Reinforcement Learning

—

Lecture 2: Dynamic Programming

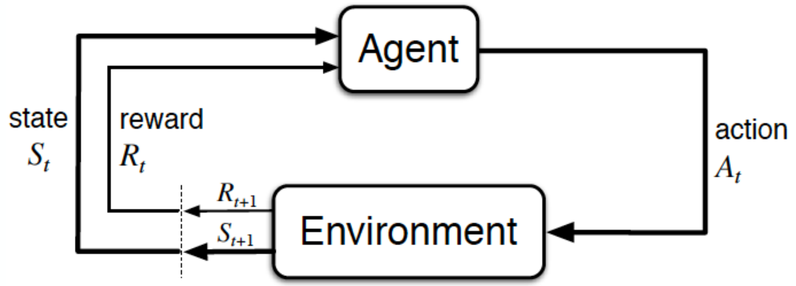
Christopher Mutschler

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

Recap
Markov Decision Processes

- Agent learns by interacting with an environment over many time-steps:
- Markov Decision Process (MDP) is a tool to formulate RL problems
  - Description of an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

- At each step t, the agent:
  - is at state S_t,
  - performs action A_t,
  - receives reward R_t.

- At each step t, the environment:
  - receives action A_t from the agent,
  - provides reward R_t,
  - moves at state S_{t+1},
  - increments time $t \leftarrow t + 1$.

Note:
If the interaction does stop at some point in time (T) then we have an *episodic RL problem.*

Recap

Markov Decision Processes: about the policy π

- Expected long-term value of state s:

$$v(s) = \mathbb{E}(\mathrm{G}) = \mathbb{E}(R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \ \ldots + \gamma^t R_t)$$

- **Goal: maximize the expected return $\mathbb{E}(\mathrm{G})$.**

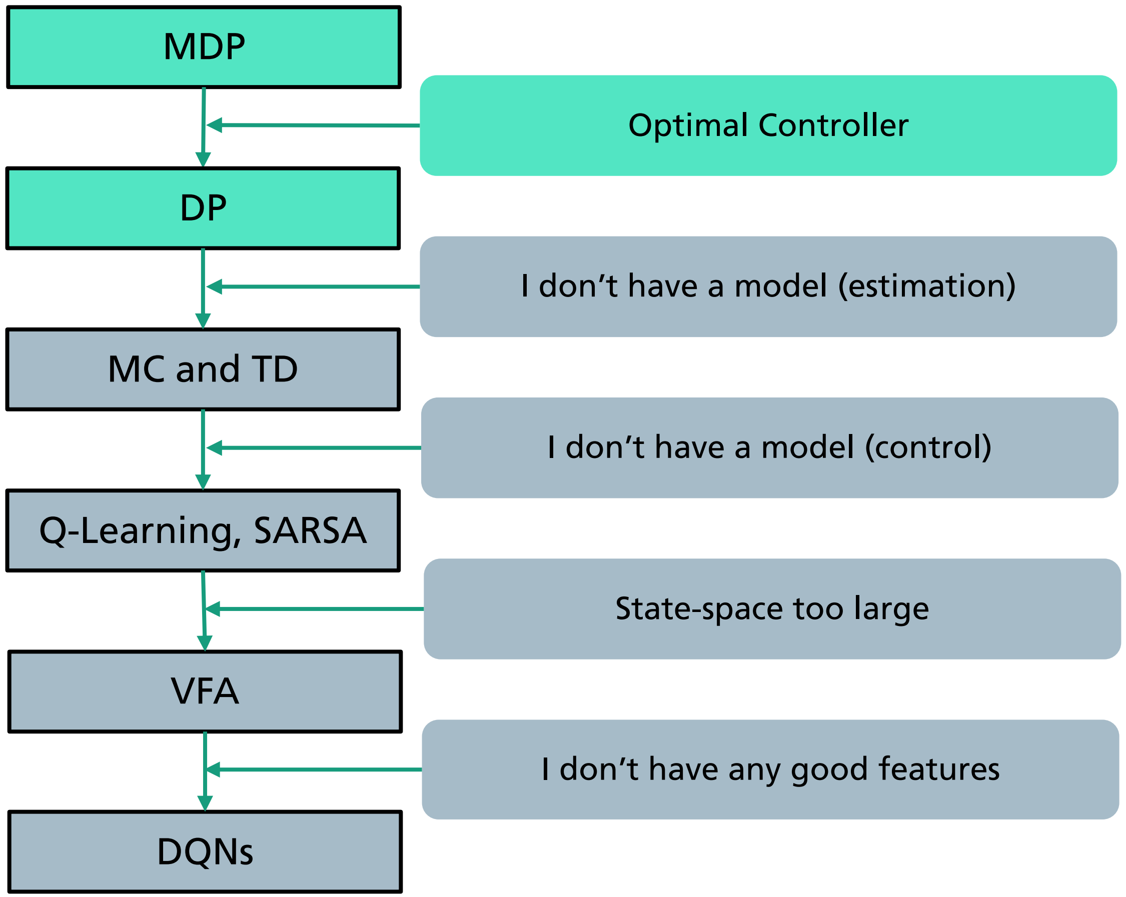
- We need a controller that helps us select the actions to maximize $\mathbb{E}(G)$!
- A policy π represents this controller:
  - π determines the agent's behavior, i.e., its way of acting
  - π is a mapping from state space $\mathcal{S}$ to action space $\mathcal{A}$

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

  - Two types of policies:
    - Deterministic policy: $a = \pi(s)$.
    - Stochastic policy: $\pi(a \mid s) = \mathbb{P}[\, A_t = a \mid S_t = s \,]$.

- **New goal: find a policy that maximizes the expected return!**

Overview

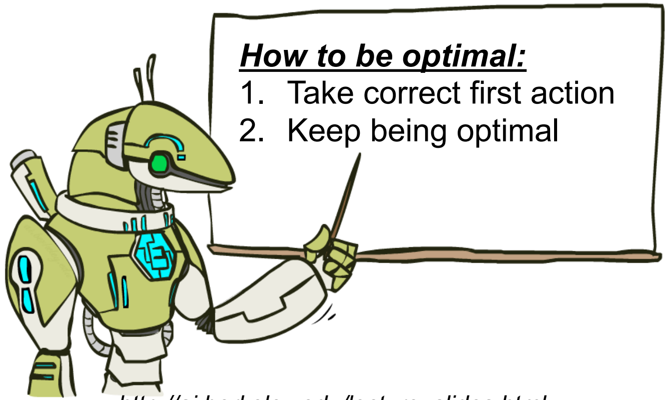

MDP
Optimal Controller
DP
I don’t have a model (estimation)
MC and TD
I don’t have a model (control)
Q-Learning, SARSA
State-space too large
VFA
I don’t have any good features
DQNs

Dynamic Programming

- How do we find **optimal** controllers for given (known) MDPs?
- Bellman equation & Bellman's principle of optimality

Principle of Optimality:
„An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.“

(see Bellman, 1957, Chap. III.3.)



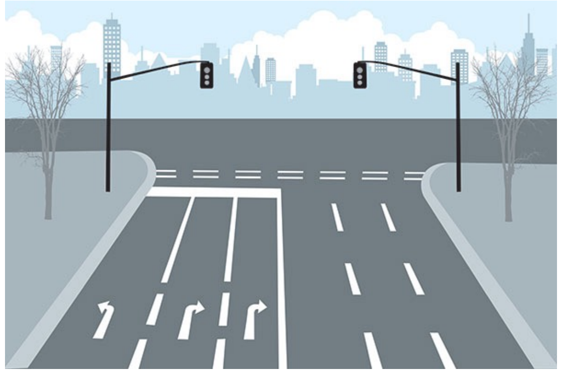
http://ai.berkeley.edu/lecture_slides.html

Dynamic Programming

Example

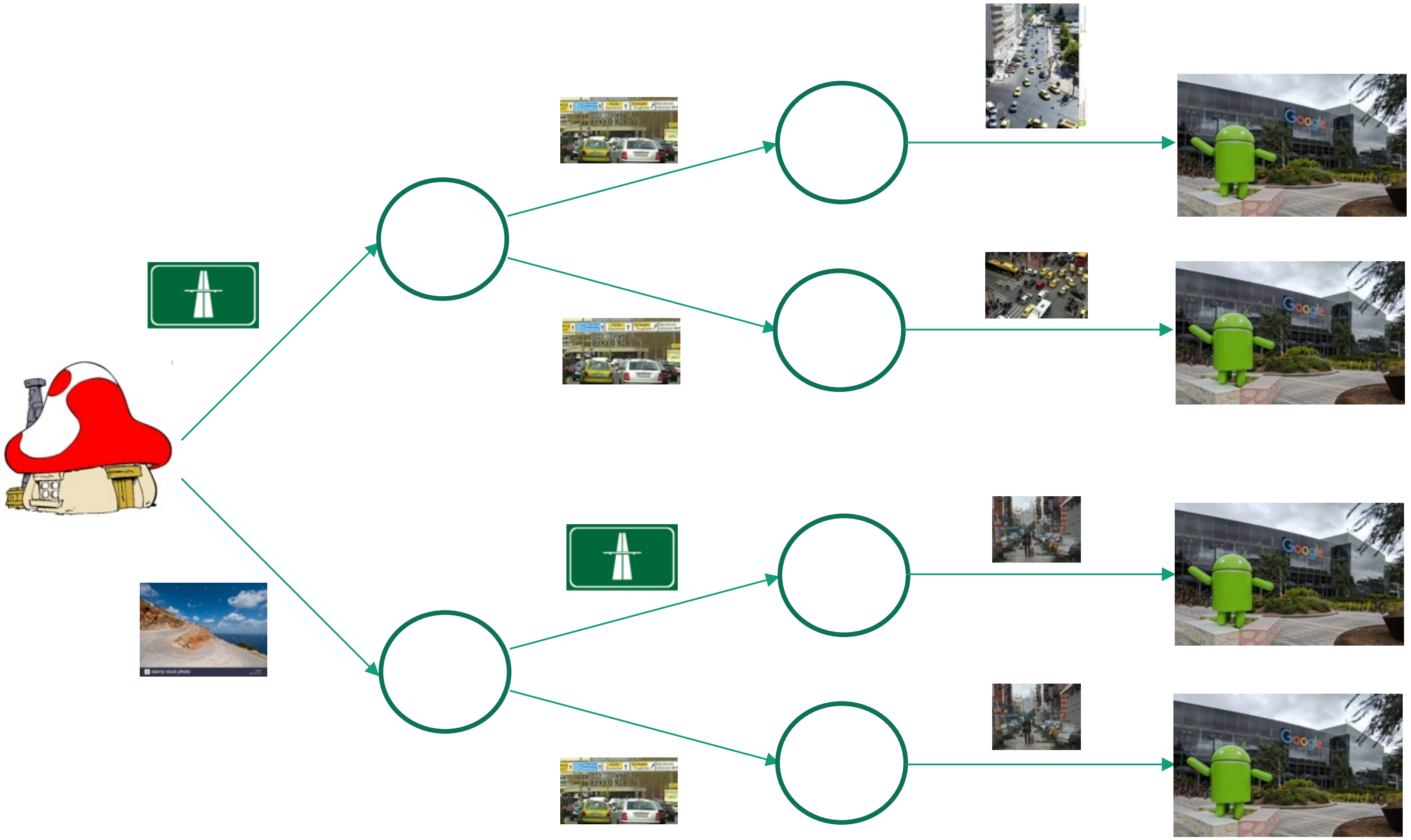
Example # 1 (Simplistic)

- We need to go from our house to our work as fast as possible
- Actions: Left, Right
- Different actions lead to different road segments (e.g., highway, country road, etc.)
- Reward: $\frac{x}{t}$, where t is the time needed for each road segment and x some scaler

