Sampled Lookahead Approaches in Dynamic Programming and Reinforcement Learning

Daniel R. Jiang

Joint work with co-authors:
Lina Al-Kanj, Warren B. Powell, Ibrahim El-Shar

National University of Singapore
Institute of Operations Research and Analytics
December 9, 2020

Outline

Introduction

Large-scale sequential decision making via reinforcement learning has seen successes in many high-profile settings, including but not limited to logistics, games, and complex control tasks. Just a few examples:

- Simao et al., *An approximate dynamic programming algorithm for large-scale fleet management: A case application*, Transportation Science, 2008
- Mnih et al., *Human-level control through deep reinforcement learning*, Nature, 2015.
- Silver et al., *Mastering the game of Go with deep neural networks and tree search*, Nature, 2016.
- Bellemare et al., *Autonomous navigation of stratospheric balloons using reinforcement learning*, Nature, 2020.

Introduction

Two common threads can be observed in successful applications of reinforcement learning (RL) in practice:

1. There is a *planning* or *approximate dynamic programming* aspect of the solution; it is rarely the case that RL is applied in its "pure" form (where an agent interacts with an environment while learning).
2. The "cost" of collecting experience (i.e., interactions with the environment) is low. In each of the above, a simulator of environment interactions is available.

Introduction

In this talk: we'll unify two recent papers, motivated by (1) and (2), that leverage *information relaxation bounds.*[1]

- **Jiang**, Al-Kanj, Powell, *Optimistic Monte Carlo tree search with sampled information relaxation dual bounds*, Operations Research, 2020.
- El-Shar, **Jiang**, *Lookahead-bounded Q-learning*, ICML, 2020.

[1] Brown, et al. *Information relaxations and duality in stochastic dynamic programs*, Operations Research, 2010

Outline

Finite Horizon MDP Model

- Let $\mathcal{S}$ be the state space, $\mathcal{A}$ be the action space, r be the reward function, f be the transition function, and T be the horizon length.

- Our objective is to find mappings $\pi_t : \mathcal{S} \to \mathcal{A}$ to solve:

$$\max_{\pi \in \Pi} \mathbf{E}\left[\sum_{t=0}^{T-1} r(S_t, \pi_t(S_t)) \,\middle|\, S_0 = s_0\right],$$

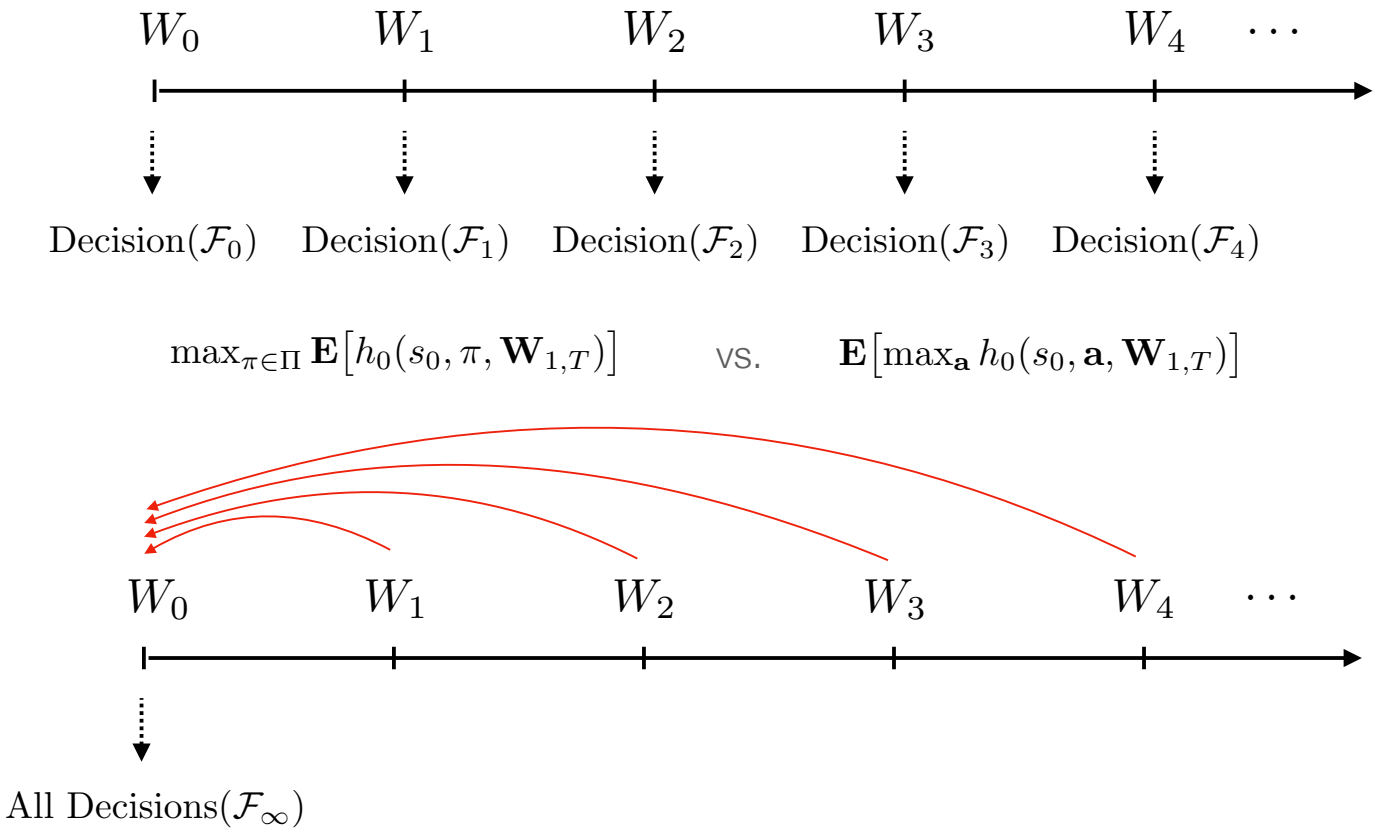
  where $\{W_t\}$ is a noise process, $S_t \in \mathcal{S}$, $a_t = \pi_t(S_t) \in \mathcal{A}$, and state transitions occur via $S_{t+1} = f(S_t, a_t, W_{t+1})$.

- Let $h_t(s, \mathbf{a}, \mathbf{w})$ and $h_t(s, \pi, \mathbf{w})$ be the total reward
  - starting at time t,
  - and state s,
  - while following either a sequence of actions $\mathbf{a}$ or a policy π.

- Equivalently and more concisely, $\max_{\pi \in \Pi} \mathbf{E}\big[h_0(s_0, \pi, \mathbf{W}_{1,T})\big]$.

Information Relaxation Dual Bound

"Knowing the future." (Brown et al., 2010)



If we can quickly approximate this, then we have some sense of how good an action might be.

Information Relaxation Dual Bound

(Brown et al., 2010)

Let $Q_t^*(s,a) = \max_{\pi \in \Pi} \mathbf{E}\left[\sum_{\tau=t}^{T-1} r(S_\tau, \pi_\tau(S_\tau)) \,\middle|\, S_t = s, a_t = a\right]$ and $V_t^*(s) = \max_a Q_t^*(s,a)$.

Proposition (Duality). Fix a stage $t \in \mathcal{T}$ and an initial state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$. Then, it holds that

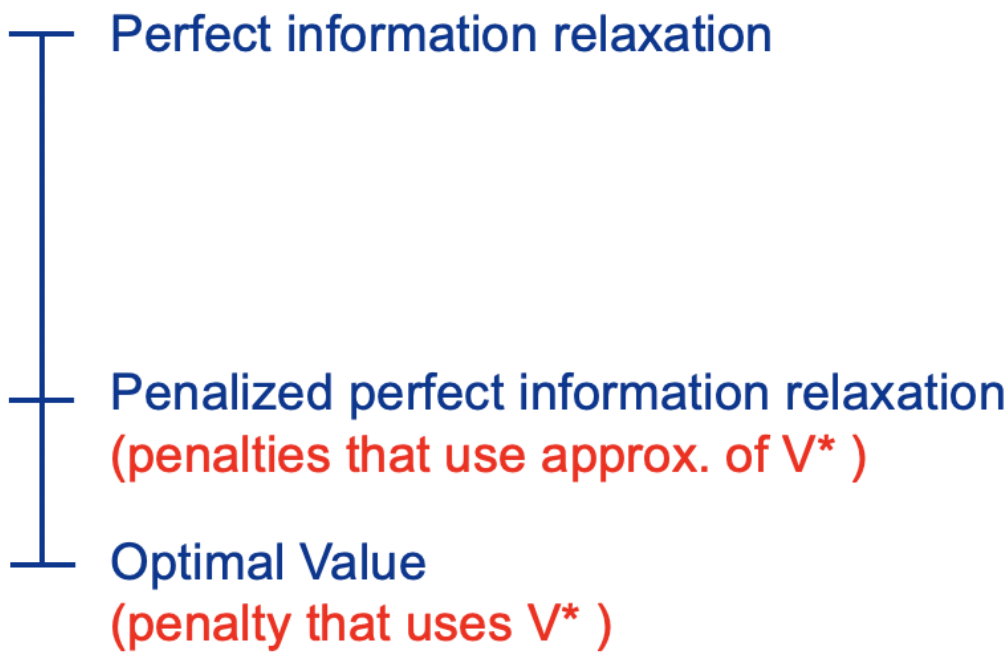
$$Q_t^*(s,a) \leq \mathbf{E}\Big[r(s,a) + \max_{\mathbf{a}} \big[h_{t+1}(S_{t+1}, \mathbf{a}, \mathbf{W}_{t+1,T}) - \text{penalty}(v, \mathbf{a}, \mathbf{W}_{t+1,T}) \big] \Big],$$

where $S_{t+1} = f(s,a,W_{t+1})$ and the optimization is over the vector of actions $\mathbf{a} = (a_{t+1}, \ldots, a_{T-1})$.

If $v = V^*$, then both sides are **equal.** *"If the value of future information is known, then a penalty can be constructed to recover the optimal value of the primal problem."*

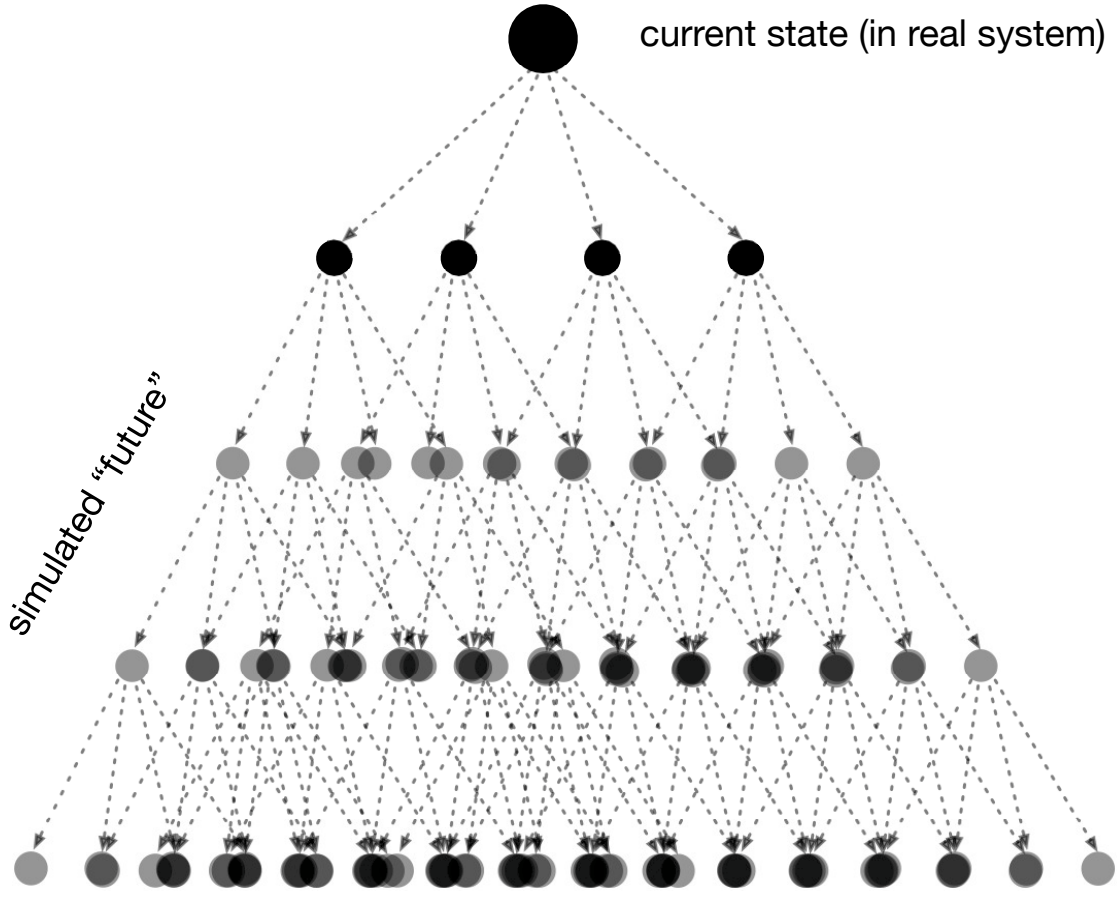
- **Idea 1.** These bounds can be *sampled*.
- **Idea 2.** A penalty function can be defined using an **approximation of** V^* (see Prop 2.3 of Brown et al. 2010 for justification).

