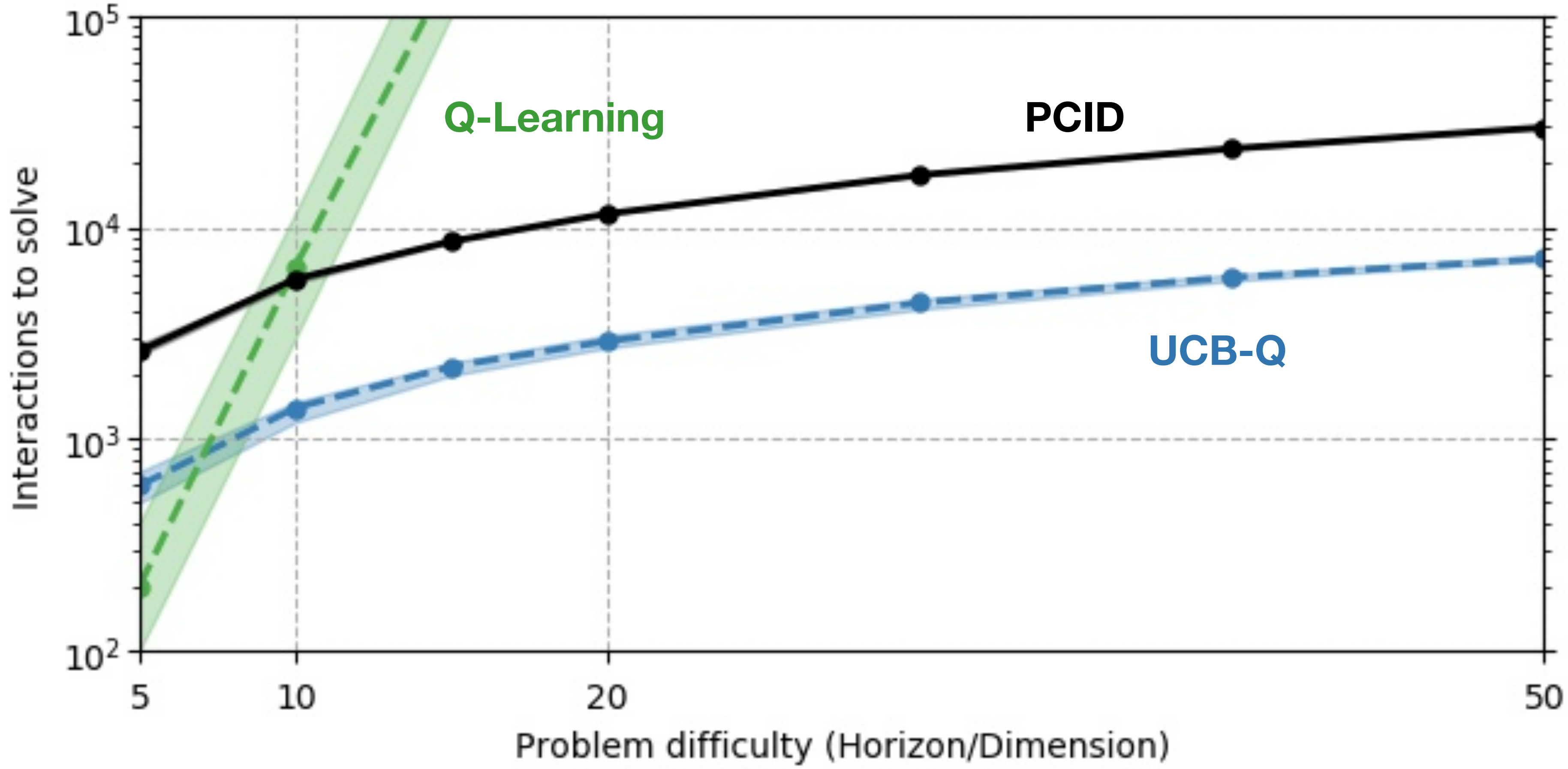
Experiment Setup



Combination lock environment

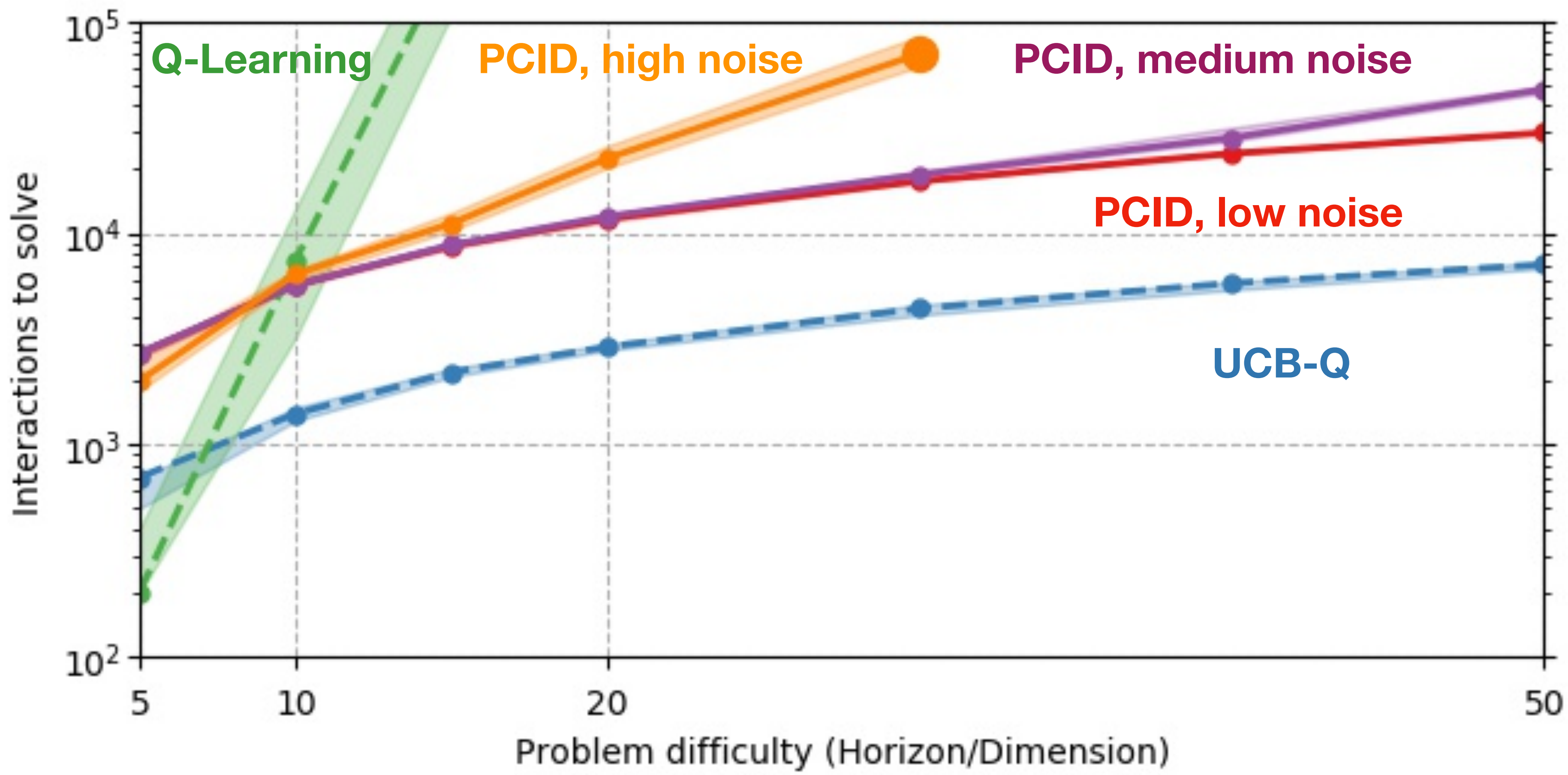
- Two good states per time, one bad state
- Good action different at each good state/time
    - Transitions stochastically to a good state at next time
- Baselines operate on small hidden state space directly
- Our method operates on rich observations (different in each experiment)