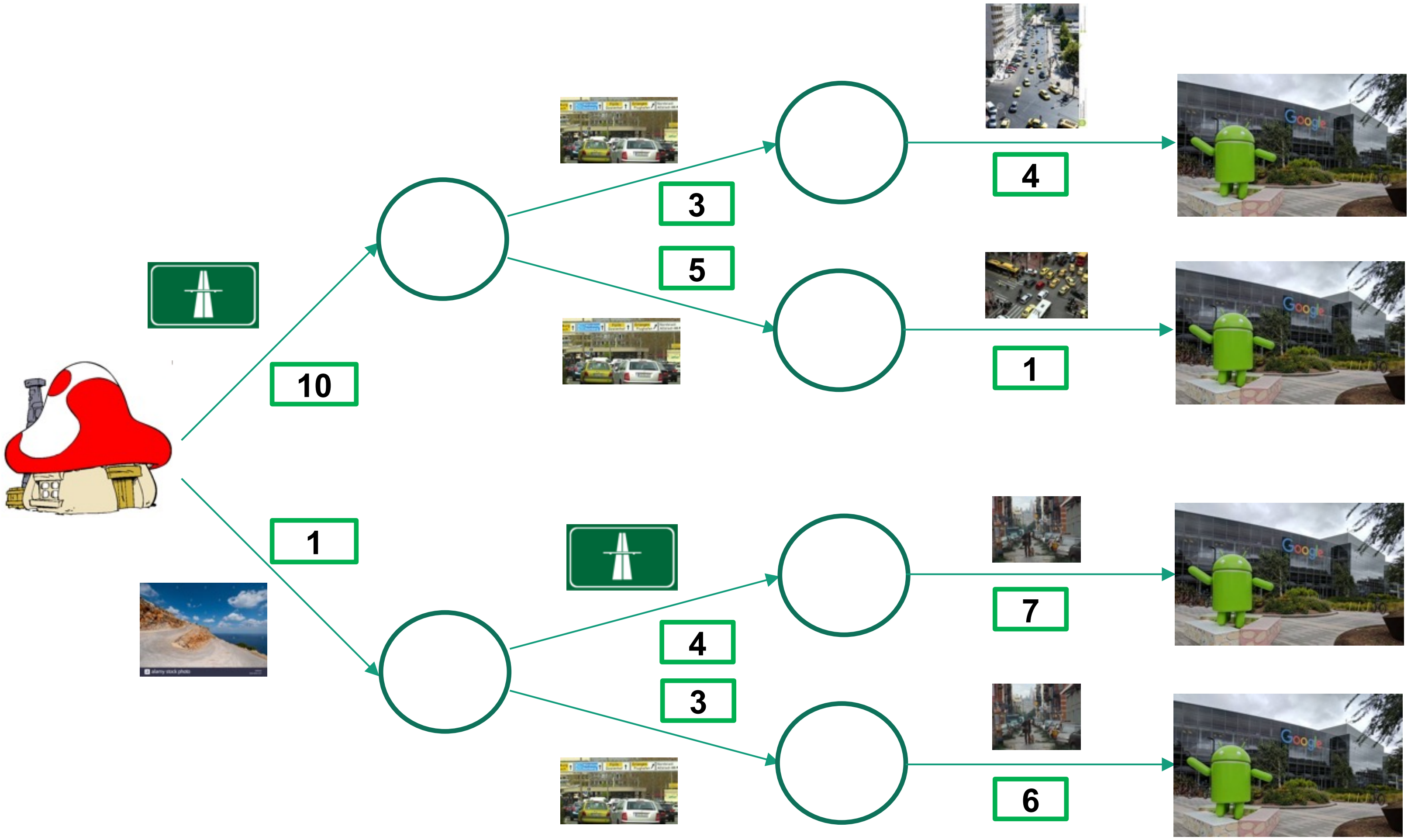


Fraunhofer IIS

Dynamic Programming

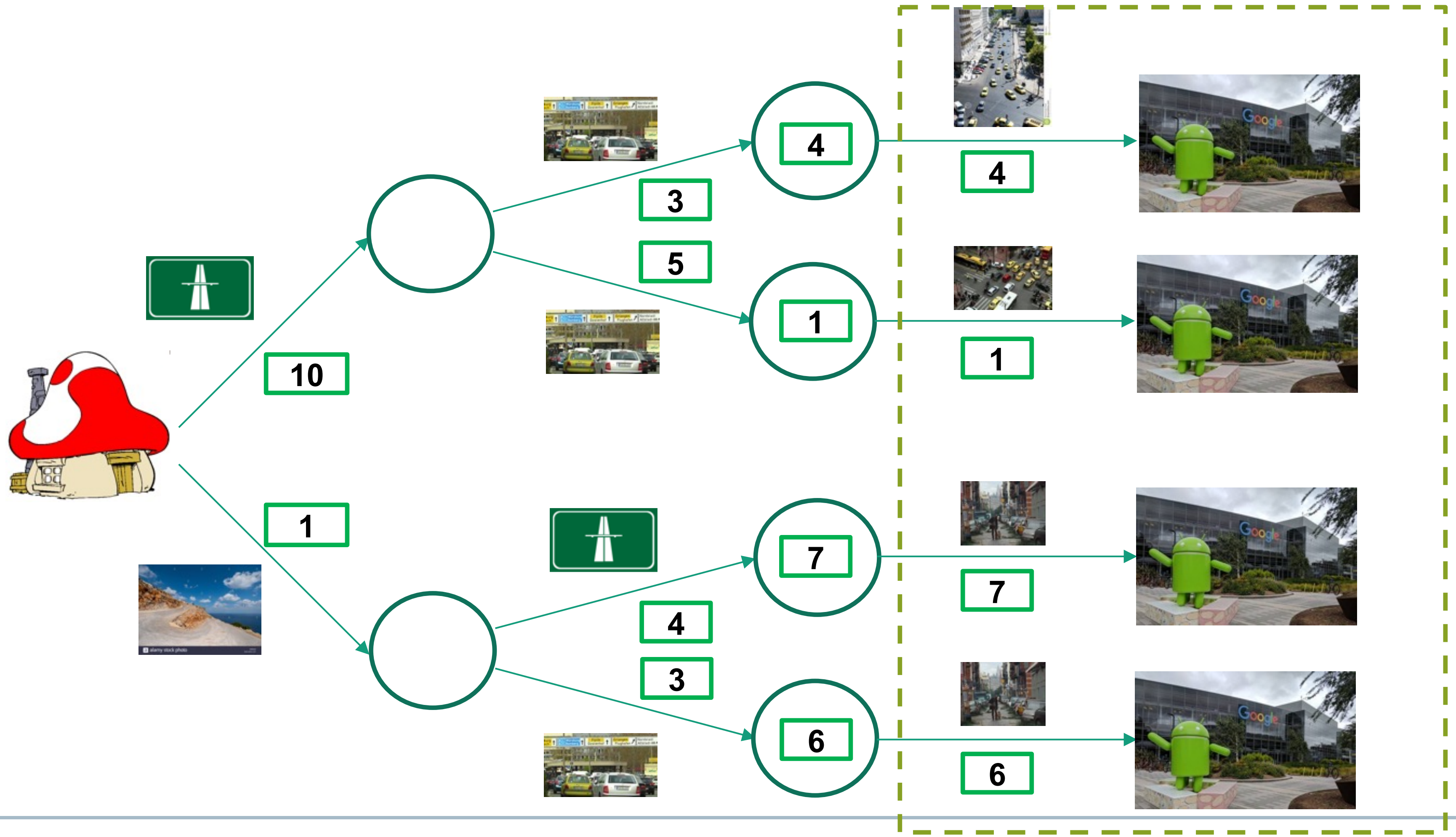


Dynamic Programming

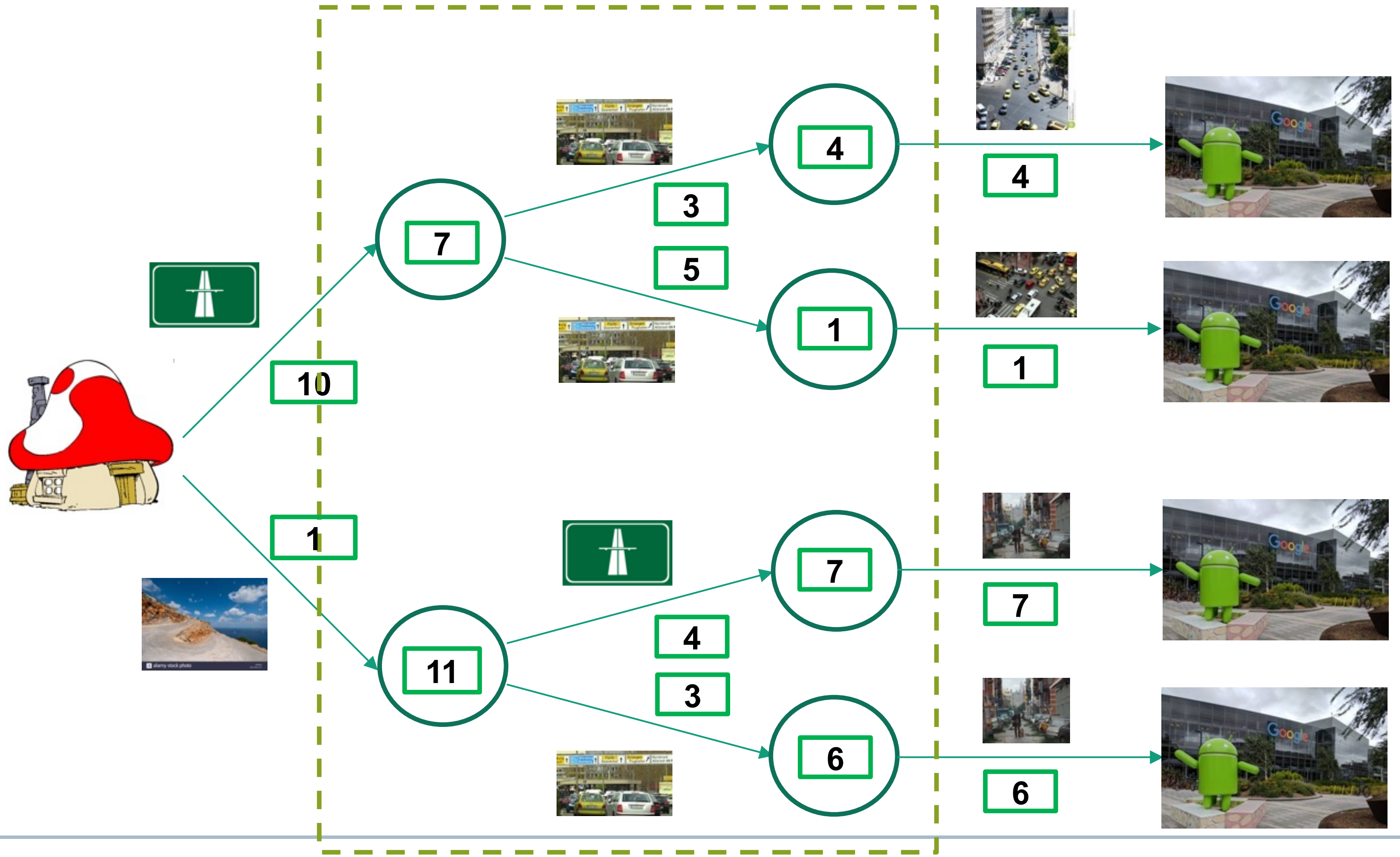


Fraunhofer IIS

Dynamic Programming

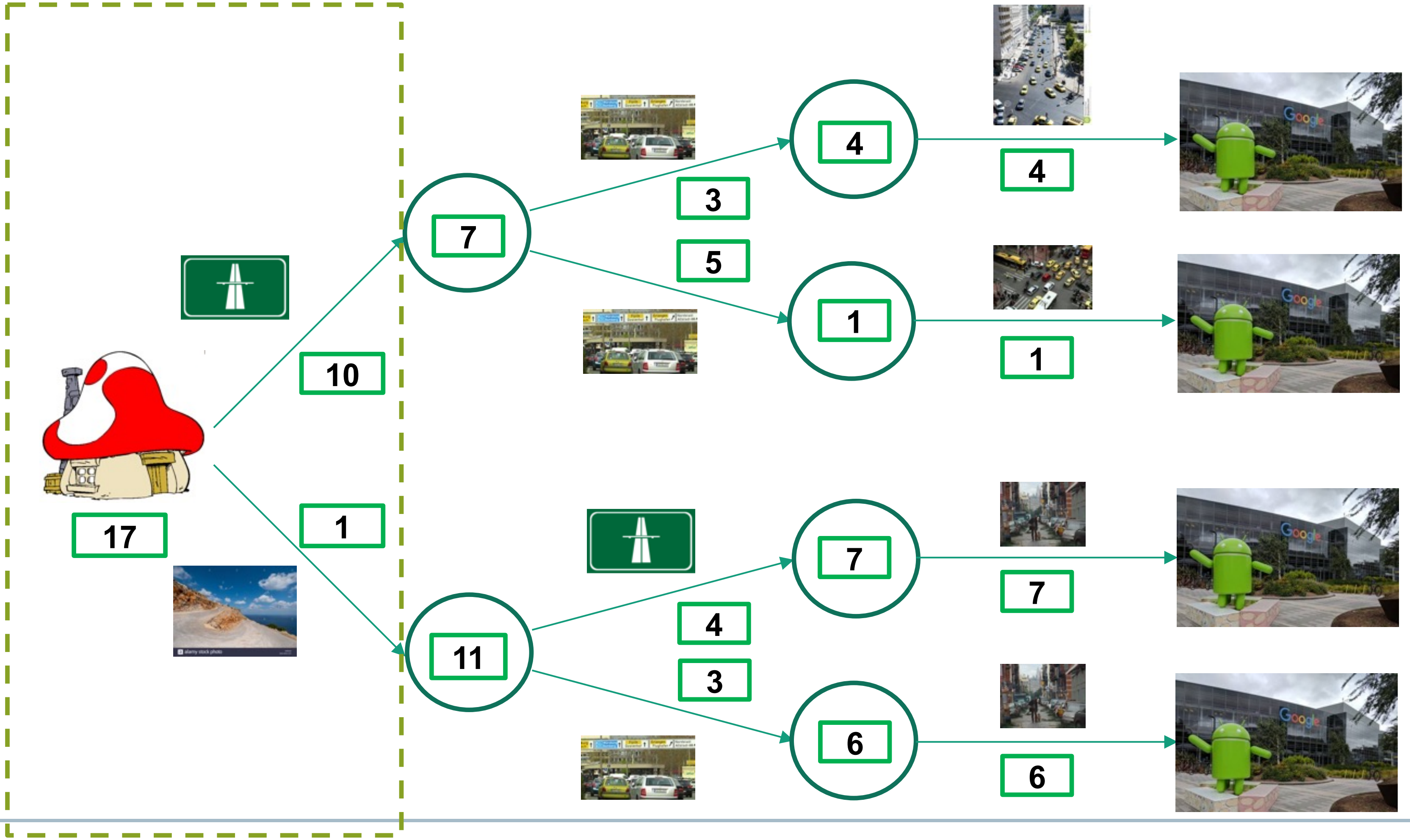


Dynamic Programming



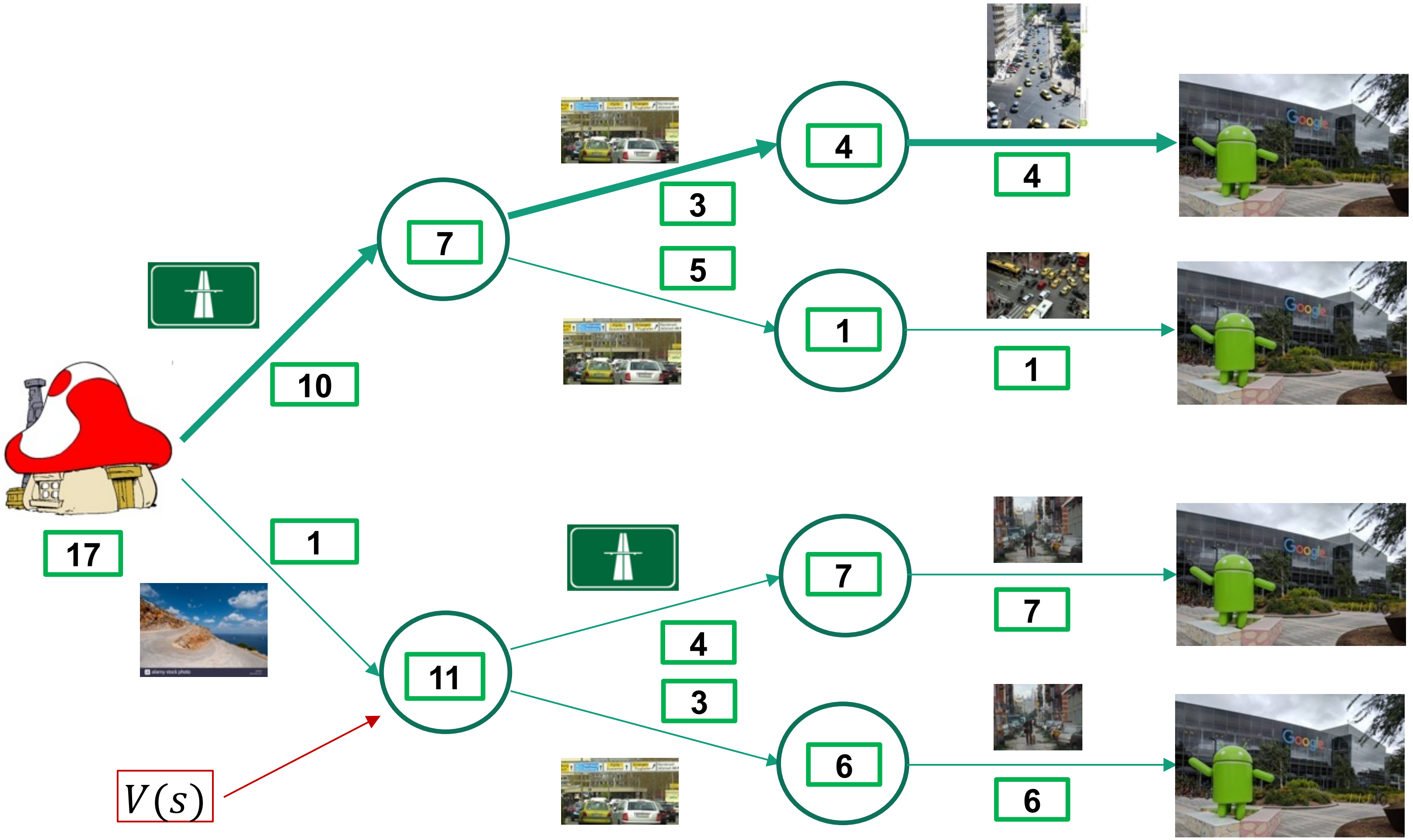
Fraunhofer IIS

Dynamic Programming



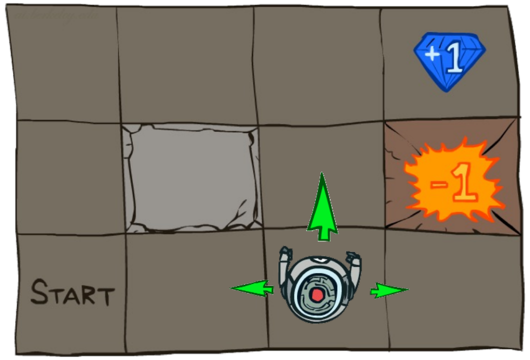
Fraunhofer
IIS

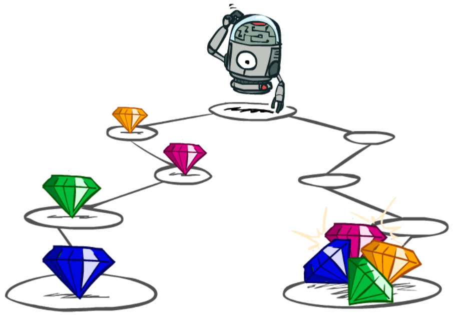
Dynamic Programming



Dynamic Programming

Example #2 (Simplistic)





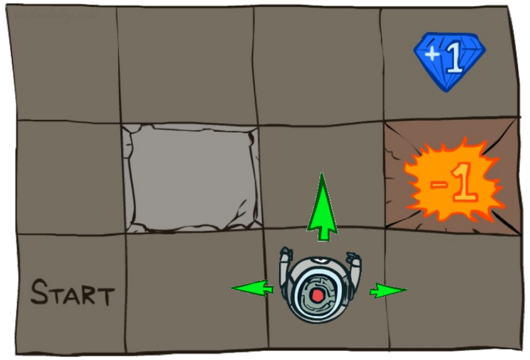
http://ai.berkeley.edu/lecture_slides.html

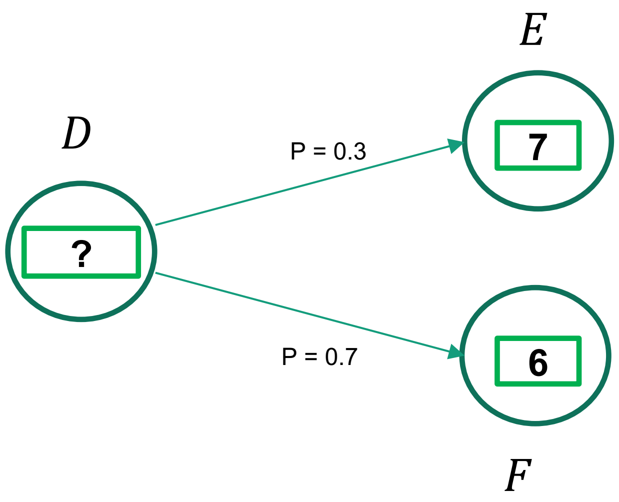
Fraunhofer IIS

Dynamic Programming

Example #2 (Simplistic)

$$P(s' = E, | a = 0, s = \mathrm{D}) = 0.3$$
$$P(s' = F, | a = 0, s = \mathrm{D}) = 0.7$$

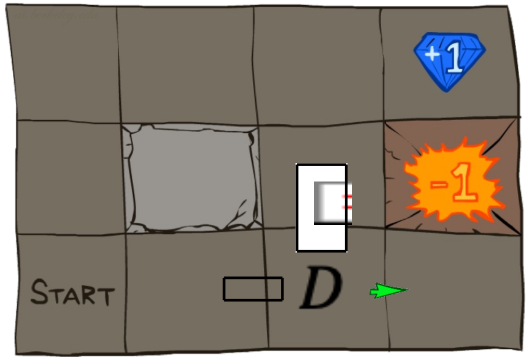


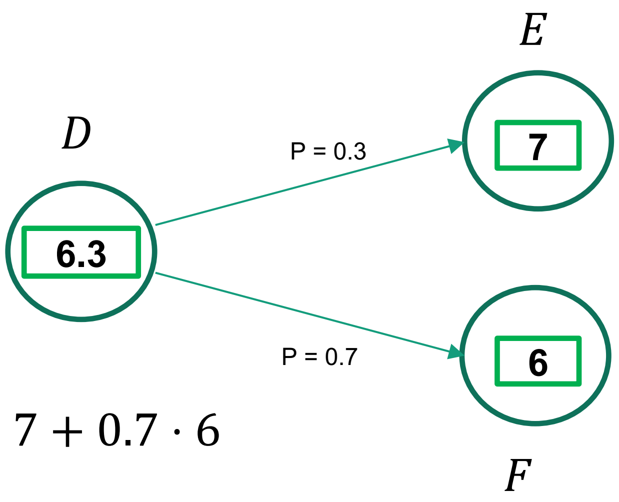


Dynamic Programming

Example #2 (Simplistic)

$$P(s' = E, | a = 0, s = \mathrm{D}) = 0.3$$
$$P(s' = F, | a = 0, s = \mathrm{D}) = 0.7$$





$$0.3 \cdot 7 + 0.7 \cdot 6$$

Dynamic Programming

- How do we find **optimal** controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - state-value function V for policy π

$$s_0 \xrightarrow{\pi(s_0), r(s, \pi(s_0))} s_1 \xrightarrow{\pi(s_1), r_1} s_2 \xrightarrow{\pi(s_2), r_2} s_3 \dots \; s_{h-1} \xrightarrow{\pi(s_{h-1}), r_{h-1}} s_h$$

$$V^\pi(s) \triangleq Q^\pi\big(s, \pi(s)\big) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$$

  - state-action-value function Q for policy π

$$s_0 \xrightarrow{a, r_0} s_1 \xrightarrow{\pi(S_1), r_1} s_2 \xrightarrow{\pi(s_2), r_2} s_3 \dots \; s_{h-1} \xrightarrow{\pi(s_{h-1}), r_{h-1}} s_h$$