Information Relaxation Dual Bound



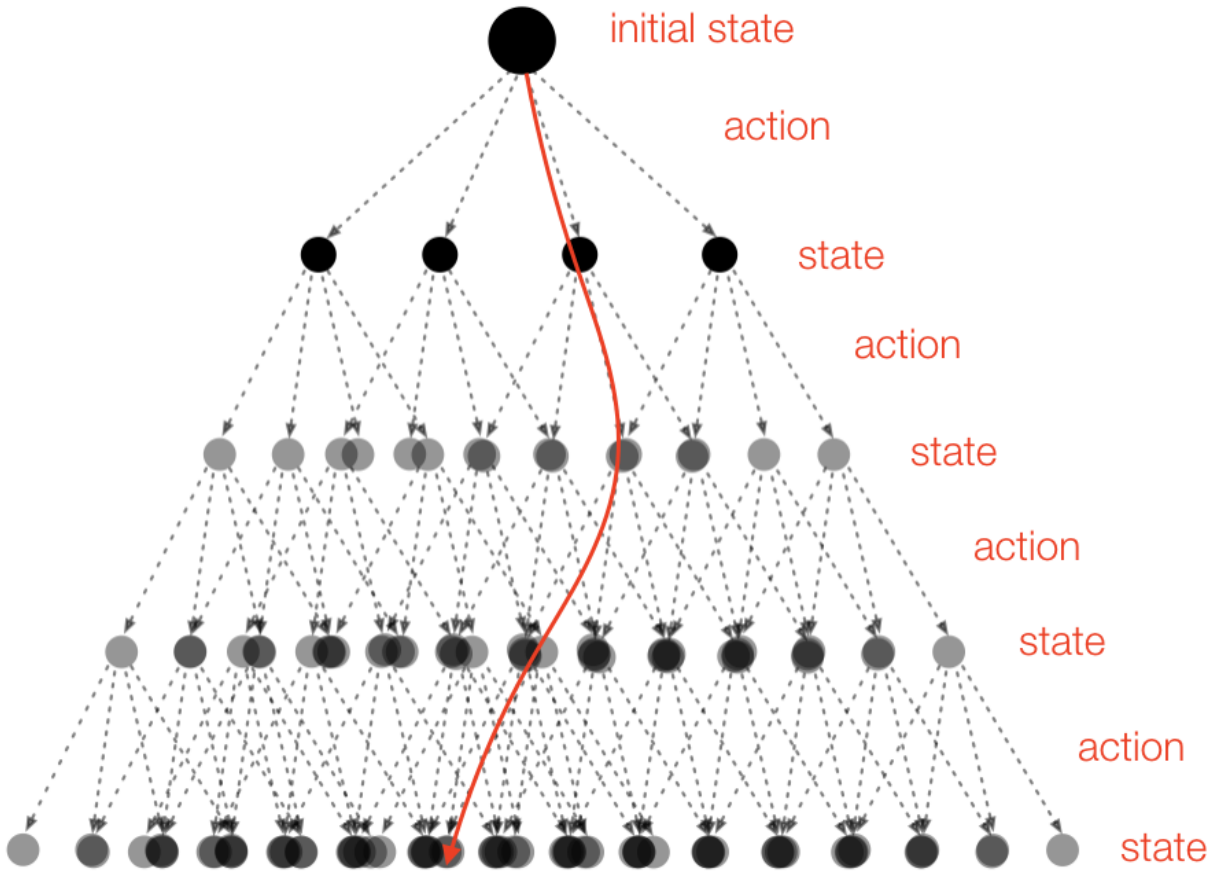
MCTS: Decision Trees in Real Time

"Looking ahead from the current node."



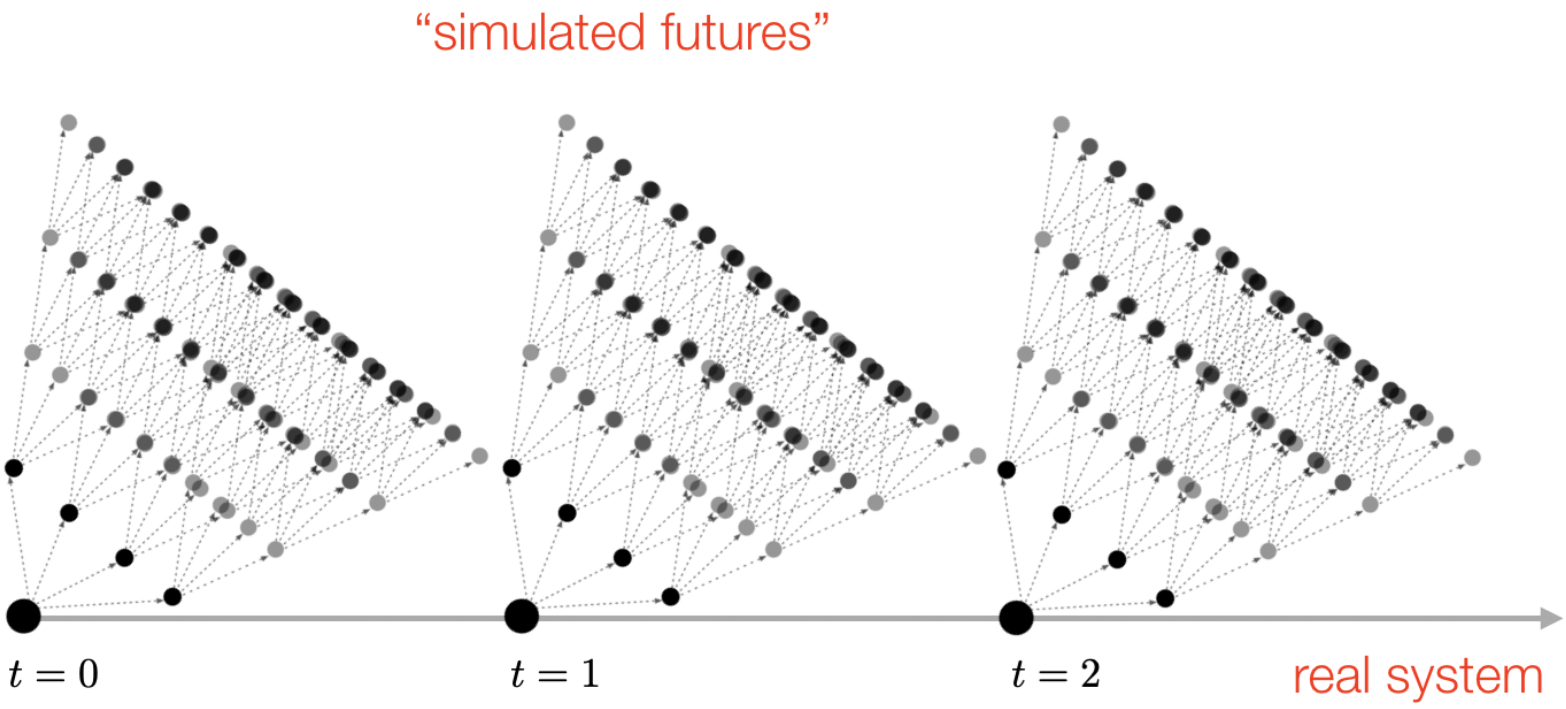
MCTS: Decision Trees in Real Time

"Looking ahead from the current node."

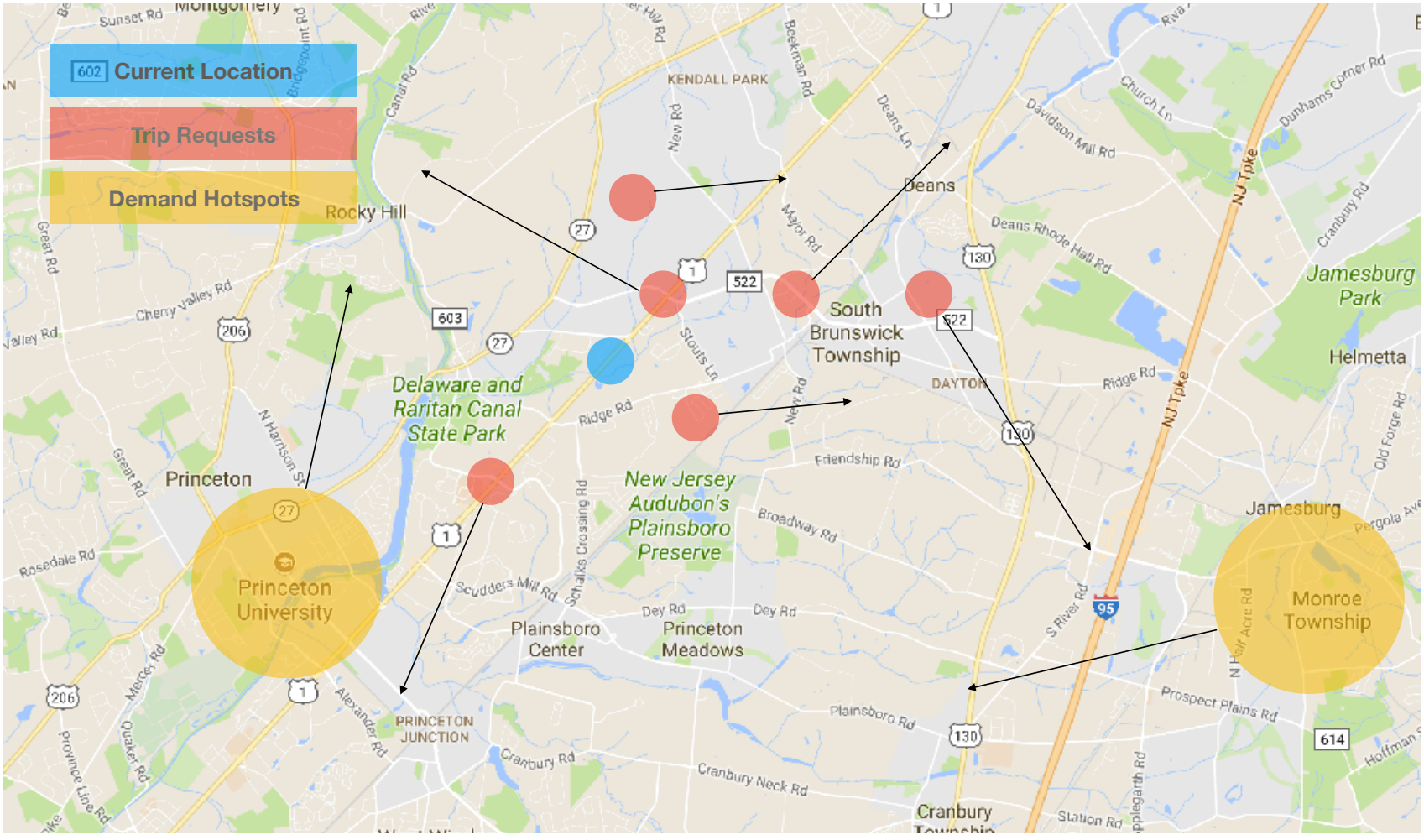


MCTS: Decision Trees in Real Time

"Looking ahead from the current node."



Driver Behavior on a Ride-Sharing Platform

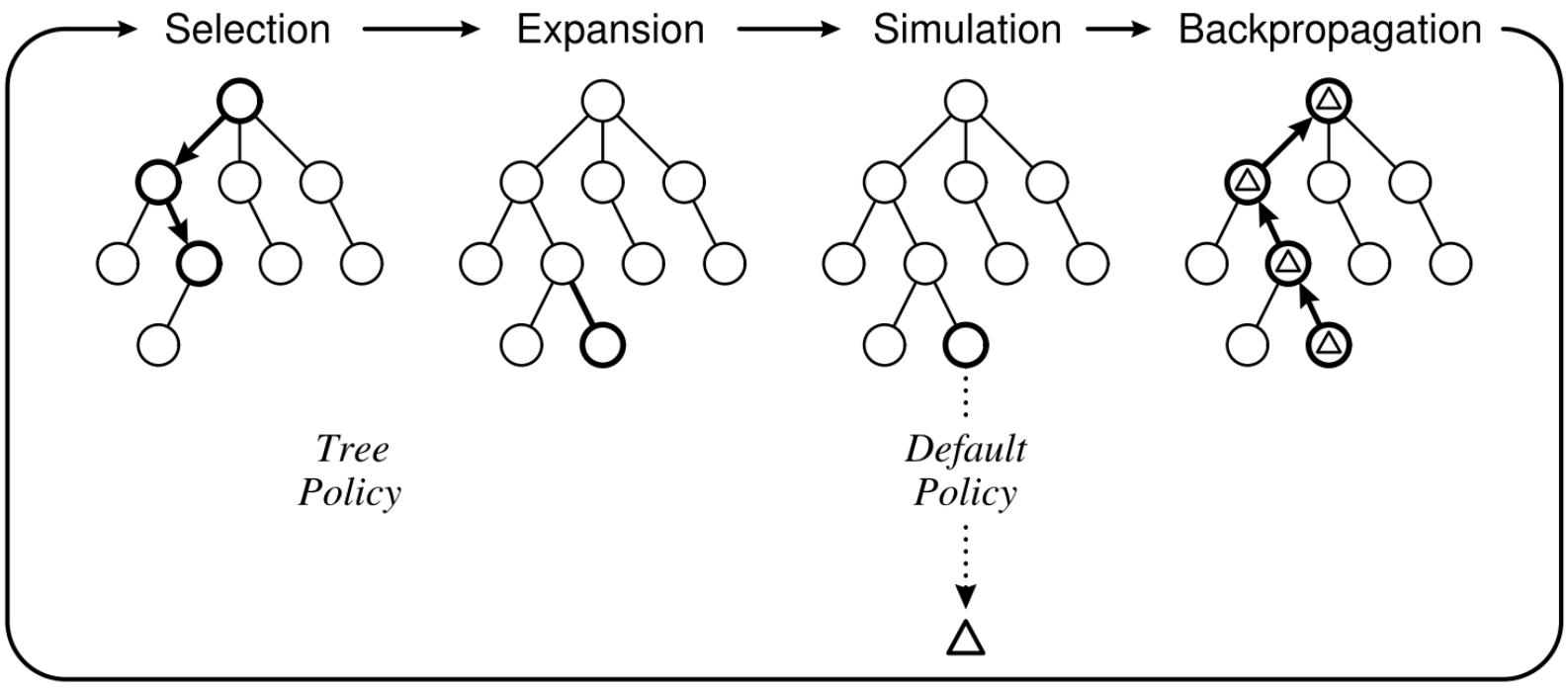


Monte-Carlo Tree Search

- MCTS is a technique, popular in gameplay artificial intelligence (AI), for solving sequential problems (MDPs or two-player games).
- Given an initial state, it **iteratively** builds the decision tree and attempts to focus on states and actions that an optimal policy might visit.
- A heuristic called the **default policy** provides Monte-Carlo estimates of downstream values to guide the tree expansion process.
- The hope is that the optimal action at the root node recommended by the **partial tree** is nearly optimal.
- Works especially well when the **action branching factor** (number of actions) is not too big. This limitation motivates our work.