Experiment #1



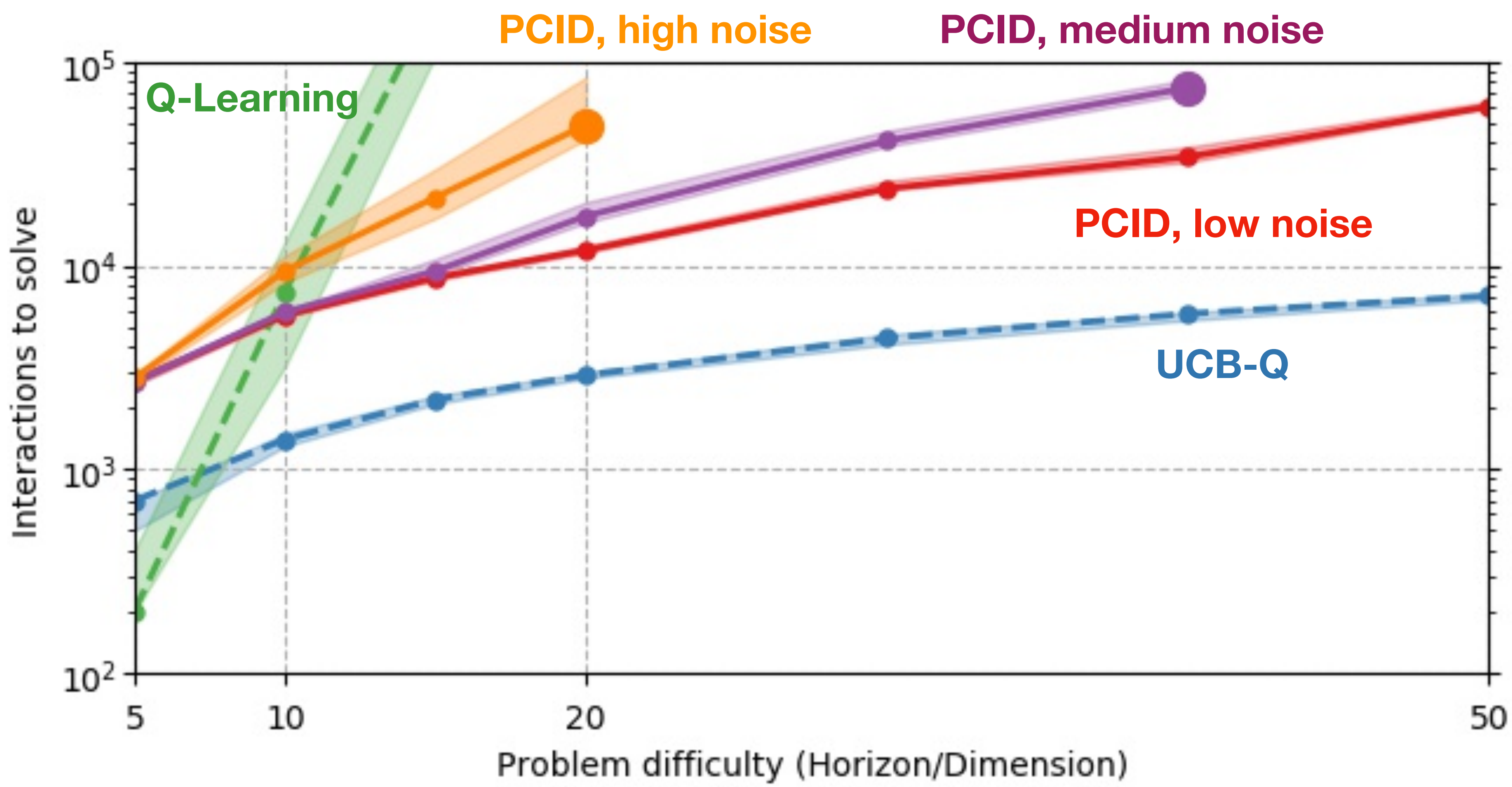
- Deterministic latent transitions
- Rich observations: one-hot state encoding padded with random coin flips
- PCID uses linear model

Experiment #2



- Stochastic hidden transitions
- Rich observations: one-hot state encoding with additive Gaussian noise
- PCID uses linear model

Experiment #3



- Stochastic hidden transitions
- Rich observations: one-hot state encoding with additive Gaussian noise
- PCID uses neural network as supervised learner

Summary and Next Steps

- PCID: New decoding-based algorithm for Block-MDPS
  - Provably sample and computationally efficient
- OLIVE: General purpose algorithm for wide class of environments
  - Sample efficient but not computationally efficient.

A practical, provably sample efficient algorithm that scales to Deep RL benchmarks?

OLIVE: https://arxiv.org/abs/1610.09512
PCID: https://arxiv.org/abs/1901.09018

Thanks!