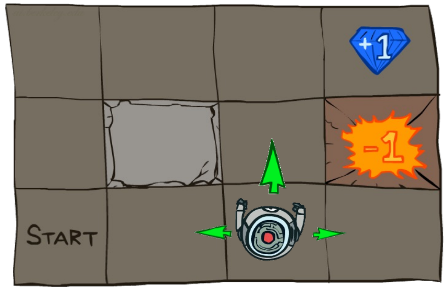
$$Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right]$$

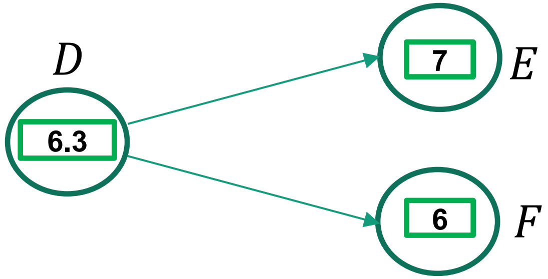
Dynamic Programming

- How do we find **optimal** controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - Bellman Equation for V, given policy π

$$s_0 \xrightarrow{\pi(s_0), r(s,\pi(s_0))} s_1 \xrightarrow{\pi(s_1), r_1} s_2 \xrightarrow{\pi(s_2), r_2} s_3 \dots\ s_{h-1} \xrightarrow{\pi(s_{h-1}), r_{h-1}} s_h$$

$$V^\pi(s) = \underbrace{r(s,\pi(s))}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,\pi(s))\, V^\pi(s')}_{\text{subsequent steps}}$$





Dynamic Programming

- How do we find **optimal** controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - Bellman Equation for Q, given policy π

$$s_0 \xrightarrow{a, r(s,a)} s_1 \xrightarrow{\pi(s_1), r_1} s_2 \xrightarrow{\pi(s_2), r_2} s_3 \cdots s_{h-1} \xrightarrow{\pi(s_{h-1}), r_{h-1}} s_h$$

$$Q^\pi(s,a) = \underbrace{r(s,a)}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, Q^\pi(s', \pi(s'))}_{\text{subsequent steps}}$$

Dynamic Programming

- How do we find **optimal** controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - Bellman Optimality Equation for V

$$s_0 \xrightarrow{\pi^*(s_0),\, r(s_0,\pi(s_0))} s_1 \xrightarrow{\pi^*(s_1),\, r_1} s_2 \xrightarrow{\pi^*(s_2),\, r_2} s_3 \cdots s_{h-1} \xrightarrow{\pi^*(s_{h-1}),\, r_{h-1}} s_h$$

$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left\{ \underbrace{r(s,a)}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^{\pi^*}(s')}_{\text{subsequent steps}} \right\}$$