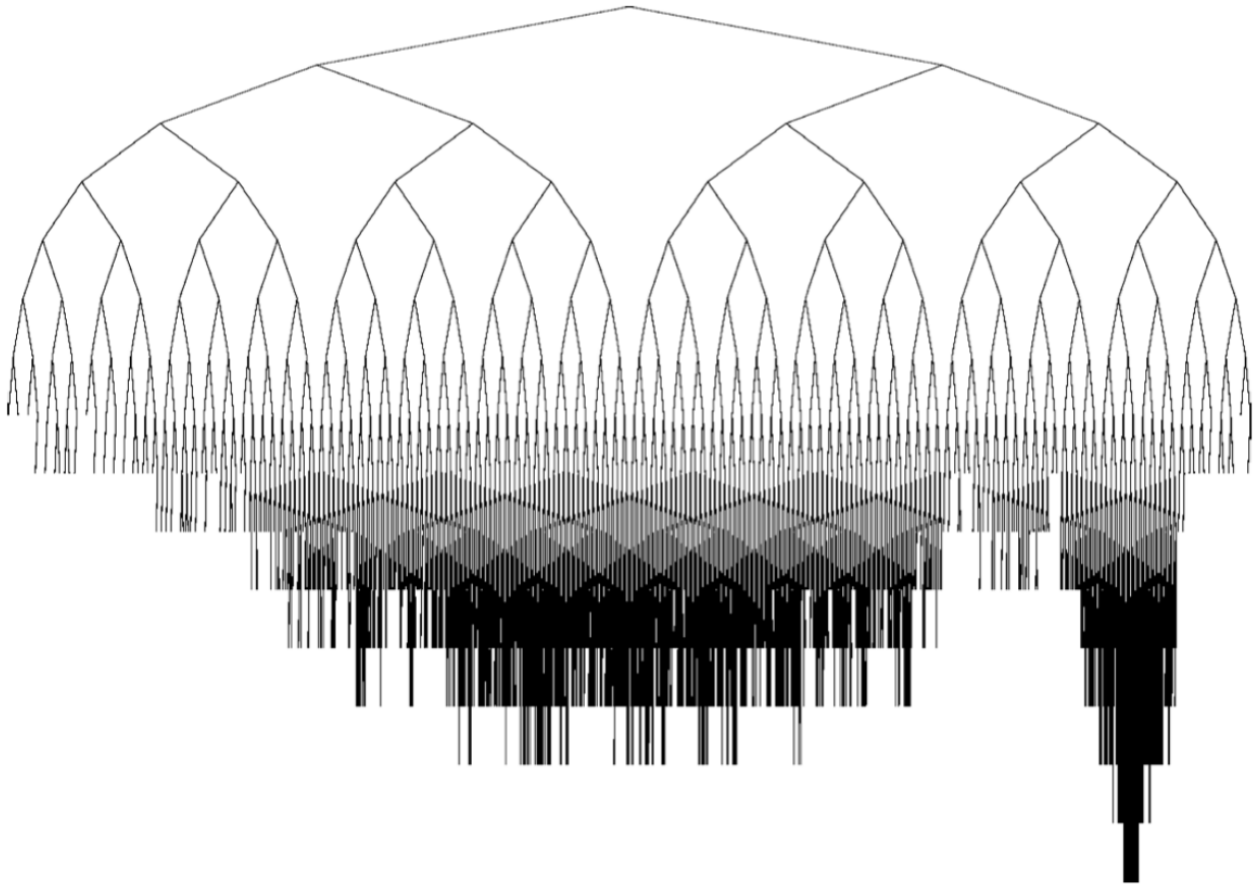
Main Steps of MCTS

(Browne et al., 2002)



An Asymmetric Decision Tree

(Browne et al., 2002)



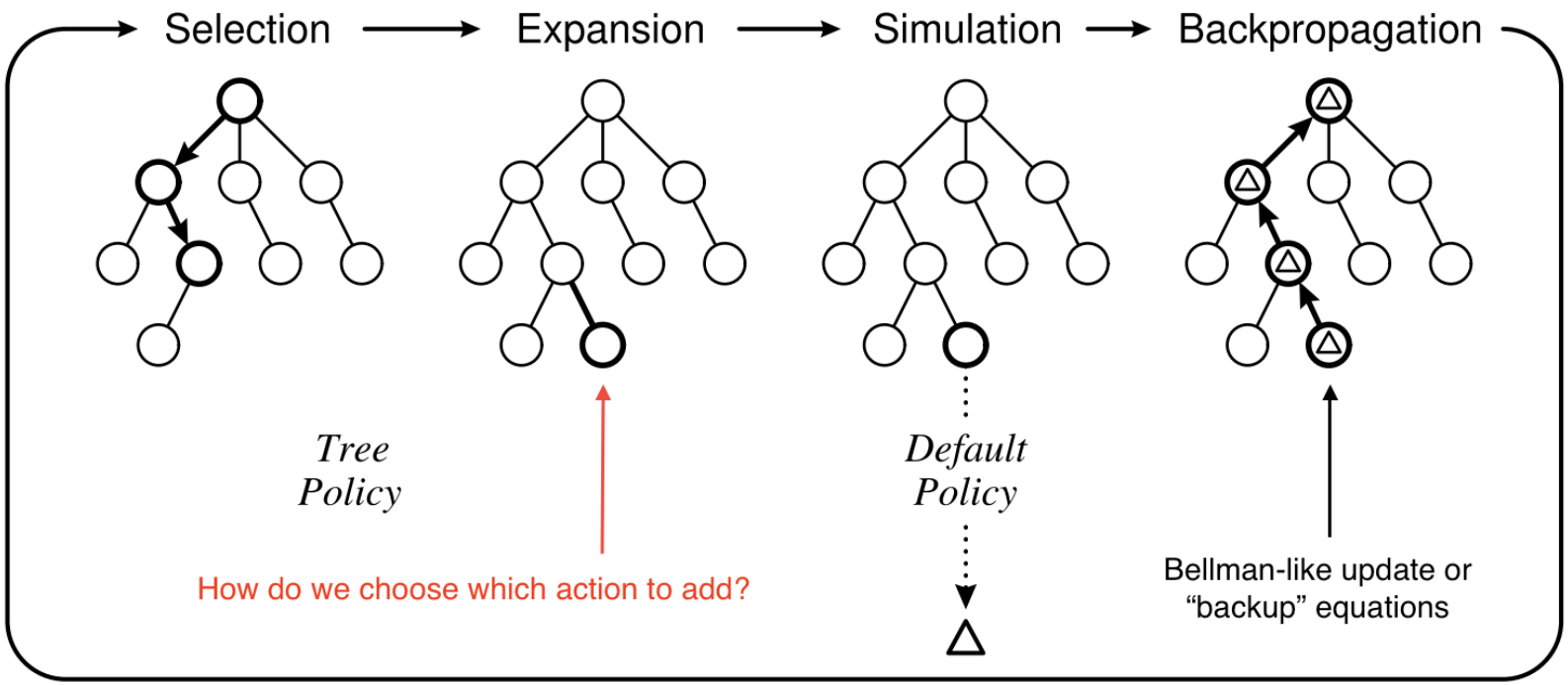
Tree Policy

(Chang et al., 2005[2], Kocsis & Szepesvári, 2006[1], Silver et al. 2016[3])

- **Selection:** A selection policy steers the algorithm down the *current version* of the decision tree.
  - UCT[1] (upper confidence trees) is a version of MCTS where the selection policy views each choice as a **multi-armed bandit**.
  - Based on prior work by called the "adaptive multistage sampling."[2]
  - Traverses down the tree until either an **leaf** or **expandable** (i.e., new child nodes can be added) node is reached.
- **Expansion:** An expansion policy decides which child to add to the tree. Often, a child node is added **at random**.
  - What if there are many actions? Can something more intelligent be done here?
  - AlphaGo[3] uses supervised learning on human moves to intelligently narrow the set of actions to consider.
  - Without supervision, it's more difficult to pre-select actions to consider without first expanding the entire subtree.

Main Steps of MCTS

(Browne et al., 2002)



Motivations

(Bertsimas et al., 2014[1])

1. Every expansion of action is "risky" in the sense that an **entire subtree** now has to be considered. Can we **control the action branching factor** by guessing which actions may be suboptimal (and thus, ignored)? It has been observed[1] that:
   - In an O.R. application, MCTS is competitive with rolling horizon techniques only on smaller instances of the problems.
   - MCTS can be quite sensitive to large action spaces, more so than large state spaces.
2. Can we design a version of MCTS that asymptotically **does not expand the entire tree**, yet is still optimal?
3. Often, the design of the default policy requires significant effort. Can we **further exploit** this heuristic within the MCTS framework?

Outline

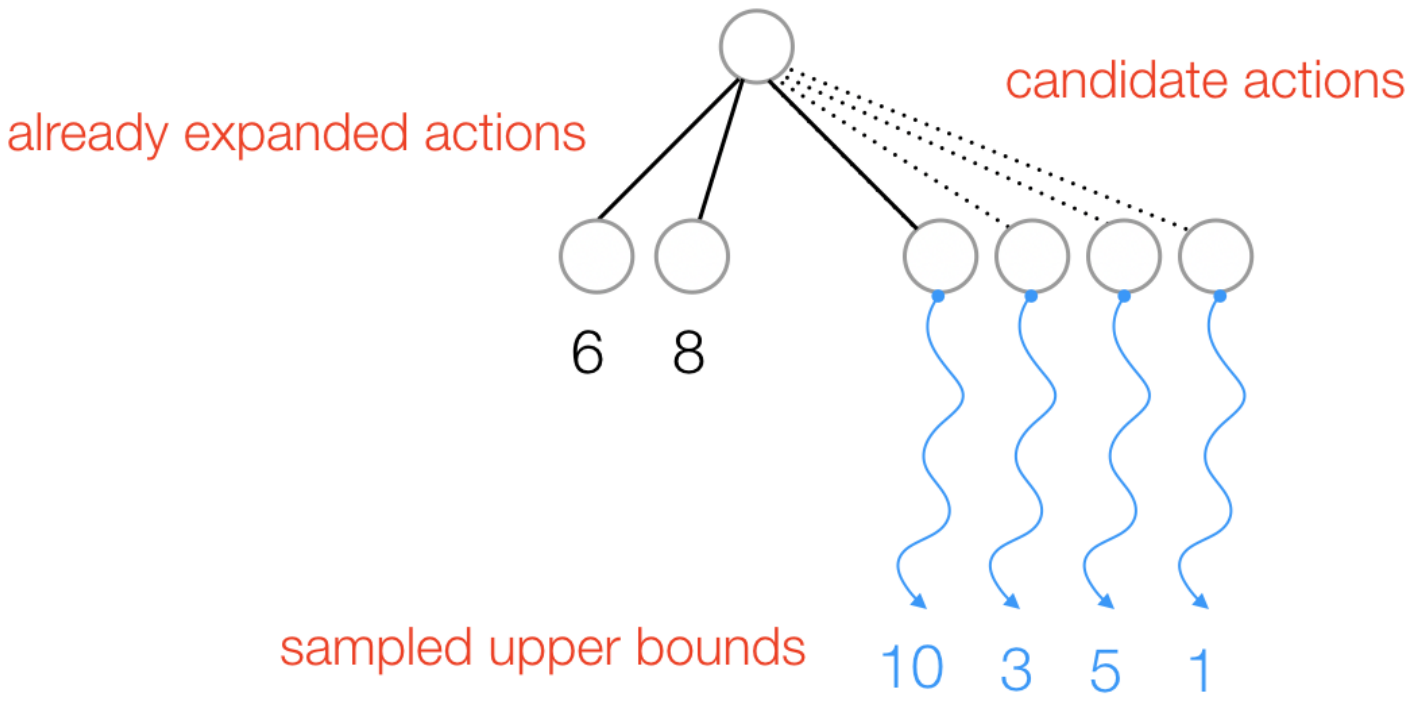
Primal-Dual MCTS

- In standard MCTS, unexpanded action nodes are expanded at random.
  - **Risky:** an entire subtree is now effectively added to the search space.
- Consider the following expansion steps:
  - With each unexpanded action, associate with it an **estimated dual upper bound** $\bar{u}^n$.
  - Obtain a single sample path $\mathbf{W}^n_{t+1,T} = (W^n_{t+1}, \dots W^n_T)$ of the exogenous information process and solve the deterministic "inner" problem:

$$\hat{u}^n(s,a) = r(s,a) + \max_{\mathbf{a}} \left[h_{t+1}\left(S_{t+1}, \mathbf{a}, \mathbf{W}^n_{t+1,T}\right) - \text{penalty}(V^{\pi^{\text{d}}}, \mathbf{a}, \mathbf{W}_{t+1,T}) \right].$$

  - Let $\bar{u}^n$ be a smoothed estimate of $\hat{u}^n$.
  - If none of the actions have a $\bar{u}^n$ higher than the current values of expanded actions, then **don't expand**.
  - Otherwise, **expand** the action with the highest $\bar{u}^n$.