Dynamic Programming

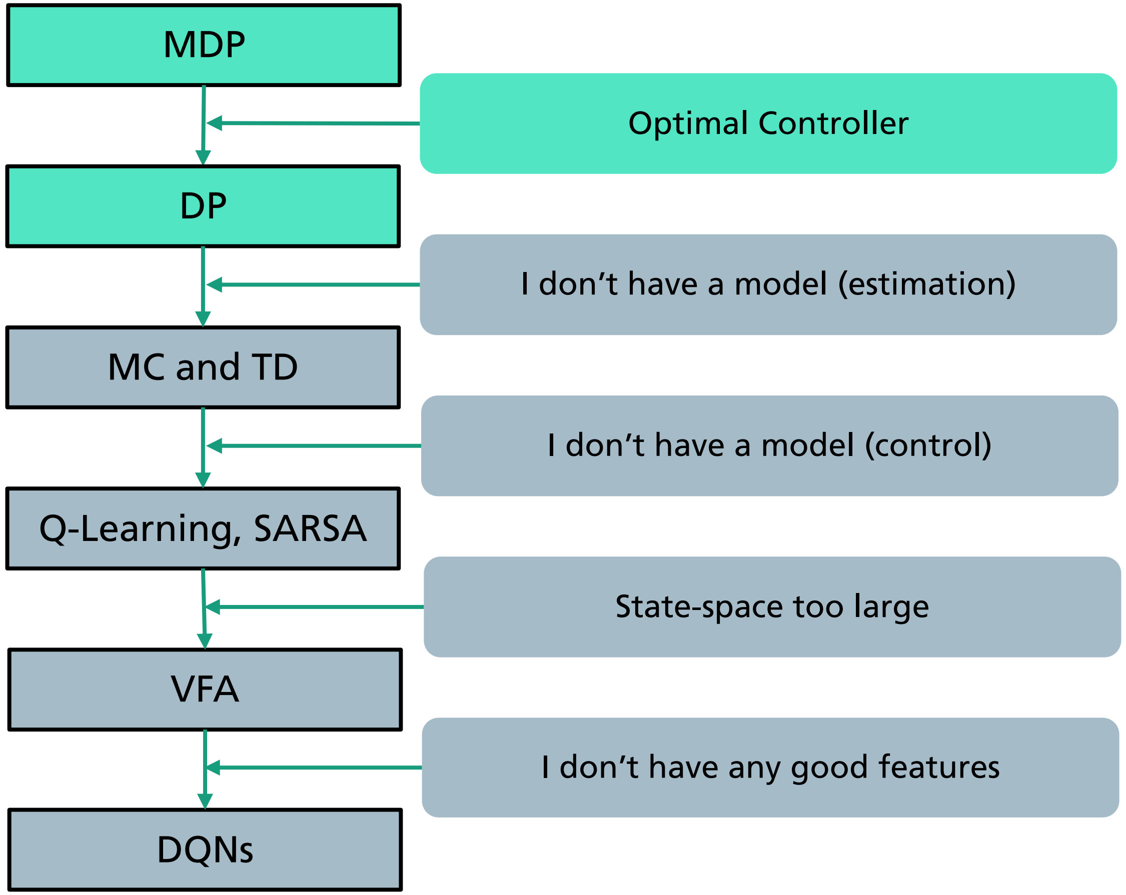
- How do we find **optimal** controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - Bellman Optimality Equation for Q

$$s \xrightarrow{a, r(s,a)} s_1 \xrightarrow{\pi^*(s_1), r_1} s_2 \xrightarrow{\pi^*(s_2), r_2} s_3 \cdots s_{h-1} \xrightarrow{\pi^*(s_{h-1}), r_{h-1}} s_h$$

$$Q^{\pi^*}(s,a) = \underbrace{r(s,a)}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^{\pi^*}(s',a')}_{\text{subsequent steps}}$$

Dynamic Programming

Value Iteration



Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

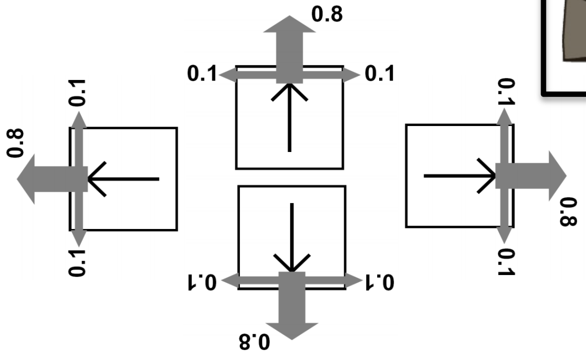
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

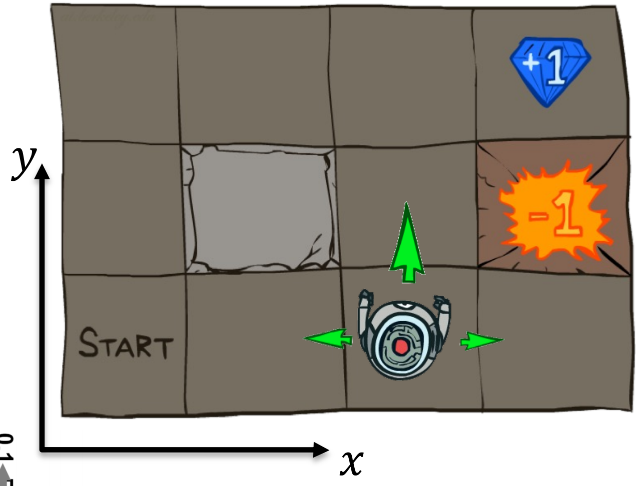
Dynamic Programming
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r \mid s, a)\,[r + \gamma V_{i-1}(s')]$$

- Value Iteration Example:
  - Noise = 0.2 (it is windy)
  - $\gamma = 0.9$
  - Living reward = 0.0
    (Transitioning from state to state)





http://ai.berkeley.edu/reinforcement.html

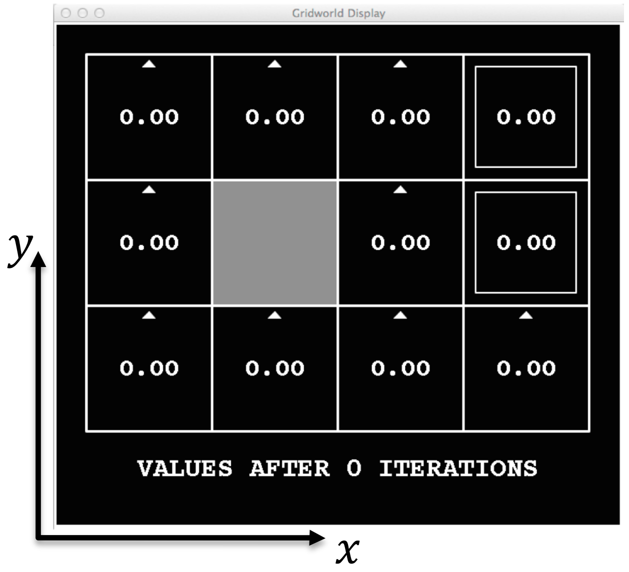
Dynamic Programming
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r | s, a) \left[r + \gamma V_{i-1}(s') \right]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$V_1(S^{3,2}) = 1.00$ (terminal state with reward 1.0)
$V_1(S^{2,2}) = 0.00$
$V_1(S^{1,2}) = 0.00$
$\ldots$
$V_1(S^{3,1}) = -1.00$ (terminal state with reward -1.0)



http://ai.berkeley.edu/reinforcement.html

Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

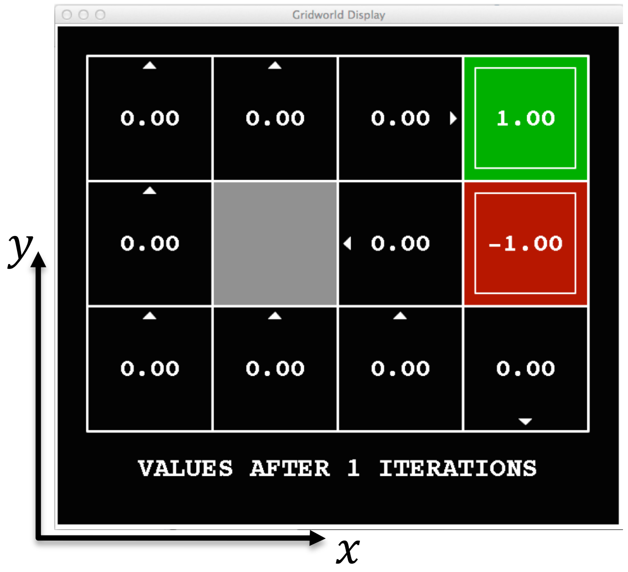
$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r \mid s, a)\,[r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$$V_2(S^{2,2}) = \max_{a \in \mathcal{A}} \begin{Bmatrix} a = R: 0.8 * [0.0 + 0.9 * 1.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] \\ a = L: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] \\ a = U: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 1.0] \\ a = D: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 1.0] + 0.1 * [0.0 + 0.9 * 0.0] \end{Bmatrix}$$

$$= \max_{a \in \mathcal{A}} \begin{Bmatrix} 0.8 * 0.9 * 1.0 + 0.1 = 0.72 \\ 0 \\ 0.1 * 0.9 * 1.0 = 0.09 \\ 0.1 * 0.9 * 1.0 = 0.09 \end{Bmatrix} = 0.72$$

$$V_2(S^{2,1}) = \cdots$$



http://ai.berkeley.edu/reinforcement.html

Fraunhofer IIS

Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

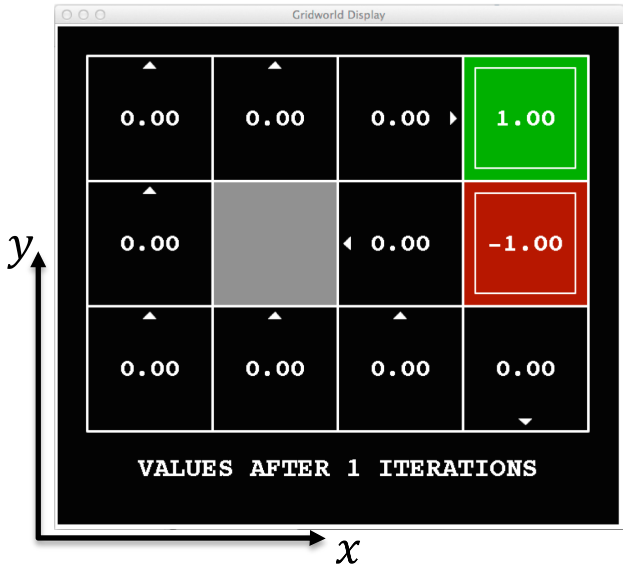
$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r | s, a) \left[r + \gamma V_{i-1}(s') \right]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$$V_2(S^{2,2}) = 0.72$$

$$V_2(S^{2,1}) = \max_{a \in \mathcal{A}} \begin{Bmatrix} a = R: 0.8 * [0.0 + 0.9 * -1.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] \\ a = L: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] \\ a = U: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * -1.0] \\ a = D: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * -1.0] + 0.1 * [0.0 + 0.9 * 0.0] \end{Bmatrix}$$

$$= \max_{a \in \mathcal{A}} \begin{Bmatrix} 0.8 * 0.9 * -1.0 + 0.1 = -0.72 \\ 0 \\ 0.1 * 0.9 * -1.0 = -0.09 \\ 0.1 * 0.9 * -1.0 = -0.09 \end{Bmatrix} = 0.00$$



http://ai.berkeley.edu/reinforcement.html

Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

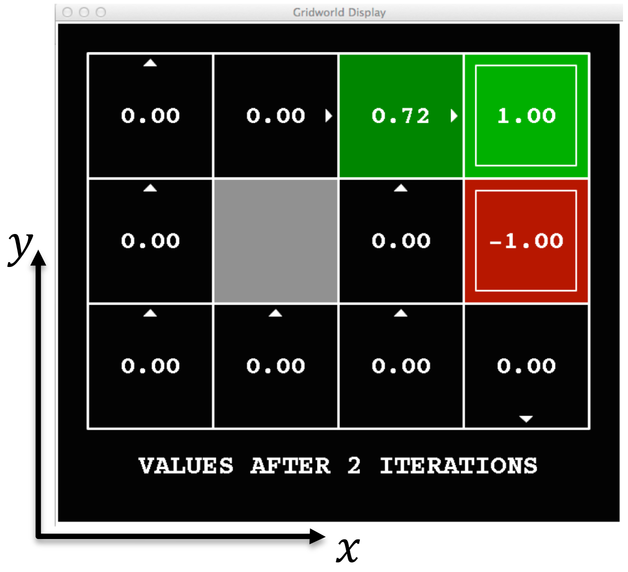
$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r \mid s, a)\,[r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$$V_3(S^{2,2}) = \max_{a \in \mathcal{A}} \begin{Bmatrix} a = R: 0.8 * [0.0 + 0.9 * 1.0] + 0.1 * [0.0 + 0.9 * 0.72] + 0.1 * [0.0 + 0.9 * 0.0] \\ a = L: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.72] \\ a = U: 0.8 * [0.0 + 0.9 * 0.72] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 1.0] \\ a = D: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 1.0] + 0.1 * [0.0 + 0.9 * 0.0] \end{Bmatrix}$$

$$= \max_{a \in \mathcal{A}} \begin{Bmatrix} 0.8 * 0.9 * 1.0 + 0.1 * 0.9 * 0.72 = 0.72 + 0.0648 \approx 0.78 \\ 0.1 * 0.9 * 0.72 = 0.0648 \approx 0.06 \\ 0.8 * 0.9 * 0.72 + 0.1 * 0.9 * 1.0 = 0.5184 + 0.09 \approx 0.61 \\ 0.1 * 0.9 * 1.0 = 0.09 \end{Bmatrix} = 0.78$$

$$V_3(S^{2,1}) = \cdots$$



http://ai.berkeley.edu/reinforcement.html

Fraunhofer IIS

Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

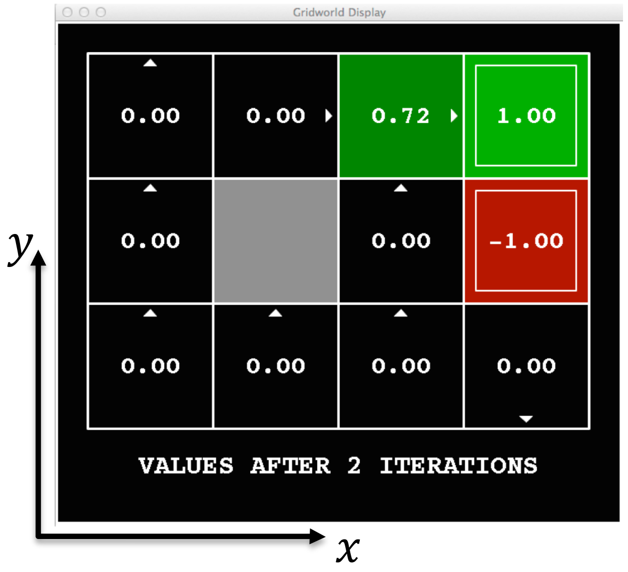
$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r \mid s, a)\,[r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$$V_3(S^{2,2}) = 0.78$$

$$V_3(S^{2,1}) = \max_{a \in \mathcal{A}} \begin{cases} a = R: 0.8 * [0.0 + 0.9 * -1.0] + 0.1 * [0.0 + 0.9 * 0.72] + 0.1 * [0.0 + 0.9 * 0.0] \\ a = L: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.72] \\ a = U: 0.8 * [0.0 + 0.9 * 0.72] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * -1.0] \\ a = D: 0.8 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * -1.0] + 0.1 * [0.0 + 0.9 * 0.0] \end{cases}$$

$$= \max_{a \in \mathcal{A}} \begin{cases} 0.8 * 0.9 * -1.0 + 0.1 * 0.9 * 0.72 = -0.72 + 0.0648 \approx -0.65 \\ 0.1 * 0.9 * 0.72 = 0.0648 \approx 0.06 \\ 0.8 * 0.9 * 0.72 - 0.1 * 0.9 * 1.0 = 0.5184 - 0.09 \approx 0.43 \\ 0.1 * 0.9 * -1.0 = -0.09 \end{cases} = 0.43$$



http://ai.berkeley.edu/reinforcement.html

Dynamic Programming

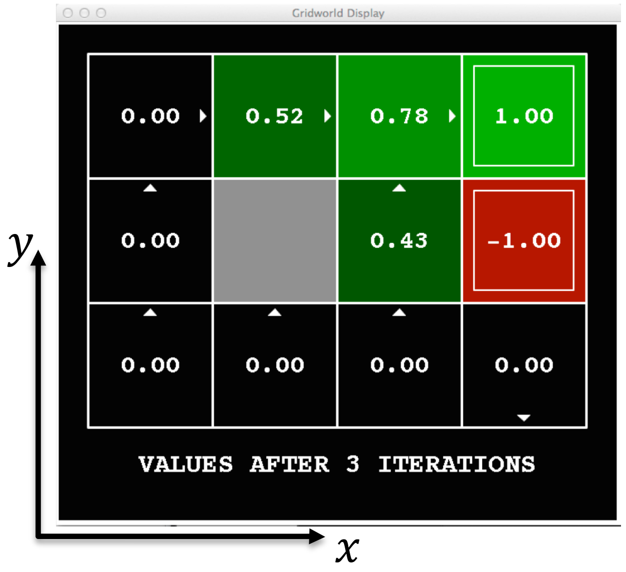
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r | s, a) [r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$V_4(S^{2,2}) = \cdots$
$V_4(S^{2,1}) = \cdots$



http://ai.berkeley.edu/reinforcement.html

Dynamic Programming

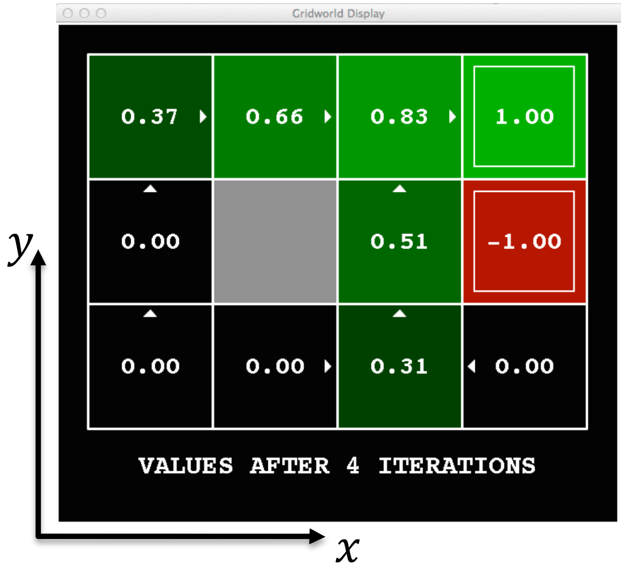
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r | s, a) [r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$V_5(S^{2,2}) = \cdots$
$V_5(S^{2,1}) = \cdots$



http://ai.berkeley.edu/reinforcement.html

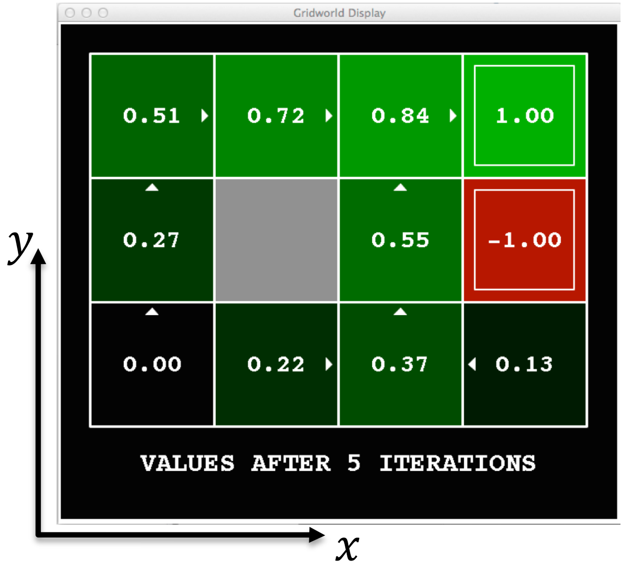
Dynamic Programming
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r \mid s, a)\,[r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$V_6(S^{2,2}) = \cdots$
$V_6(S^{2,1}) = \cdots$



http://ai.berkeley.edu/reinforcement.html

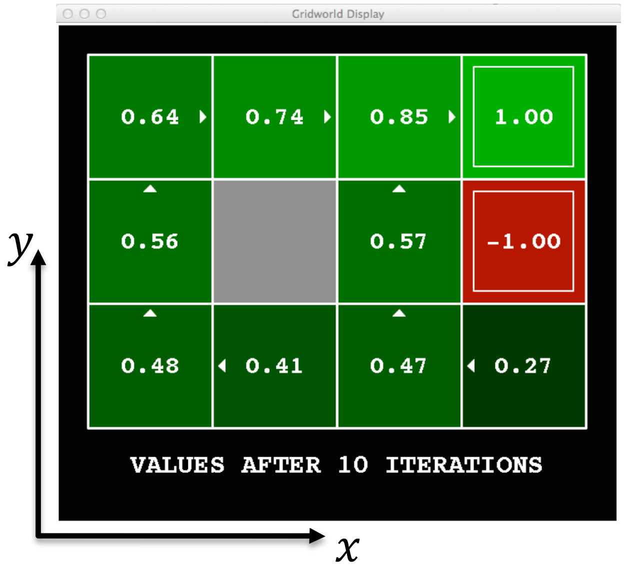
Dynamic Programming
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r \mid s, a) \left[r + \gamma V_{i-1}(s') \right]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$V_{11}(S^{2,2}) = \cdots$
$V_{11}(S^{2,1}) = \cdots$



http://ai.berkeley.edu/reinforcement.html

Dynamic Programming

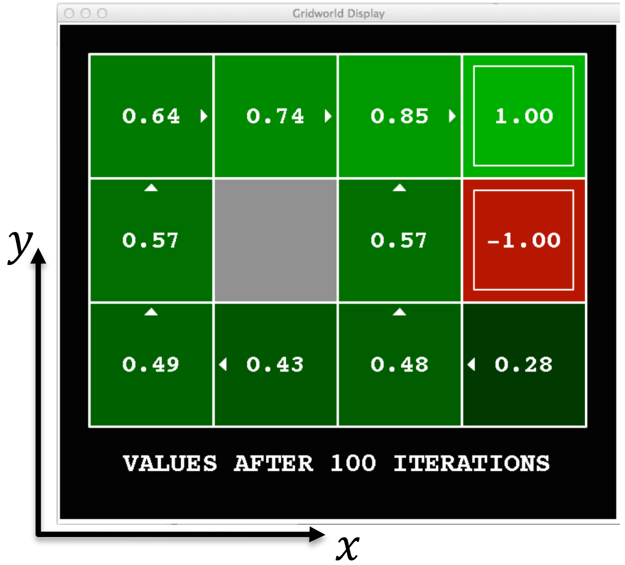
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r \mid s, a)\,[r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$V_{101}(S^{2,2}) = \cdots$
$V_{101}(S^{2,1}) = \cdots$



http://ai.berkeley.edu/reinforcement.html

Fraunhofer IIS

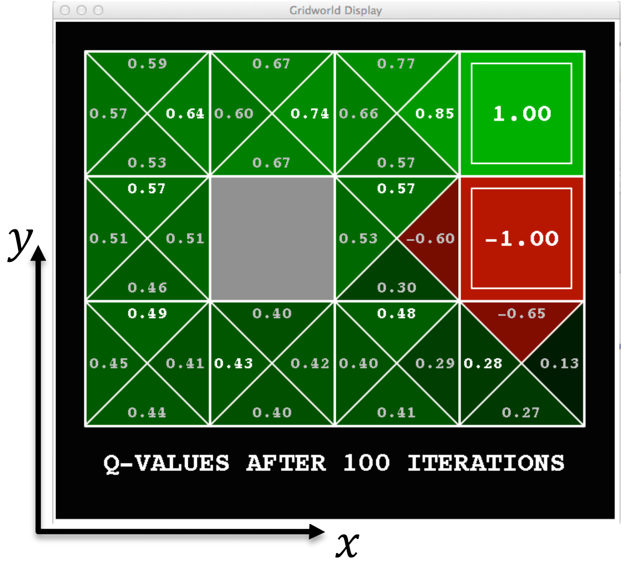
Dynamic Programming
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r | s, a) \, [r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = 0.0$

$V_{101}(S^{2,2}) = \cdots$
$V_{101}(S^{2,1}) = \cdots$



http://ai.berkeley.edu/reinforcement.html

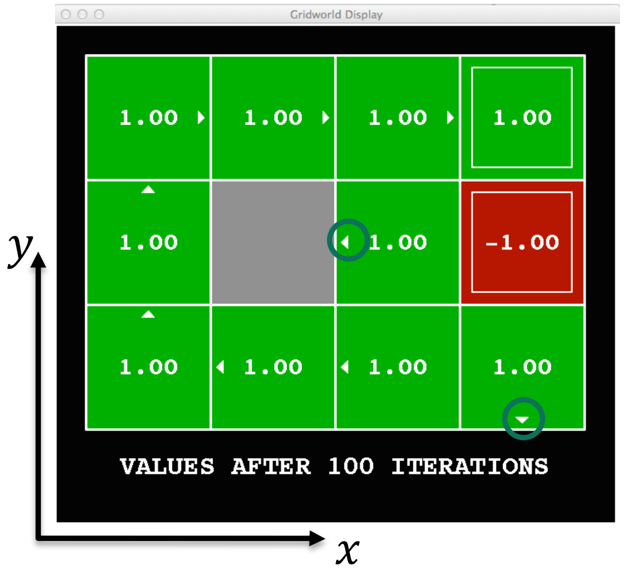
Fraunhofer IIS

Dynamic Programming
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S},\, r} \mathcal{P}(s', r \mid s, a)\,[r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\boldsymbol{\gamma} = \mathbf{1}$, $r = 0.0$
- How would it look like?



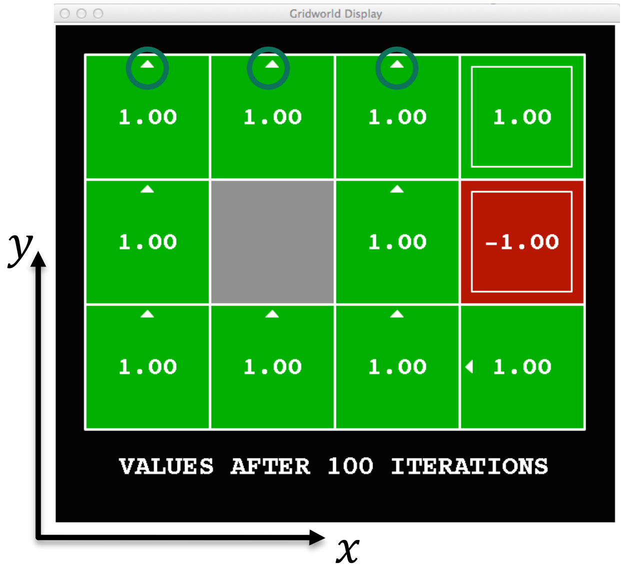
http://ai.berkeley.edu/reinforcement.html

Dynamic Programming
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r | s, a) \, [r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.0$, $\boldsymbol{\gamma} = \mathbf{1}$, $r = 0.0$
- How would it look like?



http://ai.berkeley.edu/reinforcement.html

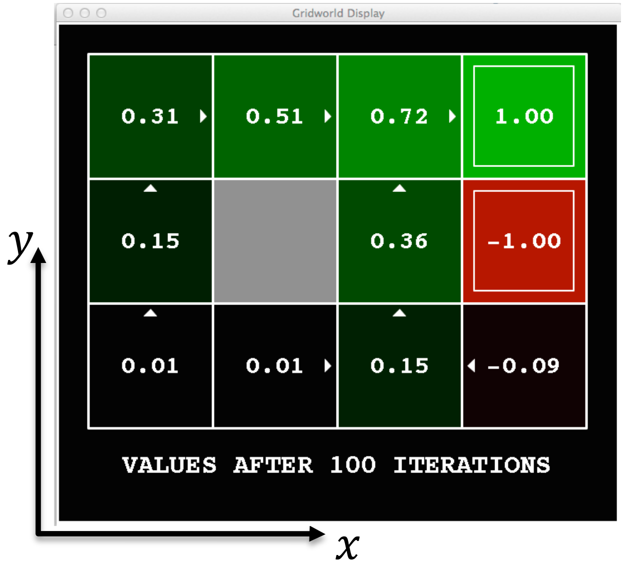
Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r \mid s, a)\,[r + \gamma V_{i-1}(s')]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $\boldsymbol{r} = \mathbf{-0.1}$
- How would it look like?