Primal-Dual MCTS



Shortest Path Problem with Random Edge Costs

"Uncertain traffic conditions."

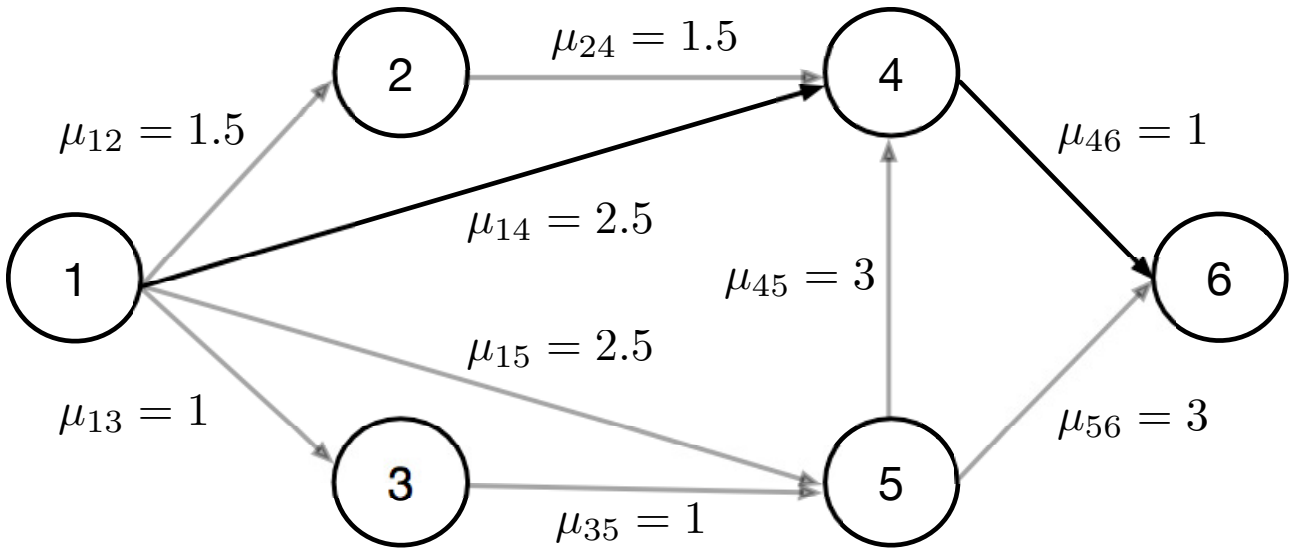


Figure: Graph with Mean Costs

Shortest Path Problem with Random Edge Costs

"A snapshot of the traffic conditions."

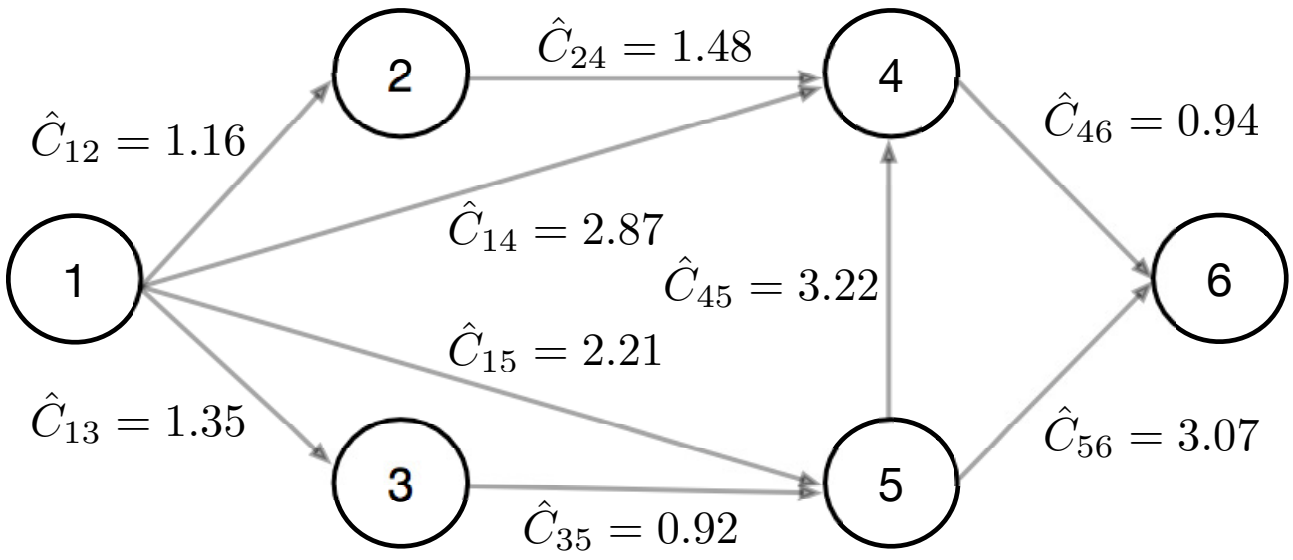
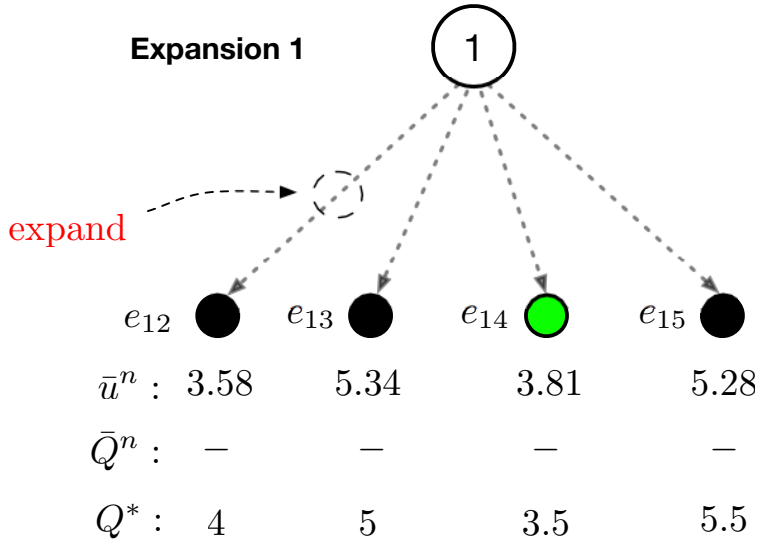
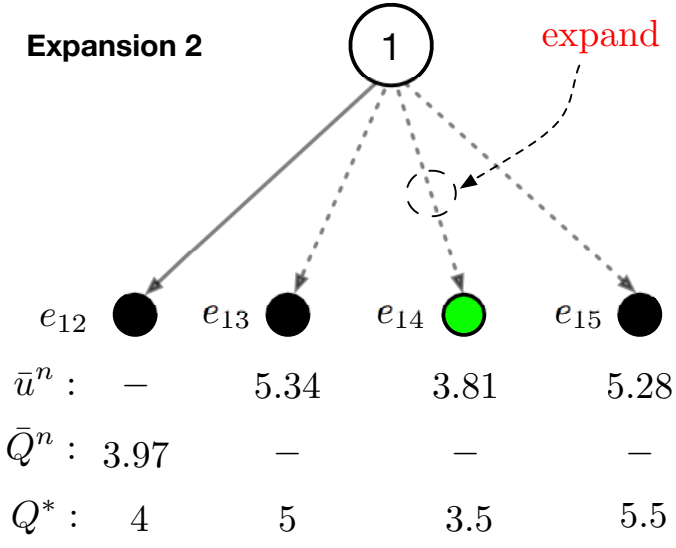
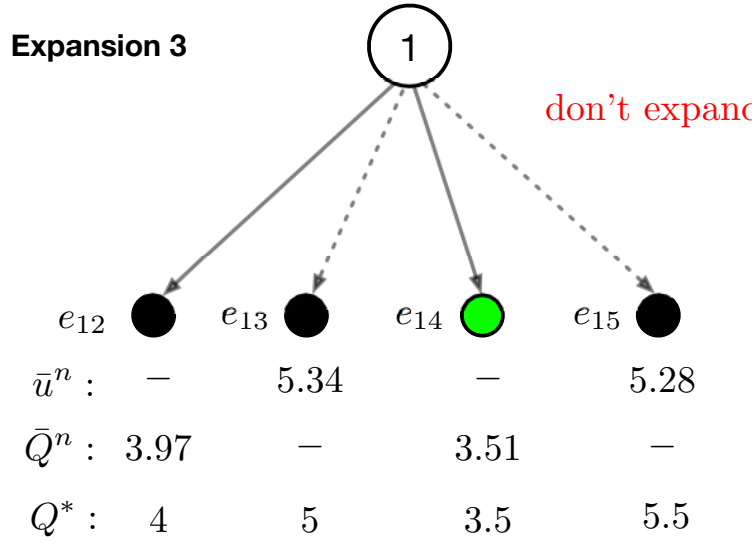


Figure: Graph with Sampled Costs

Shortest Path Problem with Random Edge Costs







Convergence of Primal-Dual MCTS

Theorem. Under some conditions, the root node satisfies:

- $\bar{V}^n(\mathbf{x}_0) \to V^*(\mathbf{x}_0) \;\; a.s.$
- $\limsup_{n\to\infty} \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^n(\mathbf{x}_0)} \bar{Q}^n(\mathbf{y}) \subseteq \operatorname{argmax}_{\mathbf{y}_0=(\mathbf{x}_0,a)} Q^*(\mathbf{y}_0) \;\; a.s.$

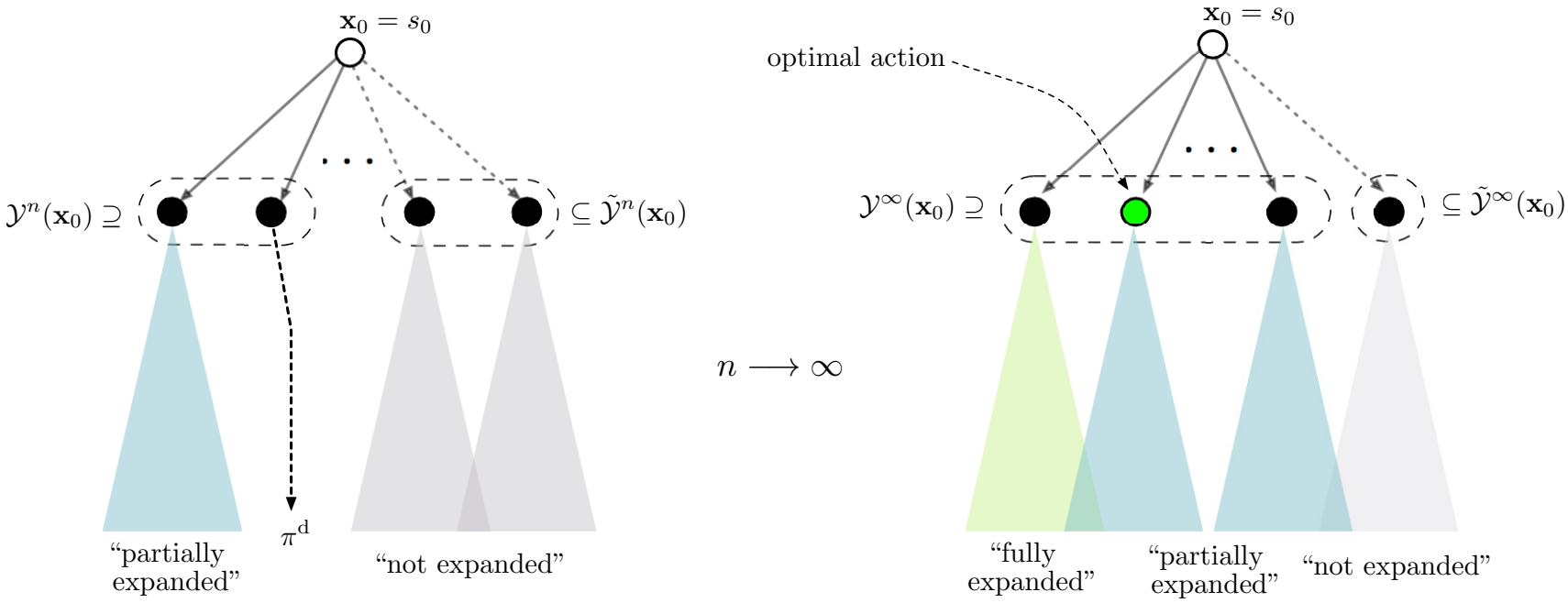
i.e., the estimated value converges to the optimal value and an optimal action is both expanded and identified.