http://ai.berkeley.edu/reinforcement.html

Fraunhofer IIS

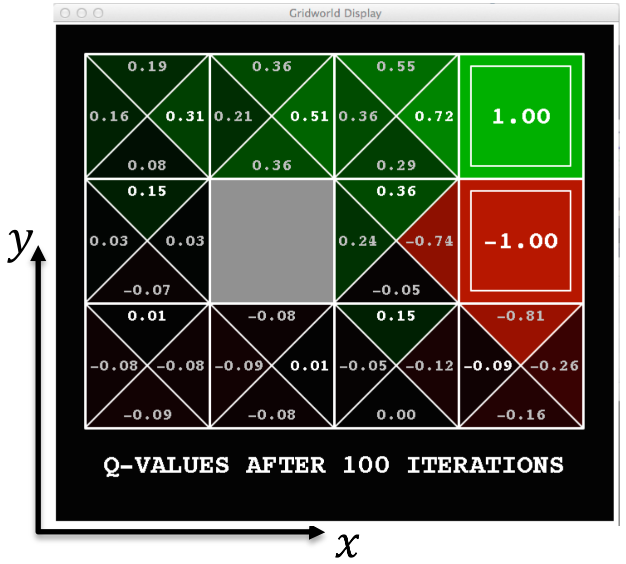
Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Optimal Solver #1: Value Iteration (convergence guaranteed)

$$V_i(S^{x,y}) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}, r} \mathcal{P}(s', r | s, a) \left[r + \gamma V_{i-1}(s') \right]$$

- Value Iteration Example: noise $= 0.2$, $\gamma = 0.9$, $r = -0.1$
- How would it look like?



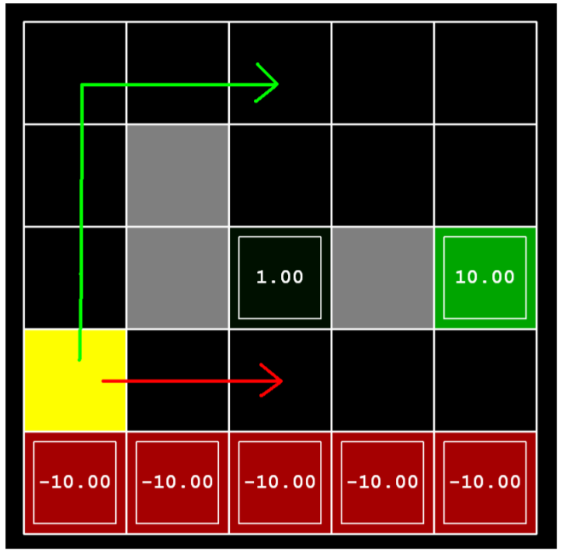
http://ai.berkeley.edu/reinforcement.html

Fraunhofer IIS

Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Important: effect of environment noise and γ



http://ai.berkeley.edu/reinforcement.html

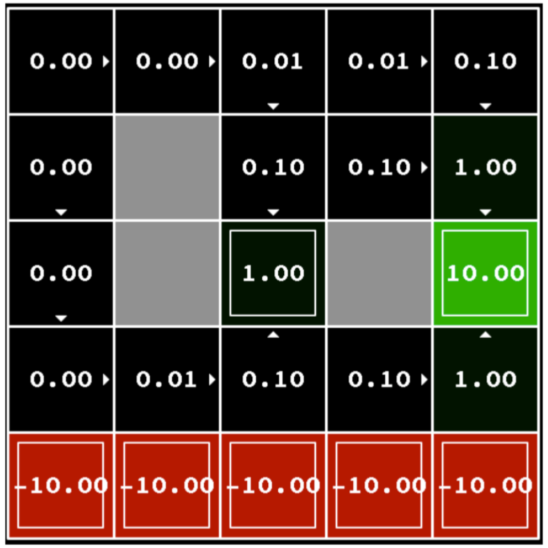
Goals:
- Close exit (Reward +1.0)
- Distant exit (Reward +10.0)

Avoid:
- Cliff on bottom (Reward -10.0)

Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Important: effect of environment noise and γ



http://ai.berkeley.edu/reinforcement.html

Solution for:
- $\gamma = 0.1$
- Noise = 0.0

Behavior:
- Prefers close exit
- Avoids cliff: No

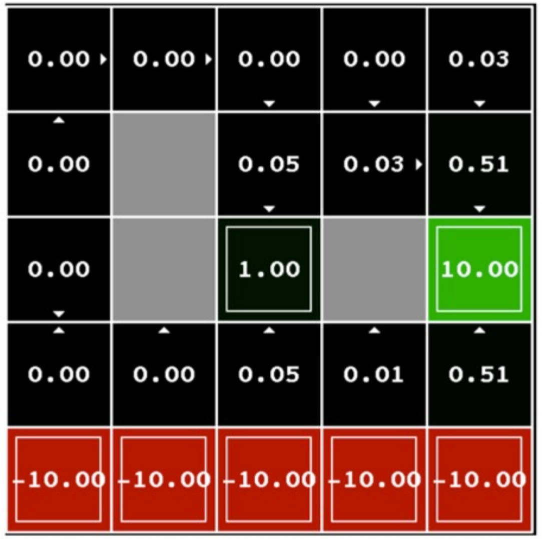
Why?
- Since noise = 0.0 there is no risk
- $\gamma = 0.1$ forces early termination

Fraunhofer IIS

Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Important: effect of environment noise and γ



http://ai.berkeley.edu/reinforcement.html

Solution for:

- $\gamma = 0.1$
- Noise = 0.5

Behavior:

- Prefers close exit
- Avoids cliff: Yes

Why?

- Since noise = 0.5 there is high risk
- $\gamma = 0.1$ forces early termination

Dynamic Programming

Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Important: effect of environment noise and γ

http://ai.berkeley.edu/reinforcement.html

Solution for:
- $\gamma = 0.99$
- Noise = 0.0

Behavior:
- Prefers distant exit
- Avoids cliff: No

Why?
- Since noise = 0.0 there is no risk
- $\gamma = 0.99$ allows for distant exit

Dynamic Programming
Value Iteration

- How do we find optimal controllers for given (known) MDPs?
- Important: effect of environment noise and γ

http://ai.berkeley.edu/reinforcement.html

Solution for:
- $\gamma = 0.99$
- Noise = 0.5

Behavior:
- Prefers distant exit
- Avoids cliff: Yes

Why?
- Since noise = 0.5 there is high risk
- $\gamma = 0.99$ allows for distant exit

Dynamic Programming

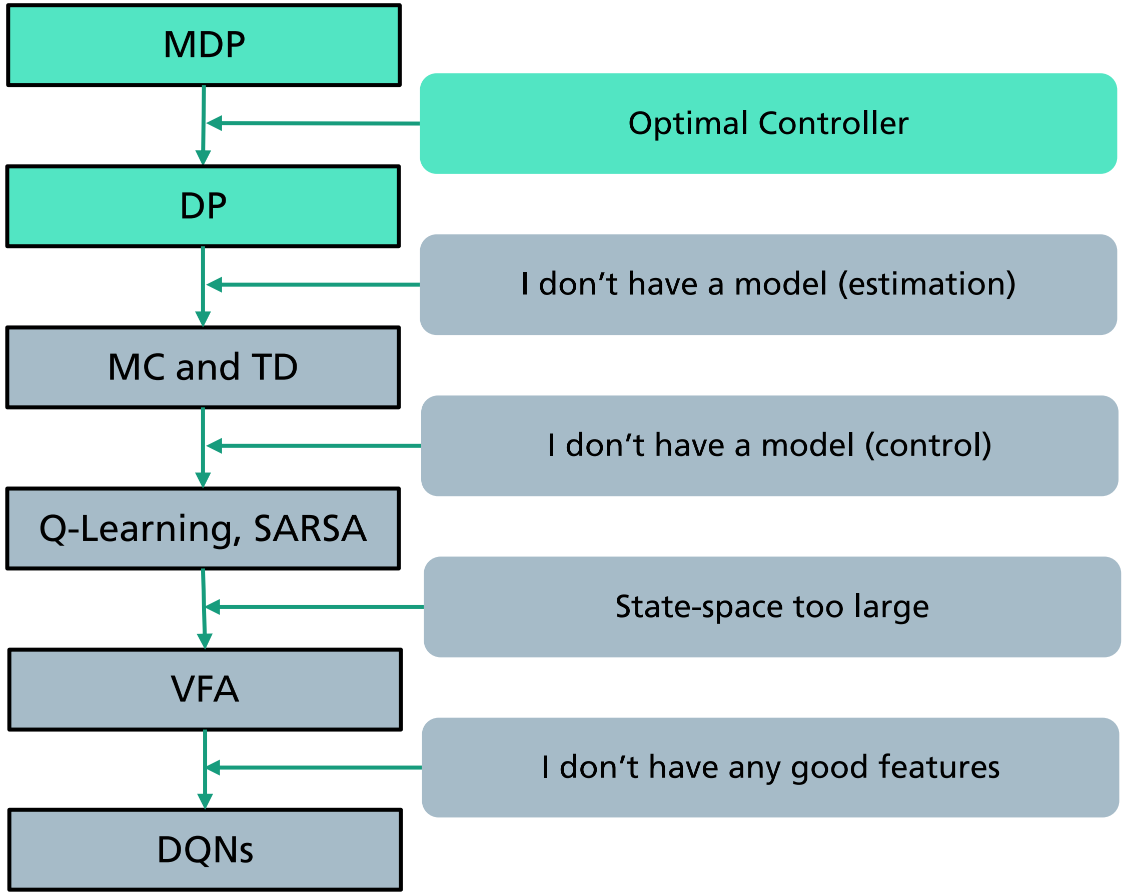
Value Iteration

Hands-On:

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

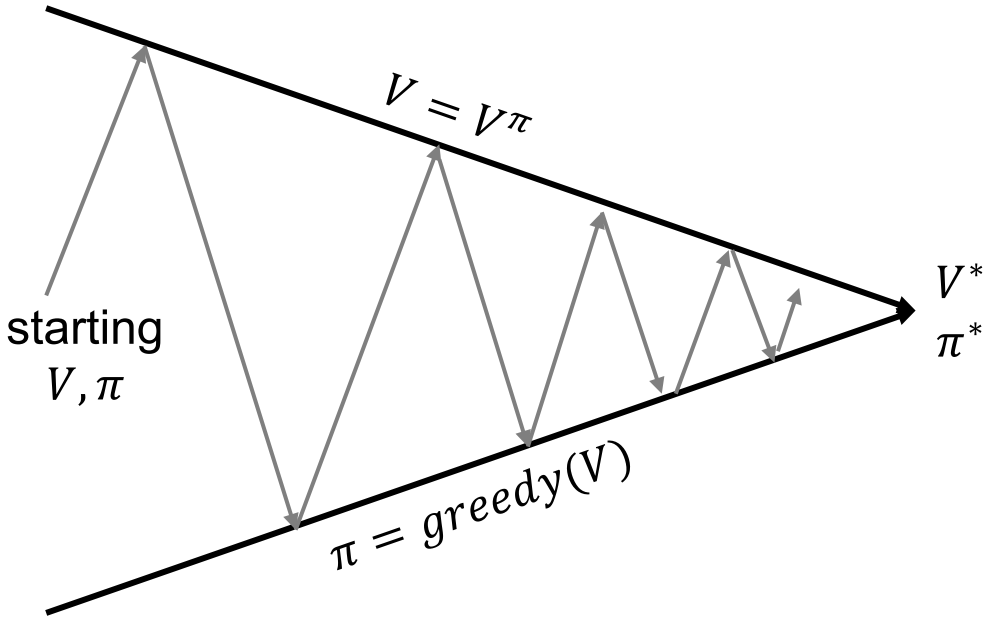
Dynamic Programming

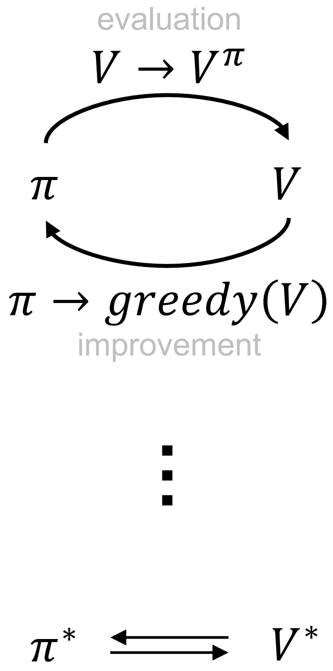
Policy Iteration



Dynamic Programming
Generalized Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?