Some Properties of Primal-Dual MCTS

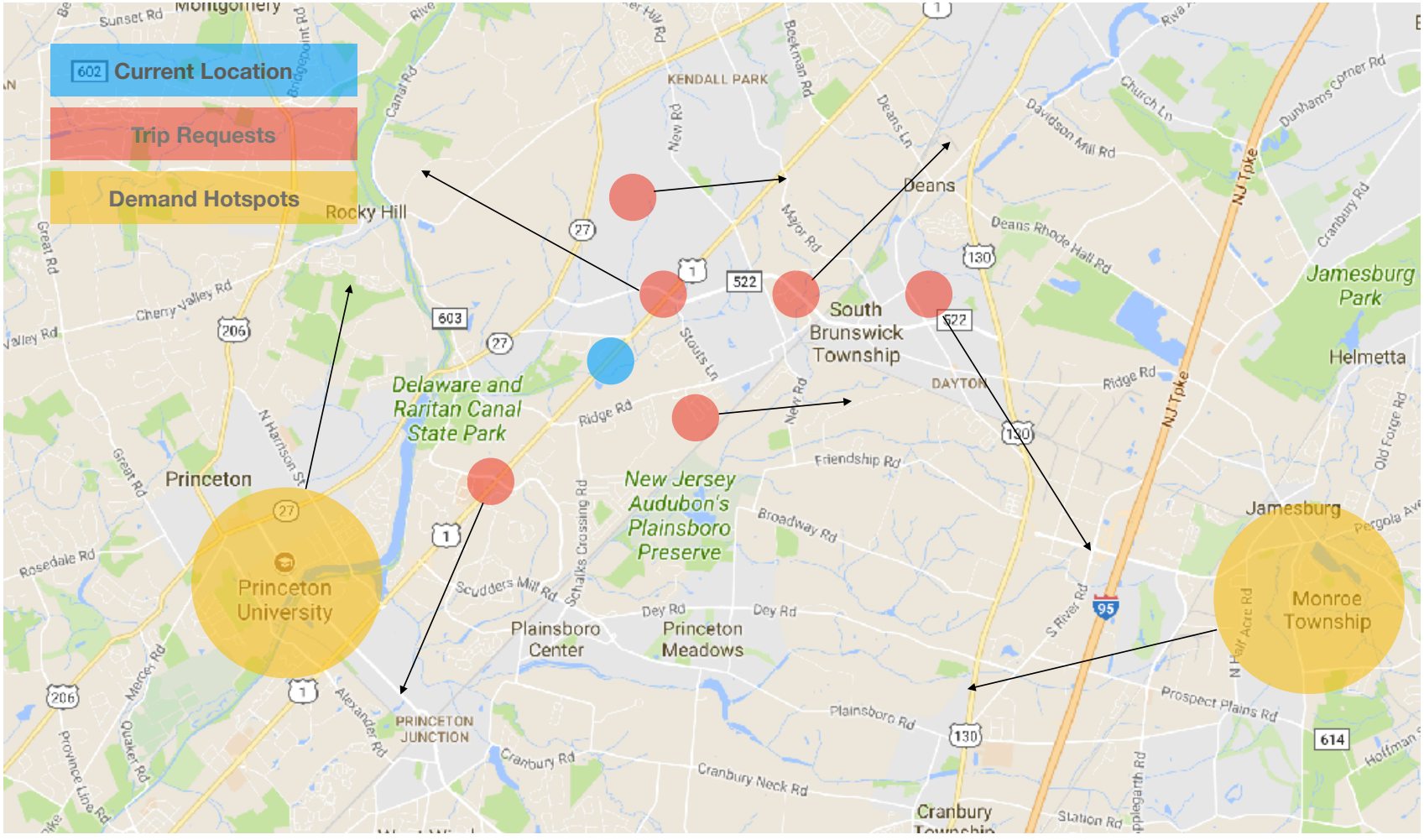
The "limiting partial decision tree" $\mathscr{T}^\infty$

- may have **unexpanded subtrees**;
- the subtree of the optimal action may itself have **unexpanded subtrees**.

Previous work in MCTS (and its variants) obtain convergence by letting the limiting tree become the full tree.



Driver Behavior on a Ride-Sharing Platform

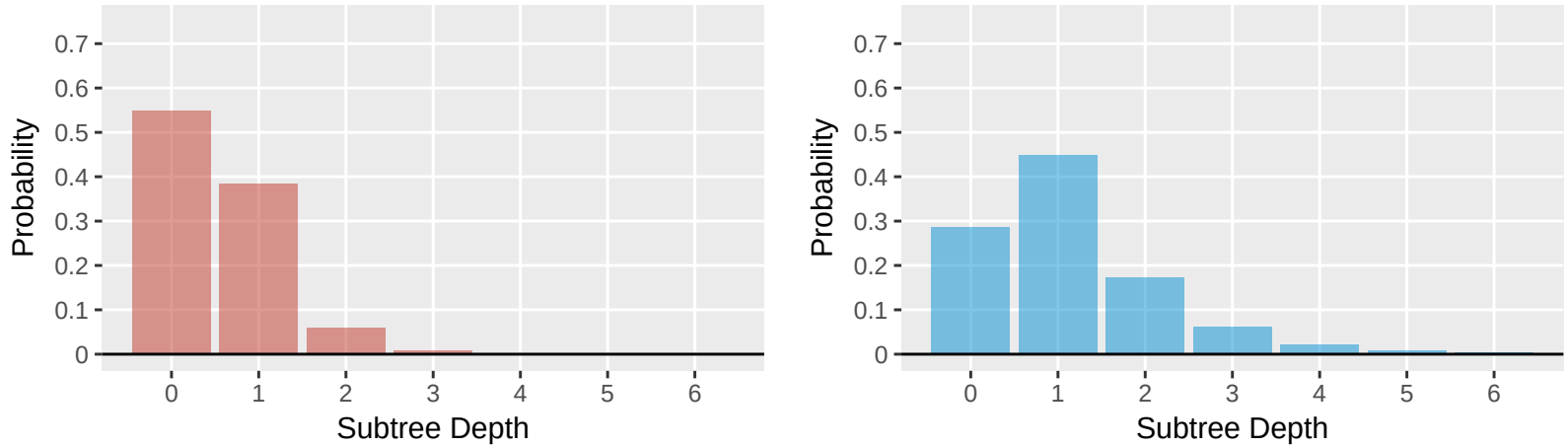


Model Details

- Dataset of trips taken in one day throughout New Jersey on a particular taxi service.
- The driver is assumed to work for 10 hours a day and makes decisions every 15 minutes, giving us $T = 40$.
- Each location in the model represents a 0.5×0.5 square mile area in the state of New Jersey.
- Every period, the driver may choose to either move to an **adjacent location** or select **one of the requests** offered by the platform.
- Objective is to **maximize profit** (taking into account mileage costs).
- Considered many instances of the problem with varying action spaces: `D5` – `D100`, where `D`x means x total actions at each period.

Deeper Trees using Primal-Dual MCTS

Given the same number of iterations, Primal-Dual MCTS produces **deeper trees**. Vanilla MCTS produces wider and shallower trees.



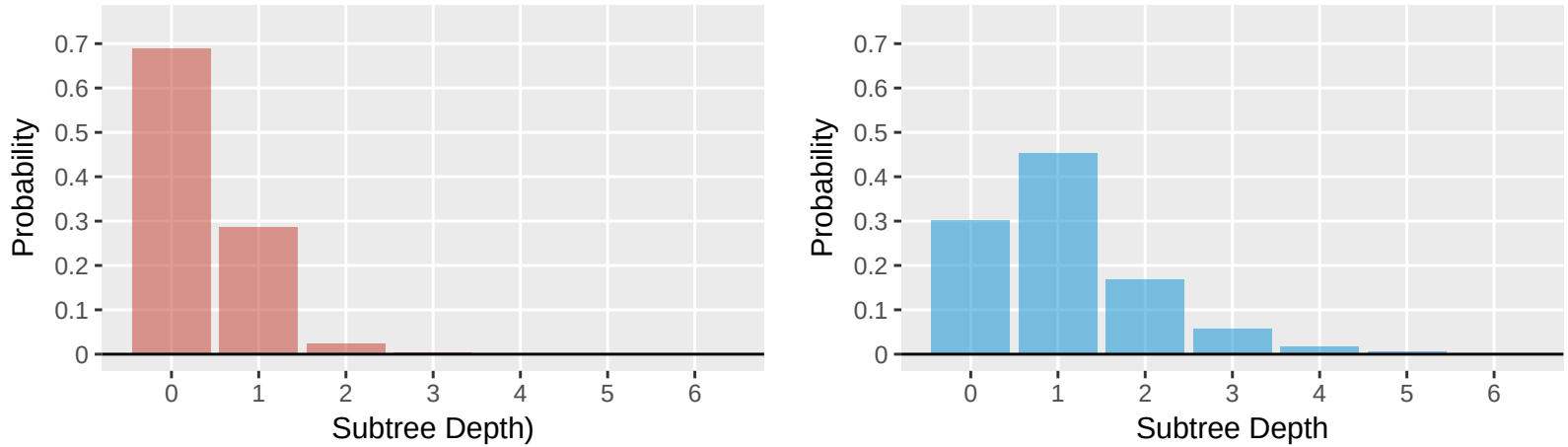
(a) `D5`, Vanilla MCTS

(b) `D5`, Primal-Dual MCTS

Figure: Subtree Depth Empirical Distributions

Deeper Trees using Primal-Dual MCTS

Given the same number of iterations, Primal-Dual MCTS produces **deeper trees**. Vanilla MCTS produces wider and shallower trees.



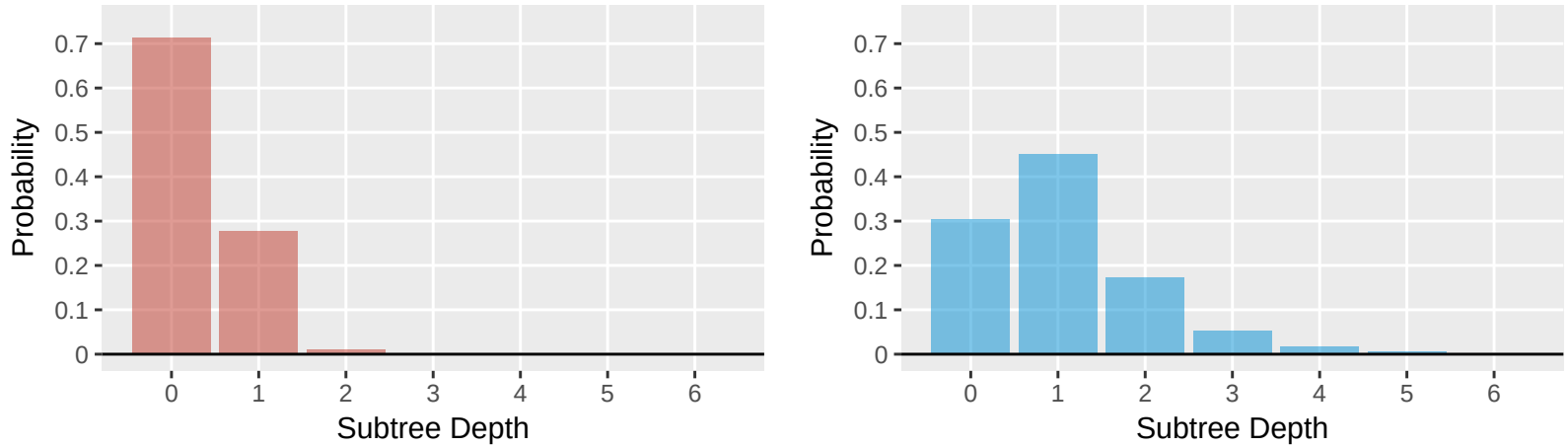
(a) `D10`, Vanilla MCTS

(b) `D10`, Primal-Dual MCTS

Figure: Subtree Depth Empirical Distributions

Deeper Trees using Primal-Dual MCTS

Given the same number of iterations, Primal-Dual MCTS produces **deeper trees**. Vanilla MCTS produces wider and shallower trees.



(a) `D15`, Vanilla MCTS

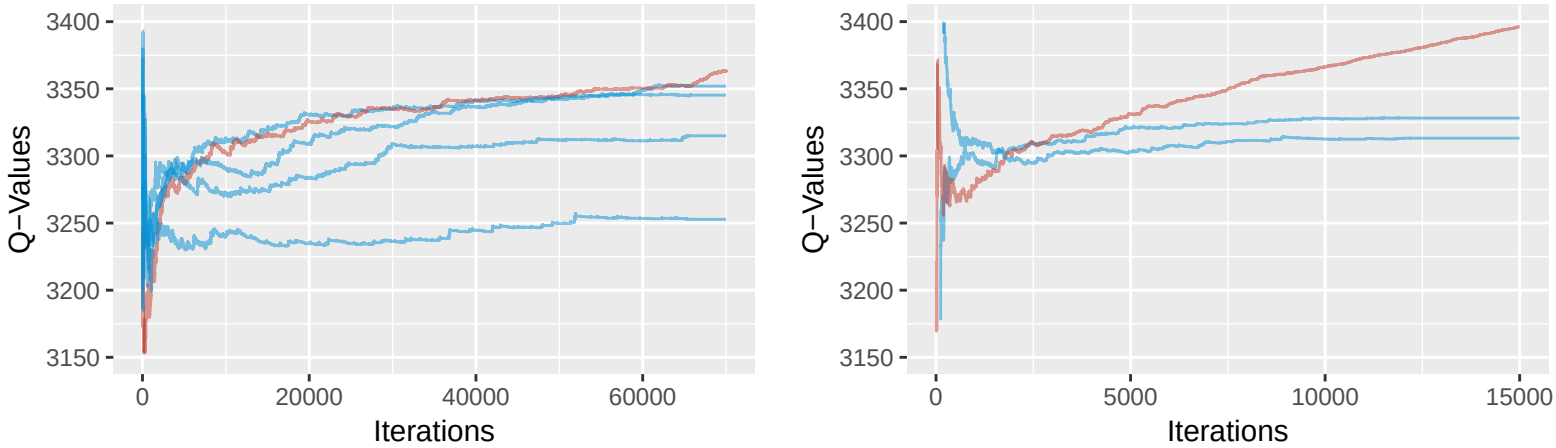
(b) `D15`, Primal-Dual MCTS

Figure: Subtree Depth Empirical Distributions

Convergence of the Action Values

Primal-Dual MCTS is able to discover the "consensus decision" (between the two algorithms) **in fewer iterations** because it does not expand all actions.