Dynamic Programming

Generalized Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration
  - Given a policy π:

1. **Evaluate** policy π (Bellman Expectation Equation):

$$V^{\pi}(s) = \mathbb{E}_{\pi}(r_0 + \gamma r_1 + \cdots \ \mid s_t = s\,)$$

2. **Improve** the policy by acting greedily with respect to V^{π}:

$$\pi' = greedy(V^{\pi})$$

Dynamic Programming

Generalized Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - Greedy Policy Improvement over Q

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

$$\forall s \in \mathcal{S}, \qquad Q^{\pi'}\big(s, \pi'(s)\big) \geq Q^{\pi}\big(s, \pi(s)\big)$$

  - Greedy Policy Improvement over V

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^{\pi}(s') \right\}$$

$$\forall s \in \mathcal{S}, \qquad V^{\pi'}(s') \geq V^{\pi}(s')$$

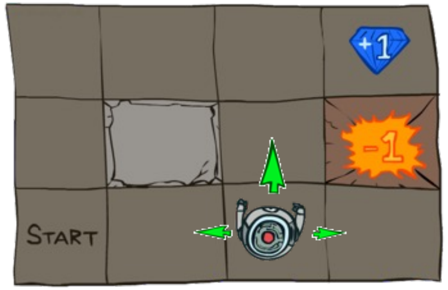
Dynamic Programming

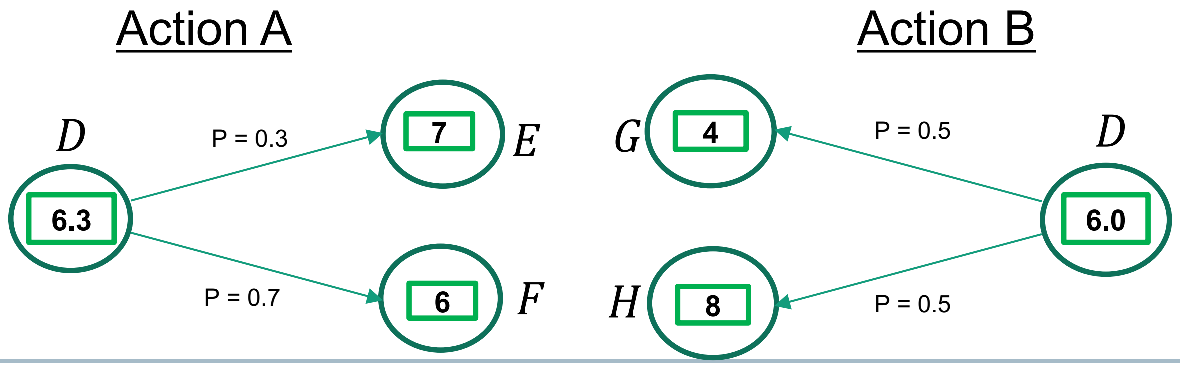
Generalized Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Greedy Policy Improvement over V

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^{\pi}(s') \right\}$$

$$\forall s \in \mathcal{S}, \qquad V^{\pi'}(s') \geq V^{\pi}(s')$$





Dynamic Programming:

Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   - $\Delta \leftarrow 0$
   - Loop for each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\big[r + \gamma V(s')\big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
   - If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

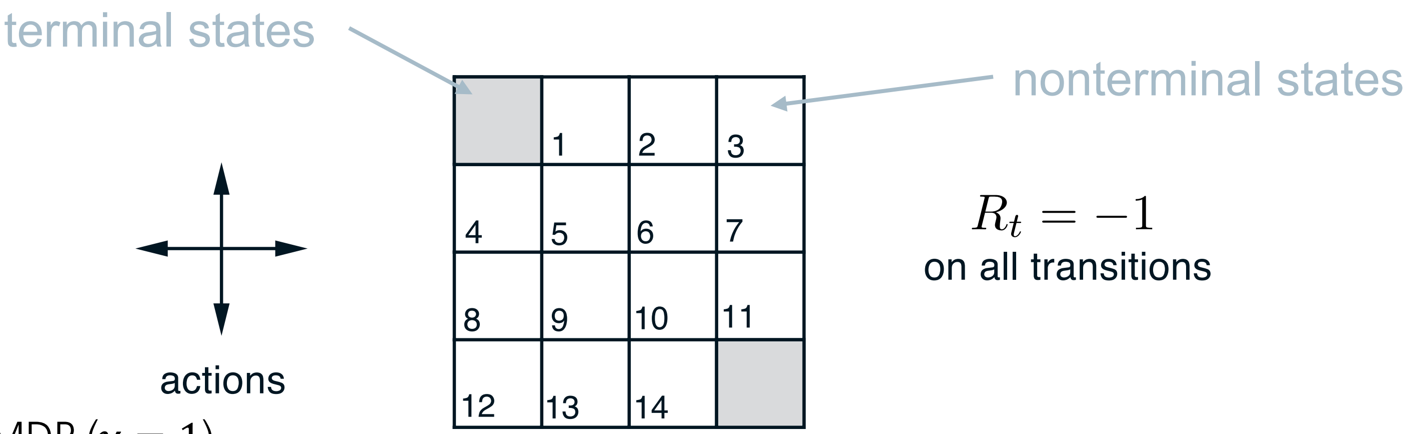
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Dynamic Programming

Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration: **Example for Policy Evaluation**



- Undiscounted episodic MDP ($\gamma = 1$)
- Actions leading out of the grid leave state unchanged
- Agent follows uniform random policy:

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Dynamic Programming

Policy Iteration

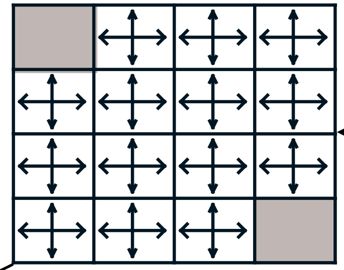
- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration: **Example for Policy Evaluation**

v_k for the random policy

greedy policy w.r.t. v_k

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

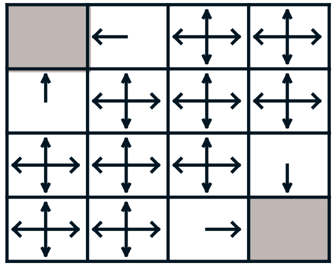


random policy

policy evaluation

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

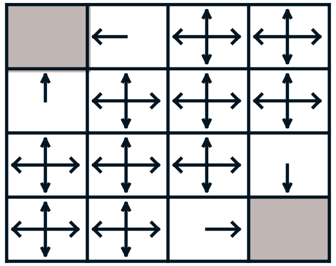


Dynamic Programming
Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration: **Example for Policy Evaluation**

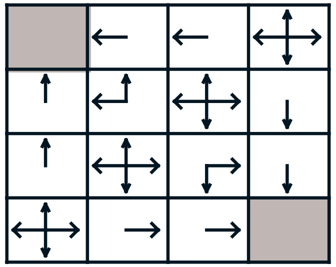
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



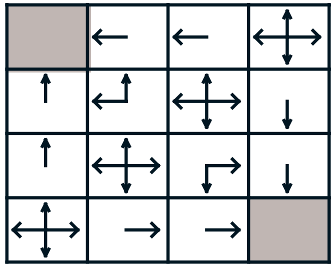
Dynamic Programming

Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration: **Example for Policy Evaluation**

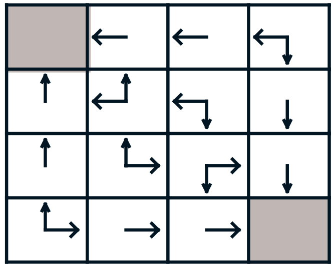
$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



Fraunhofer IIS

Dynamic Programming
Policy Iteration

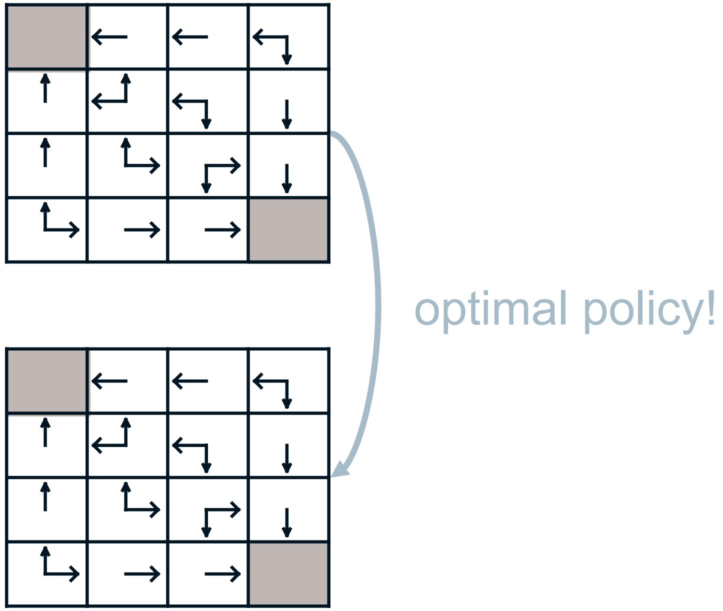
- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration: **Example for Policy Evaluation**

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



Dynamic Programming
Policy Iteration

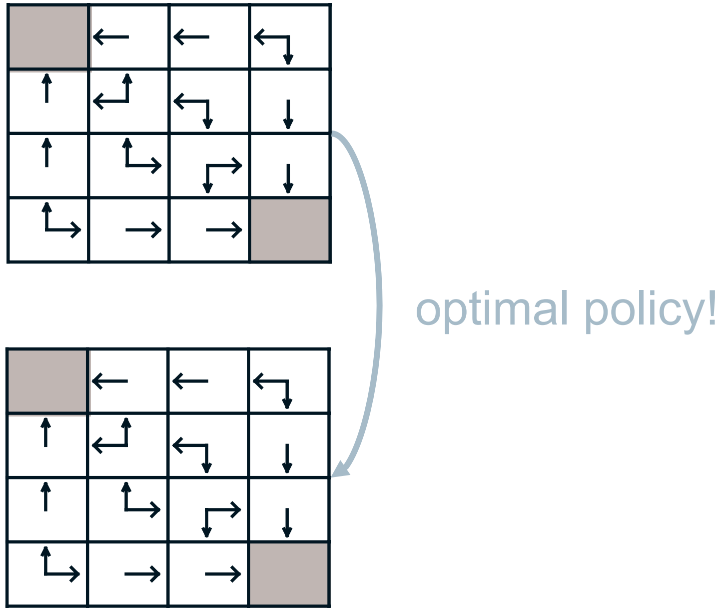
- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration: **Example for Policy Evaluation**

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

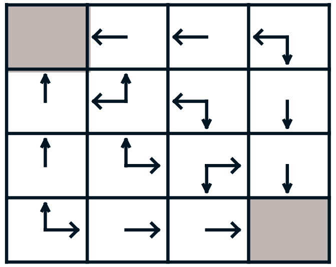


Dynamic Programming
Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration: **Example for Policy Evaluation**

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



- Iterative Policy Evaluation converges to V^{π}
- The converged greedy policy is guaranteed to be an improvement over the random policy
- In this case (any greedy policies after the third iteration) are optimal policies

Dynamic Programming
Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration: **Proof - Intuition**
  - Is **policy improvement** making π' better than π?
  - Assume that for some state $\pi'(s) = a \neq \pi(s)$. *Should we use the new policy? Is it better or is it worse?*
  - How good is it to choose a in s and then keep on following π:

$$Q^{\pi}(s,a) \doteq \mathbb{E}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, a_t = a]$$
$$= \underbrace{\sum_{s',r} \mathcal{P}(s',r|s,a)[r + \gamma V^{\pi}(s')]}_{\geq V^{\pi}(s)\ ?}$$

→ If this is better, we would expect that using a is always better when we are in s

→ …and that the new policy would then in turn be a better one overall!

Dynamic Programming

Policy Iteration

- Special case of the **Policy Improvement Theorem**
  - Let π and π' be any pair of deterministic policies such that
$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s), \qquad \forall\, s \in \mathcal{S}$$
  - Then π' must be as good as, or better than π, which means
$$V^{\pi'}(s) \geq V^{\pi}(s), \qquad \forall s \in \mathcal{S}$$
  - Consider again the special case $\pi'(s) = a \neq \pi(s)$ for exactly one $s \in \mathcal{S}$ (this is in line for the non-strict inequality formulation above)
  - Thus, if $Q^{\pi}(s, a) > V^{\pi}(s)$ then the changed policy is indeed better than π

$$\begin{aligned}
V^{\pi}(s) &\leq Q^{\pi}\big(s, \pi'(s)\big) \\
&= \mathbb{E}\;\;[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, a_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s] \\
&\leq \mathbb{E}_{\pi'}\big[r_{t+1} + \gamma Q^{\pi}\big(s_{t+1}, \pi'(s_{t+1})\big) | s_t = s\big] \\
&\leq \mathbb{E}_{\pi'}\big[r_{t+1} + \gamma r_{t+2} + \gamma^2 Q^{\pi}\big(s_{t+2}, \pi'(s_{t+2})\big) | s_{t+1}, a_{t+1} = \pi'(s_{t+1}) | s_t = s\big] \\
&\;\dots \\
&\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \cdots | s_t = s] \\
&= V^{\pi'}(s)
\end{aligned}$$

(see slide before)

(following from above

Note: a strict inequality at **any state** above leads to a strict inequality of at **least one** state below!