- 2795 iterations versus 65,611 iterations.
- The CPU usage until convergence is 27% of vanilla MCTS.

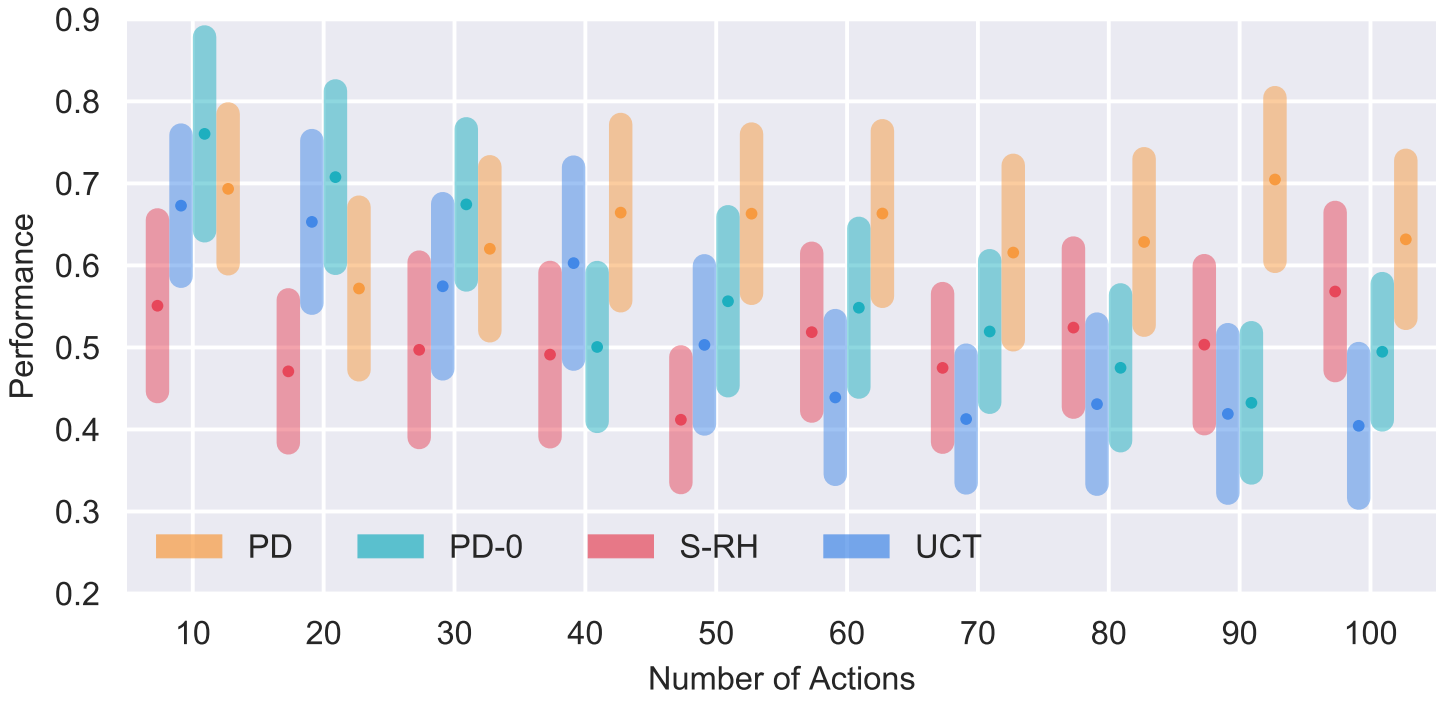


(a) `D5`, Vanilla MCTS (b) `D5`, Primal-Dual MCTS

Figure: Convergence of Q-Values of State-Action Nodes

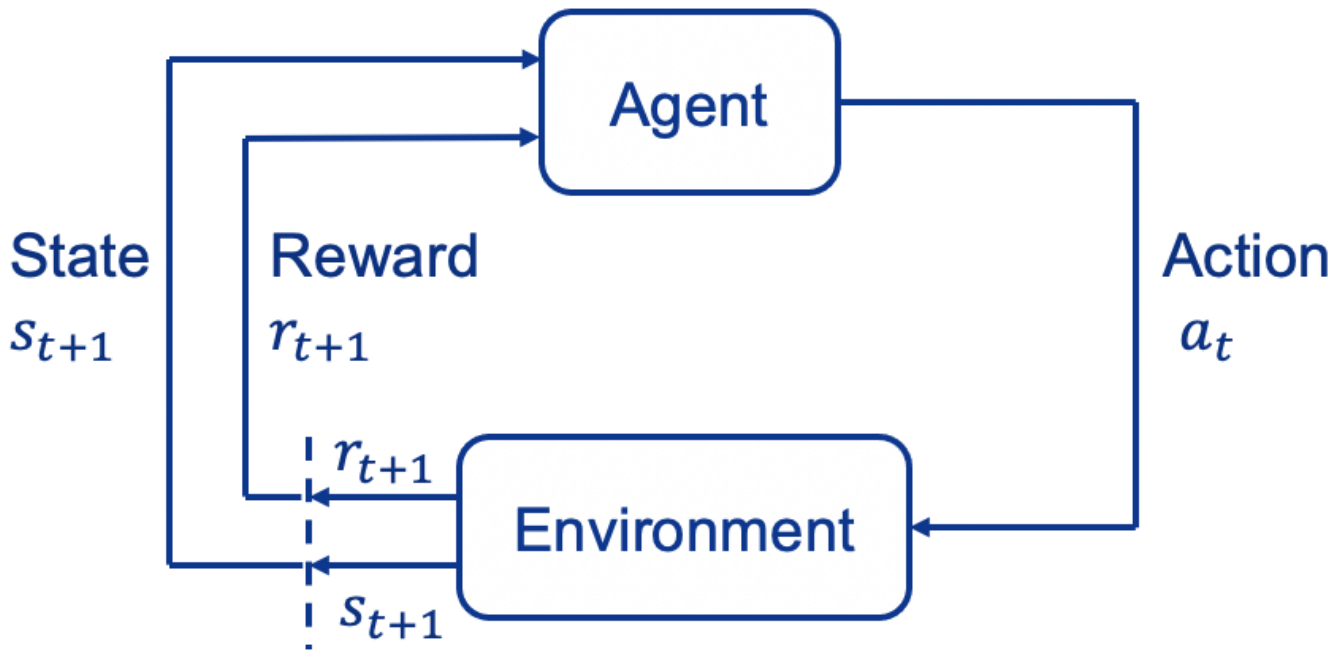
Comparisons to Other Policies

Performance improvement is most noticeable on larger problems.



Outline

The RL Setting



Q-Learning

(Watkins, 1989)

- Q-learning is *the* canonical reinforcement learning algorithm (enjoys conceptual simplicity, ease of implementation, convergence guarantees).
- It can be thought of as an *asynchronous* and *sampled* version of the value-iteration algorithm for MDPs.
- Has gained renewed attention in recent years after serving as the core underlying methodology of several successful deep reinforcement learning algorithms, such as DQN (Mnih et al., 2013), Double DQN (van Hasselt et al., 2016), and Rainbow DQN (Hessel et al., 2018).
- Q-learning and its variants are known to be challenging to apply to real-world settings, due to the cost of collecting experience, due to expensive simulations/real-world interactions.

Q-Learning

(Watkins, 1989)

- Consider a γ-discounted *infinite horizon* problem with finite state and action spaces. The state-action value function of a policy π is:

$$Q^{\pi}(s,a) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) \,\middle|\, s_0 = s, a_0 = a, \pi\right].$$

- An optimal policy selects actions according to

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s,a),$$

where $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ is the optimal action-value function.

- The Bellman optimality equation gives the following recursion:

$$Q^*(s_t,a_t) = r(s_t,a_t) + \gamma\, \mathbf{E}\left[\max_{a_{t+1}} Q^*(s_{t+1},a_{t+1})\right].$$

Q-Learning

(Watkins, 1989)

Algorithm 1: Q-Learning

Input: Initial estimate Q_0 and stepsize rule $\alpha_n(s,a)$.
Output: Approximations $\{Q_n\}$.

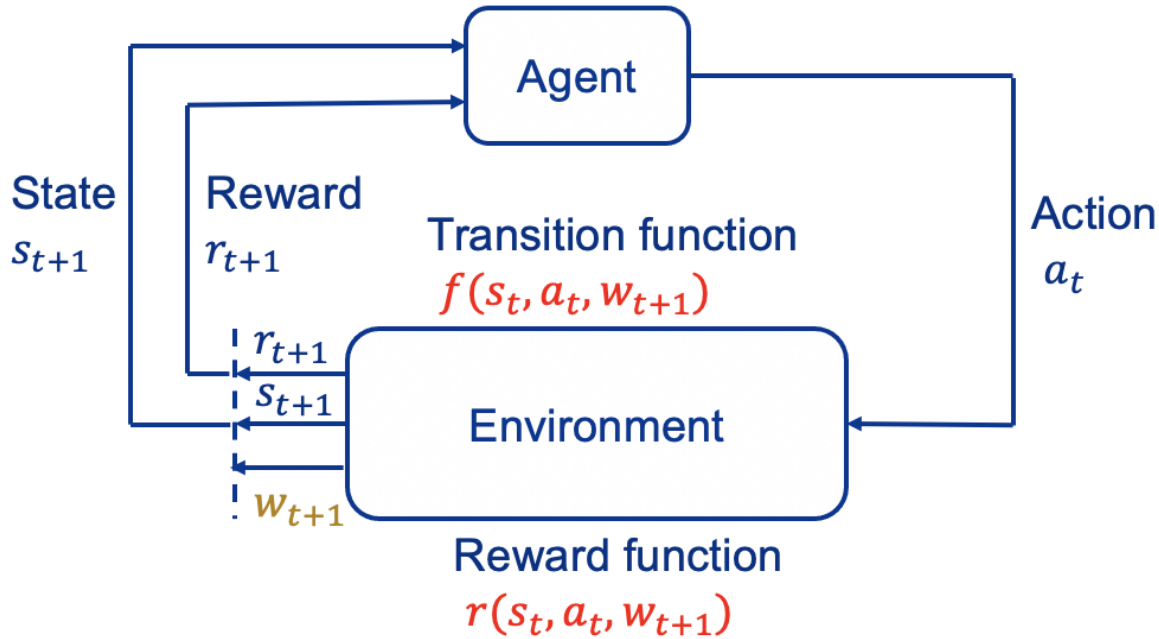
Choose an initial state s_0.
for $n = 0, 1, 2, \ldots$ **do**
  Choose an action a_n via some behavior policy (e.g., ϵ-greedy). Observe w_{n+1}.
  Perform a standard Q-learning update:

$$Q_{n+1}(s_n, a_n) = Q_n(s_n, a_n) + \alpha_n(s_n, a_n)\Big[r(s_n, a_n) + \gamma \max_a Q_n(s_{n+1}, a) - Q_n(s_n, a_n)\Big].$$

end

- **Question 1:** If we knew upper and lower bounds on Q^*, could we improve this algorithm?
- **Question 2:** Can we dynamically estimate upper and lower bounds and get better over time?

Our Setting: Problems with Partially Known Dynamics



- It is often the case that we know the transition function f and reward r. In other words, *given* s_t, a_t, and w_{t+1}, we can evaluate the next state and the reward.[2]

- What is often **not known** is the random noise w_{t+1} (range of values, distribution, etc). But w_{t+1} is often *observable* (prices, weather patterns, demands, etc).

[2] Typical RL setting: one observes s_{t+1} and r_{t+1} after taking action a_t in state s_t.

Examples of Problems with Partially Known Dynamics

- Although not the standard RL paradigm, typical assumption in the OR and control theory literature.
- Backed by abundant real-world applications:
  - Inventory problems,
  - Car-sharing problems,
  - Vehicle routing,
  - Renewable energy,
  - Maintenance problems,
  - Option pricing and trading.

Infinite Horizon Information Relaxation Dual Bound

(Brown and Haugh, 2017)

Let $\zeta_t^{\pi}(s_t, a_t, w_{t+1} \,|\, \varphi)$ be a penalty function generated by some function φ (e.g., $\varphi = Q^*$) and policy π (e.g., $\pi = \pi_\varphi$):

$$\begin{aligned}&\zeta_t^{\pi}(s_t, a_t, w_{t+1} \,|\, \varphi) \\ &\quad = \gamma^{t+1}\Big(\varphi(s_{t+1}, \pi(s_{t+1})) - \mathbf{E}\Big[\varphi\big(f(s_t, a_t, w), \pi(f(s_t, a_t, w))\big)\Big]\Big).\end{aligned}$$

Then, it holds that:

$$Q^*(s_0, a_0) \leq \mathbf{E}\left[\max_{\mathbf{a}}\left\{\sum_{t=0}^{\infty}\gamma^t\Big(r(s_t, a_t) - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi)\Big)\right\}\right],$$

where π_φ = greedy policy with respect to the value function φ.

Two challenges here are (1) computing the inner expecation (due to not having the distribution of w) and (2) maximizing over an infinite sequence of actions $\mathbf{a}$.

Inner Expecation? Infinite Sequence of Actions?