Fraunhofer IIS

Dynamic Programming

Policy Iteration

- Special case of the **Policy Improvement Theorem**
  - It is a natural extension to consider changes at all states and to all possible actions, in other words: to consider the new greedy policy π' given by:

$$\pi'(s) \doteq \arg\max_a Q^\pi(s,a)$$
$$\pi'(s) = \arg\max_a \mathbb{E}[r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a]$$
$$\pi'(s) = \arg\max_a \sum_{s',r} \mathcal{P}(s',r|s,a)[r + \gamma V^\pi(s')]$$

  - Greedy policy takes action that looks best (with one-step lookahead) and by construction, the greedy policy meets the policy improvement theorem (slide before)
  - Suppose that the new greedy policy π' is as good as but not better than π:

$$V^\pi(s) = \max_a \mathbb{E}\left[r_{t+1} + \gamma V^{\pi'}(s_{t+1}) | s_t = s, a_t = a\right]$$
$$V^\pi(s) = \max_a \sum_{s',r} \mathcal{P}(s',r|s,a)\left[r + \gamma V^{\pi'}(s')\right]$$

  - → This is the Bellman Optimality Equation; hence: $\pi = \pi' = \pi^*$

Dynamic Programming
Policy Iteration

- How do we find **optimal** controllers for given (known) MDPs?
- Optimal Solver #2: Policy Iteration

- Final remarks:
  1. We considered deterministic policies.
     - In the general case π specifies probabilities $\pi(a|s)$
     - Everything seen so far can easily be extended to this
  2. Do we need to evaluate upon convergence? → NO.
     - It is sufficient to execute one single sweep (instead of going to convergence)
     - Policy iteration breaks down to value iteration!

Dynamic Programming

Policy Iteration

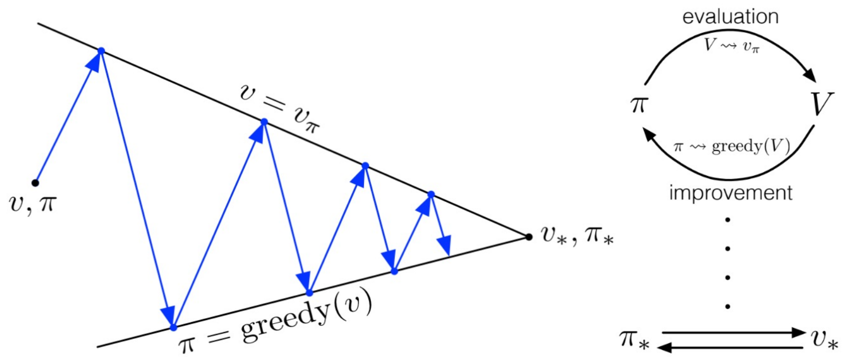
Hands-On:

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

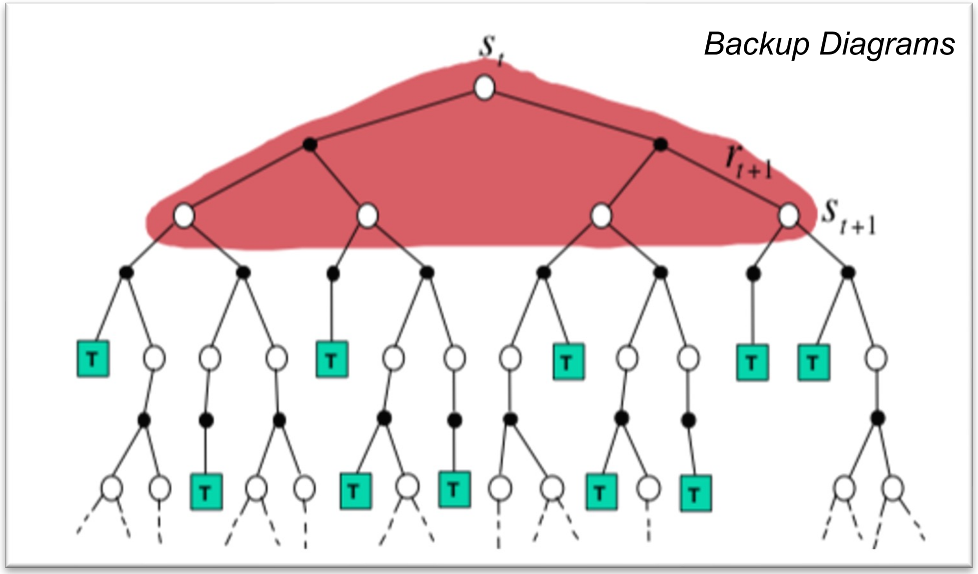
Dynamic Programming

Summary

- Dynamic Programming (DP) methods to find optimal controllers
  - DP methods are guaranteed to find optimal solutions for Q and V in polynomial time (in number of states and actions) and are exponentially faster than direct search
  - Policy Iteration computes the value function under a given policy to improve the policy while value iteration directly works on the states
    - Perform sweeps through the state set
    - Implement the Bellman equation update
    - Use bootstrapping



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

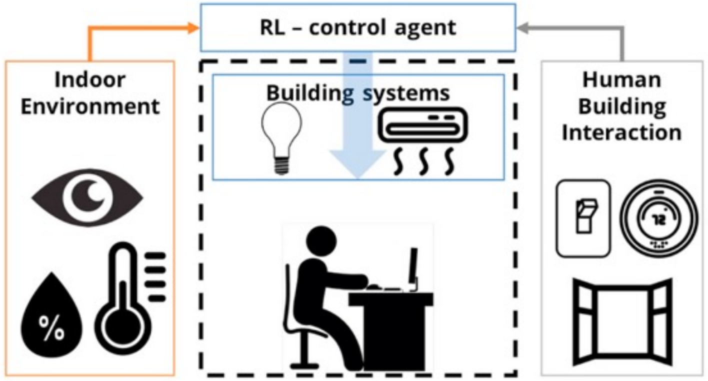
Dynamic Programming

Summary

- **But are these simple algorithms usable?**
- LightLearn: personalized lighting control with Value Iteration and learned transition model

Table 3
State definitions and rewards (note that P_4 is defined as both early morning and late night).

Occupancy	Switch position	Indoor light levels	Period of day				Reward
			P_4	P_1	P_2	P_3	
Occupied	Off	Dark	s_1	s_6	s_{13}	s_{20}	− 1
		Comfort	s_2	s_7	s_{14}	s_{21}	+ 1
		Bright	–	s_8	s_{15}	s_{22}	+ 1
	On	Comfort	s_3	s_9	s_{16}	s_{23}	+ 1
		Bright	–	s_{10}	s_{17}	s_{24}	0
Unoccupied	Off	–	s_4	s_{11}	s_{18}	s_{25}	+ 1
	On	–	s_5	s_{12}	s_{19}	s_{26}	− 1



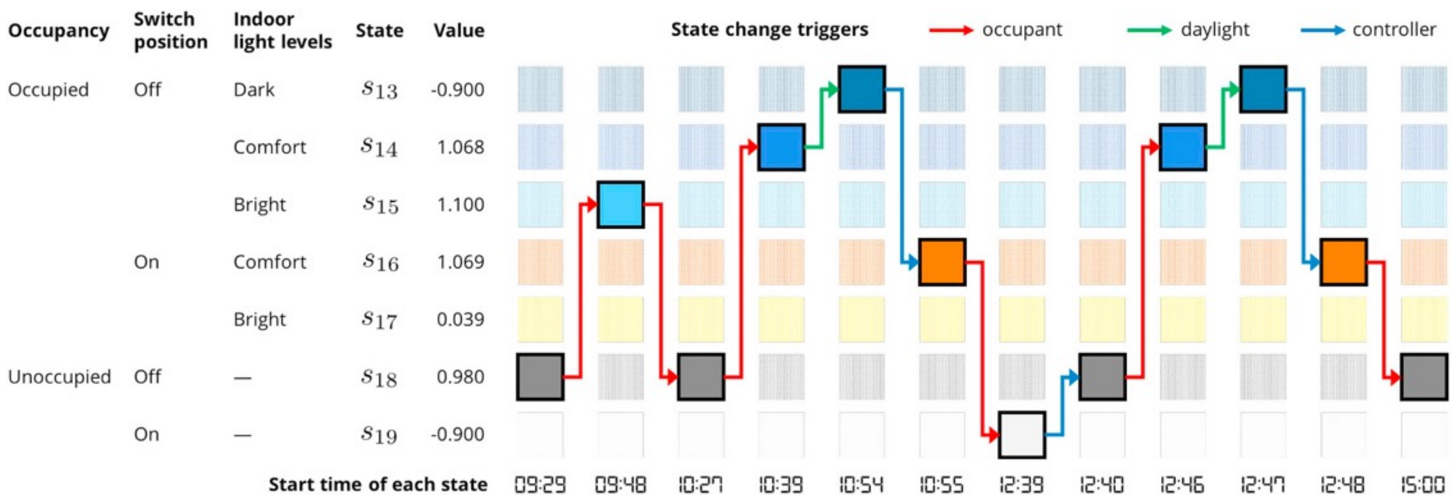
Park, J. Y., Dougherty, T., Fritz, H., & Nagy, Z. (2019). LightLearn: An adaptive and occupant centered controller for lighting based on reinforcement learning. *Building and Environment*, *147*, 397-414.

Fraunhofer IIS

Dynamic Programming

Summary

- **But are these simple algorithms usable?**
- LightLearn: personalized lighting control with Value Iteration and learned transition model

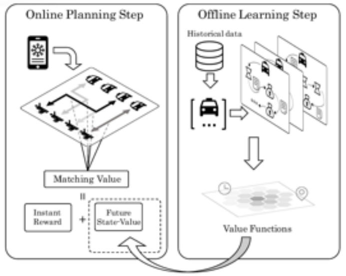


Park, J. Y., Dougherty, T., Fritz, H., & Nagy, Z. (2019). LightLearn: An adaptive and occupant centered controller for lighting based on reinforcement learning. *Building and Environment*, *147*, 397-414.

Dynamic Programming

Summary

- **But are these simple algorithms usable?**
- Large-scale order dispatch in on-demand ride-hailing platforms
  - Problem: find the best matching between drivers and orders (e.g. Uber)
  - Available information: each taxi uploads occupancy status and location in central platform
  - Classical solution: during each short time slot (say one or two seconds), the platform’s decision center first collects all the available drivers and active orders, and then matching is based on a combinatorial optimization algorithm.

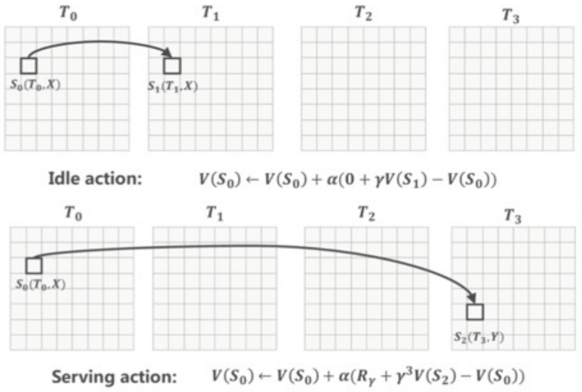


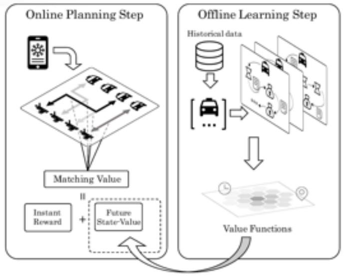
Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., ... & Ye, J. (2018, July). Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 905-913). ACM.

Dynamic Programming

Summary

- **But are these simple algorithms usable?**
- Large-scale order dispatch in on-demand ride-hailing platforms
  - Idea: design an order dispatch algorithm that optimizes the platform's global efficiency in a long horizon (e.g., two or three hours or a day), by formulating order dispatch as a large-scale sequential decision-making problem





Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., ... & Ye, J. (2018, July). Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 905-913). ACM.

Dynamic Programming

Summary

- **But are these simple algorithms usable?**
- Large-scale order dispatch in on-demand ride-hailing platforms
  - Reward: the price of an order → Goal: maximize the Gross Merchandise Volume for the entire platform
  - Goal: Online planning: takes the learned value functions as inputs and determines the final matching between drivers and orders in real-time

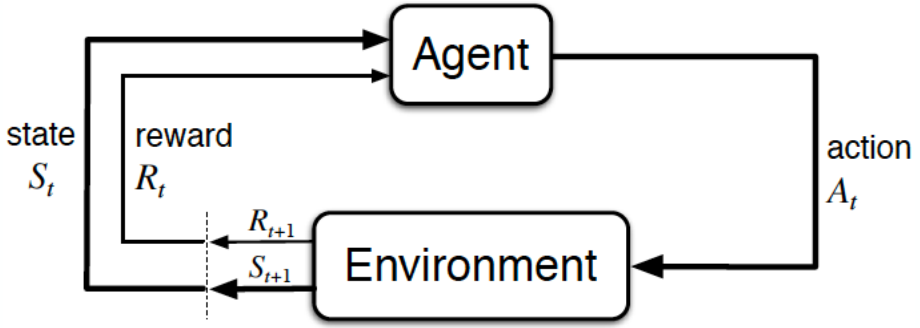
Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., ... & Ye, J. (2018, July). Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 905-913). ACM.

Recap: Intro to Reinforcement Learning

Summary

- The RL Paradigm (revisited):
  - Do you agree with following statement?

"All goals can be described by the maximization of expected cumulative reward."



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

thanks to @karpathy

Fraunhofer IIS

Summary
References

- Books:
  - Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.
  - Bellman, R.E. 1957. Dynamic Programming. Princeton University Press, Princeton, NJ. Republished 2003: Dover, ISBN 0-486-42809-5.

- Lectures:
  - UC Berkeley CS188 Intro to AI. http://ai.berkeley.edu/lecture_slides.html
  - UCL Course on RL. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
  - Advanced Deep Learning and Reinforcement Learning (UCL + DeepMind). http://www.cs.ucl.ac.uk/current_students/syllabus/compgi/compgi22_advanced_deep_learning_and_reinforcement_learning

- Blogs etc.:
  - https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html