Empirical Penalty: Given K observations of exogenous information $\{w^1, \ldots, w^K\}$,

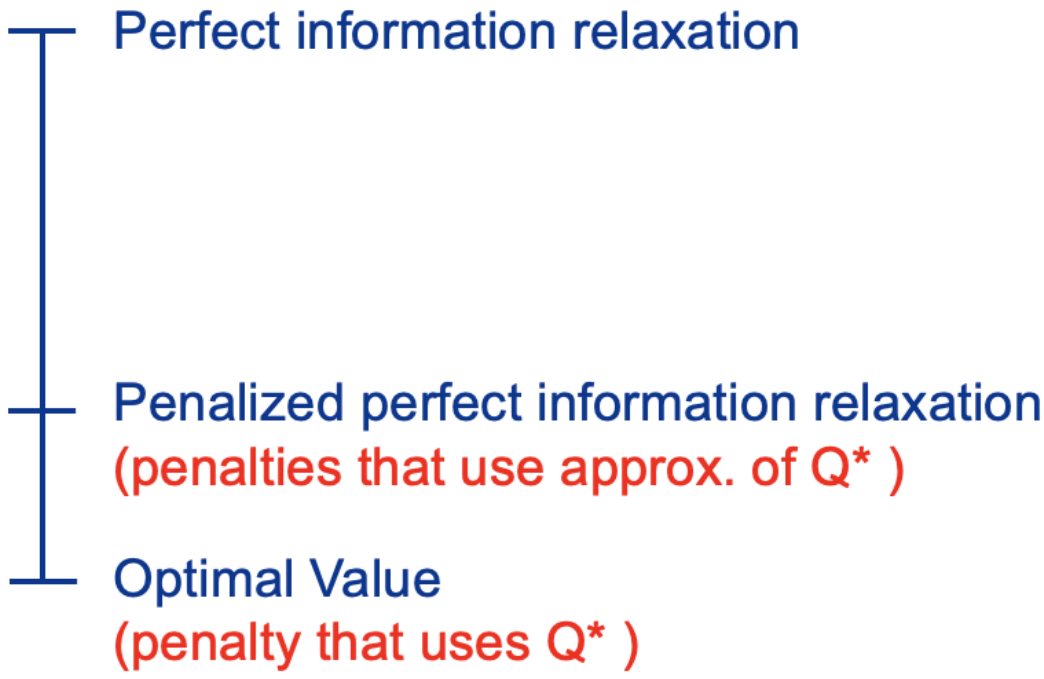
$$\hat{\zeta}_t^{\pi}(s_t, a_t, w_{t+1} \mid \varphi)$$
$$= \gamma^{t+1}\Big(\varphi(s_{t+1}, \pi(s_{t+1})) - \frac{1}{K}\sum_{k=1}^{K}\Big[\varphi\big(f(s_t, a_t, w^k), \pi(f(s_t, a_t, w^k))\big)\Big]\Big).$$

Indefinite Horizon: Solve an undiscounted τ-horizon MDP where each state transitions to zero reward terminal state w.p. $1-\gamma$ (thus, $\tau \sim \text{Geom}(\gamma)$):

$$Q^*(s_0, a_0) \leq \mathbf{E}\left[\max_{\mathbf{a}}\left\{\sum_{t=0}^{\tau}\Big(r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \mid \varphi)\Big)\right\}\right].$$

(Theory still holds using these two changes.)

Infinite Horizon Information Relaxation Dual Bound



Sampled Infinite Horizon Information Relaxation Dual Bound

For a given sample path $\mathbf{w} = (w_1, \ldots, w_\tau)$ where $\tau \sim \text{Geom}(\gamma)$, we can compute sampled bounds.

Upper Bound: Each of the "inner" DP problems can be solved via the backward recursion

$$Q_t^U(s_t, a_t) = r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + \max_a Q_{t+1}^U(s_{t+1}, a),$$

for $t = \tau - 1, \tau - 2, \ldots, 0$ with $s_{t+1} = h(s_t, a_t, w_{t+1})$ and $Q_\tau^U \equiv 0$ (as there is no additional reward after entering the absorbing state $\tilde{s}$). The optimal value of the inner problem is given by $Q_0^U(s_0, a_0)$.

Lower Bound: A sampled lower bound is:

$$Q_t^L(s_t, a_t) = r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1} \,|\, \varphi) + Q_{t+1}^L(s_{t+1}, \pi(s_{t+1})),$$

for $t = 0, \ldots, \tau - 1$, with $s_{t+1} = h(s_t, a_t, w_{t+1})$ and $Q_\tau^L \equiv 0$.

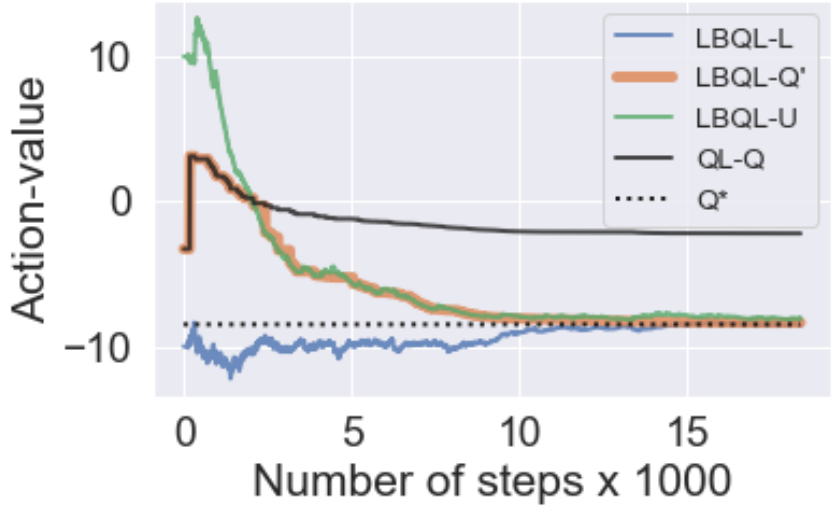
Behavior of an Algorithm that Uses Dynamically Learned Bounds

(Showing some "intended" results before the proposed algorithm.)

Ingredients:

- Q-learning, which produces a sequence of value function approximations converging to Q^*,
- Bound procedure (input: value function approximation, output: estimated lower and upper bounds), with the property that better value function approximations produce tighter bounds (Prop. 2.3 of Brown et al., 2010).

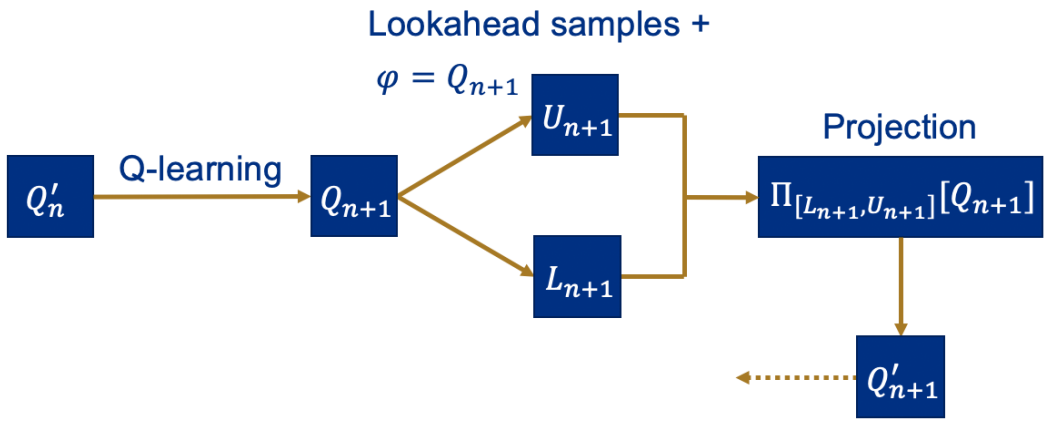
Could we use these ingredients to design an algorithm that uses these bounds to "squeeze" the iterates toward the optimal value (something like the plot below)?



Q-Learning with Lookahead Upper and Lower Bounds

Main Idea: Generate improving upper/lower bounds such that the Q-iterates are "squeezed" toward optimality by setting φ to the current Q-iterate.

1. On a given iteration n, experience a realization of the exogenous information w_{t+1} and make a standard Q-learning update.
2. We then set φ to be the newly updated Q-iterate and compute upper and lower bounds on the true Q^*, which are then tracked and averaged using a stochastic approximation step.
3. Finally, we project the Q-iterate so that it satisfies the averaged upper and lower bounds and return to Step 1.



Lookahead-Bounded Q-Learning

Q improves $\Rightarrow$ bounds improve $\Rightarrow$ Q improves more $\Rightarrow$ bounds improve more

Algorithm 2: Lookahead-Bounded Q-Learning

Input: Initial estimates $L_0 \leq Q_0 \leq U_0$, batch size K, and stepsize rules $\alpha_n(s,a)$, $\beta_n(s,a)$.
Output: Approximations $\{L_n\}$, $\{Q'_n\}$, and $\{U_n\}$.

Set $Q'_0 = Q_0$ and choose an initial state s_0.
for $n = 0, 1, 2, \ldots$ **do**
  Choose an action a_n via some behavior policy (e.g., ϵ-greedy). Observe w_{n+1}.
  Perform a standard Q-learning update:

$$Q_{n+1}(s_n, a_n) = Q'_n(s_n, a_n) + \alpha_n(s_n, a_n)\Big[r_n(s_n, a_n) + \gamma \max_a Q'_n(s_{n+1}, a) - Q'_n(s_n, a_n)\Big].$$

  Set $\varphi = Q_{n+1}$. Using one sample path $\mathbf{w} = (w_1, w_2, \ldots, w_\tau)$, compute noisy bounds $\hat{Q}_0^U(s_n, a_n)$ and $\hat{Q}_0^L(s_n, a_n)$ by solving a deterministic DP / "simulating forward."
  Update and enforce upper and lower bounds:

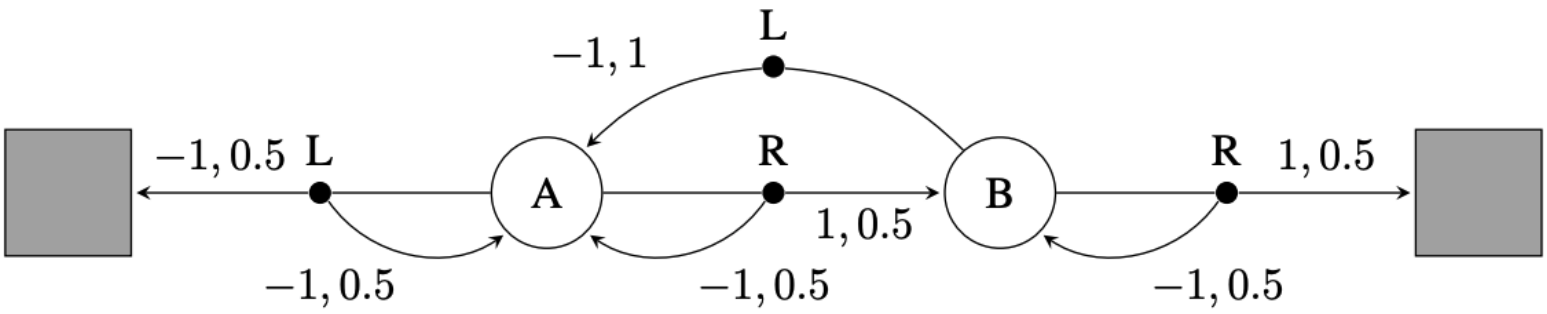
$$U_{n+1}(s_n, a_n) = U_n(s_n, a_n) + \beta_n(s_n, a_n)\,[\hat{Q}_0^U(s_n, a_n) - U_n(s_n, a_n)], \tag{11}$$

$$L_{n+1}(s_n, a_n) = L_n(s_n, a_n) + \beta_n(s_n, a_n)\,[\hat{Q}_0^L(s_n, a_n) - L_n(s_n, a_n)], \tag{12}$$

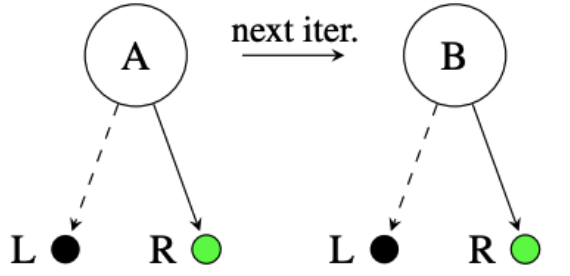
$$Q'_{n+1}(s_n, a_n) = \max\{\min\{U_{n+1}(s_n, a_n), Q_{n+1}(s_n, a_n)\}, L_{n+1}(s_n, a_n)\}.$$

end

An Example of LBQL on a Simple Problem

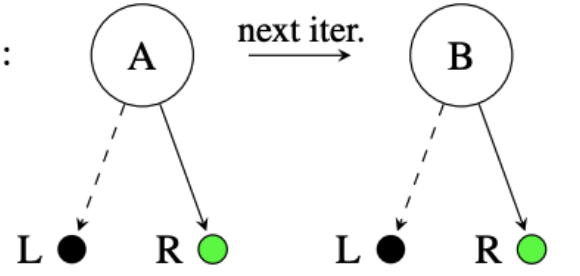


Episode 1:



	L	R	L	R
Q_n :	0	−0.1	0	−0.1
L_n :	−20	−18.9	−20	−18.98
Q'_n :	0	−0.1	0	−0.1
U_n :	20	19.3	20	−19.02

Episode 58 :



	L	R	L	R
Q_n :	−1.05	0.15	−0.87	−0.12
L_n :	−5.56	−0.12	−8.29	−0.08
Q'_n :	−1.05	0.07	−0.87	−0.08
U_n :	4.08	0.07	4.98	0.05

Convergence

Lemma (Asymptotic Bounds on L_n and U_n). For all $(s, a) \in \mathcal{S} \times \mathcal{A}$, we have the following:

1. If $L_0(s, a) \leq U_0(s, a)$, then $L_n(s, a) \leq U_n(s, a)$ for all iterations n.

2. For every $\eta > 0$, and with probability one, there exists some finite iteration index n_0 such that

$$L_n(s, a) \leq Q^*(s, a) + \eta \quad \text{and} \quad Q^*(s, a) - \eta \leq U_n(s, a)$$

for all iterations $n \geq n_0$.

Convergence

Theorem (Convergence of LBQL). Suppose that:

1. $\sum_{n=0}^{\infty} \alpha_n(s,a) = \infty$, $\sum_{n=0}^{\infty} \alpha_n^2(s,a) < \infty$,
2. $\sum_{n=0}^{\infty} \beta_n(s,a) = \infty$, $\sum_{n=0}^{\infty} (\beta_n(s,a))^2 < \infty$,
3. Each state $s \in \mathcal{S}$ is visited infinitely often with probability one.

The following hold:

1. With probability one, $Q'_n(s,a)$ converges to the optimal action-value function $Q^*(s,a)$ for all state-action pairs (s,a).
2. If the penalty terms were computed exactly, then with probability one, the iterates $L_n(s,a), Q'_n(s,a), U_n(s,a)$ converge to the optimal action-value function $Q^*(s,a)$ for all state-action pairs (s,a).

Practical Extension: LBQL with Experience Replay

Issue: In a real setting, we cannot get samples of the exogenous information w_{t+1}, let alone use its distribution. Instead, we can:

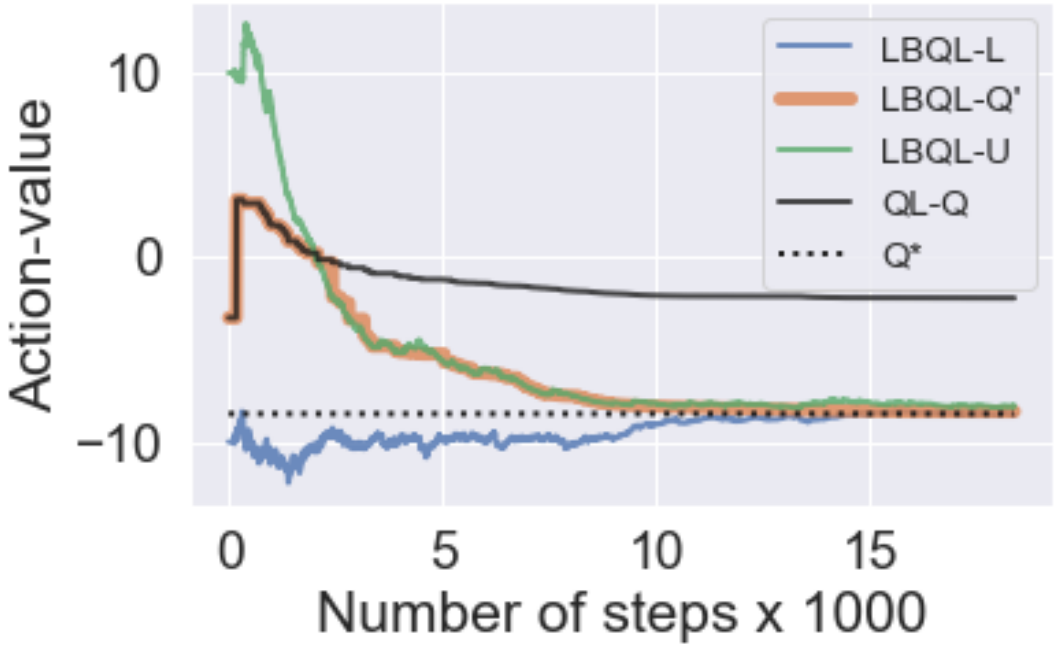
1. Use a noise buffer $\mathcal{B}$ is used to record observed w's. We can then sample from the buffer $\mathcal{B}$ whenever we need samples of w (both batch samples for computing the expecation within the penalty, and sample path to compute the inner DP).
2. The same convergence results can be obtained for this practical extension (but need to account for additional bias due to sampling from the buffer).

Numerical Illustrations

- Windy Gridworld
  - *There is an upward crosswind with a random intensity, and the agent moves extra steps in the wind direction whenever it reaches an affected square.*
- Stormy Gridworld
  - *We then consider a new domain that adds the additional complexity of rain and multi-directional wind to windy gridworld. The location of the rain is random and when it occurs, puddles that provide negative rewards are created.*
- Spatial Pricing for Carsharing (2 and 4 Locations)
  - *The actions are to set a price at each station, which influence the station's (stochastic) demand for rentals. Rentals can either be one-way or round-trip. The goal is to maximize revenue, where unmet demands are charged a lost sales cost. The largest version of this problem has 1771 states and 16 actions.*

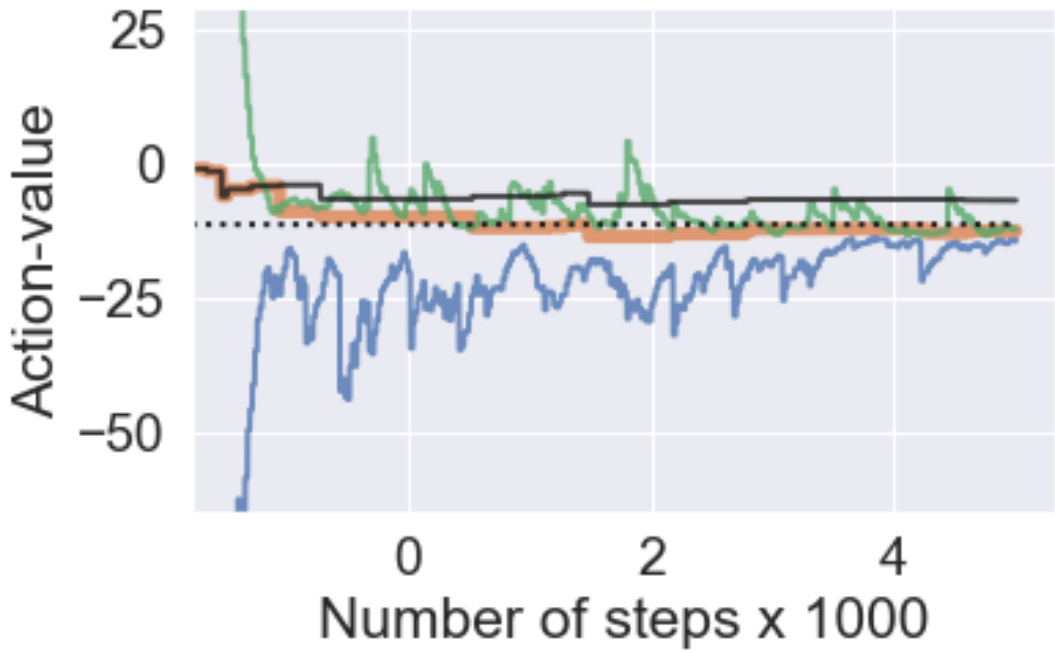
Behavior of LBQL (Windy Gridworld)

Q improves $\Rightarrow$ bounds improve $\Rightarrow$ Q improves more $\Rightarrow$ bounds improve more



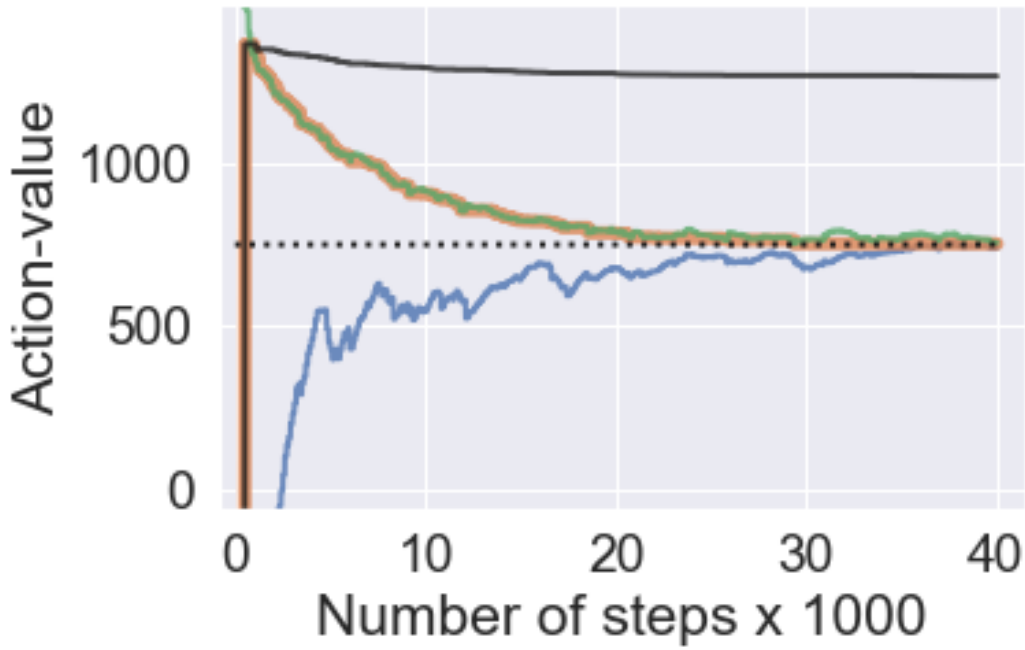
Behavior of LBQL (Stormy Gridworld)

Q improves $\Rightarrow$ bounds improve $\Rightarrow$ Q improves more $\Rightarrow$ bounds improve more



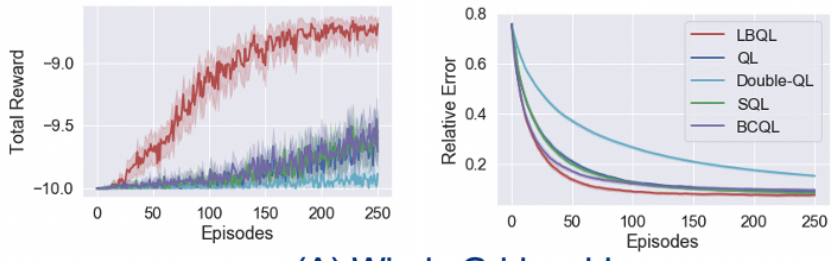
Behavior of LBQL (Pricing in Carsharing)

Q improves $\Rightarrow$ bounds improve $\Rightarrow$ Q improves more $\Rightarrow$ bounds improve more

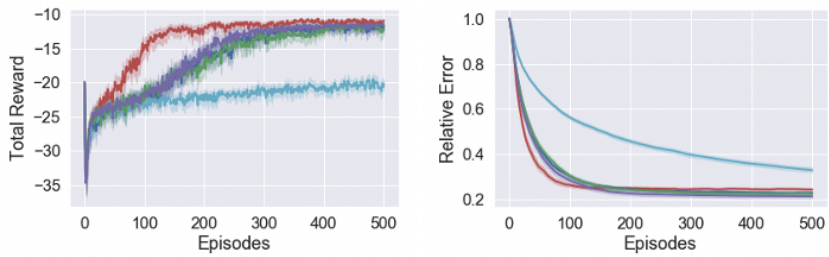


Comparison to Other Algorithms

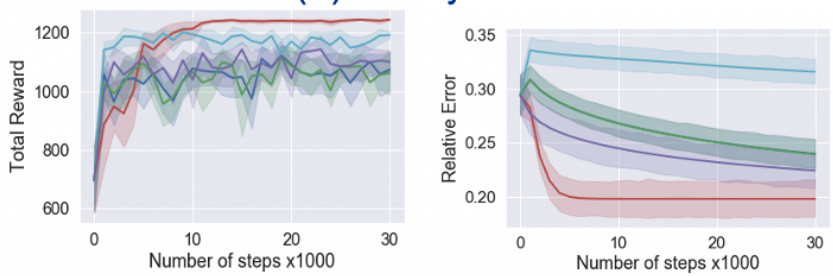
Relative Error = $\|V - V^*\|_2 / \|V^*\|_2$



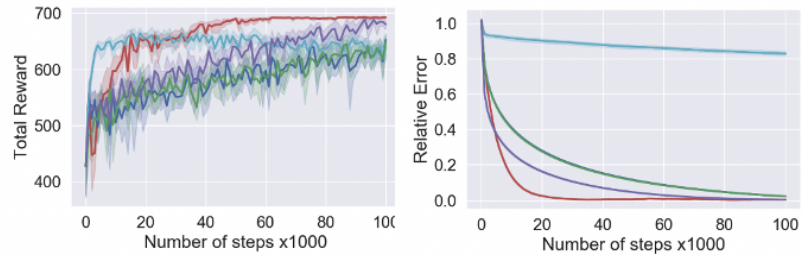
(A) Windy Gridworld



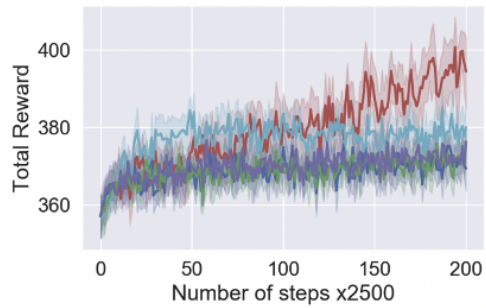
(B) Stormy Gridworld



(C) Repositioning for car-sharing (2 stations)



(D) Pricing for car-sharing (2 stations)



(E) Pricing for car-sharing (4 stations)

When to use LQBL?

- Many sampled DPs need to be solved for this algorithm to work.
  - *If experience is cheap, then no need for LBQL – simply run Q-learning. But if samples are expensive to obtain, solving the DPs may be reasonable.*
  - *Each DP "solve" actually provides DP solutions to the entire state space – one can actually do updates to the bounds everywhere.*
- Need to know the transition function f to solve DPs.
  - *Because the transition function f needs to be known, LBQL is not strictly a model-free RL algorithm. Knowledge of f, however, is common in OR/OM applications (e.g., inventory, RM, scheduling, routing), as discussed previously.*

Thank you for the invitation! Any questions?

Please feel free to email me at drjiang@pitt.edu and we can chat anytime.