Markov Decision Process*

Prof. Mohan Sridharan
Chair in Robot Systems
University of Edinburgh, UK
https://homepages.inf.ed.ac.uk/msridhar/
m.sridharan@ed.ac.uk

*Revised original slides that accompany the book: PR by Thrun, Burgard and Fox.

Motivation

- Robot perception not the ultimate goal:
  - Choose right sequence of actions to achieve goal.
- Planning/control applications:
  - Navigation, Surveillance, Monitoring, Collaboration etc.
  - Ground, air, sea, underground!
- Action selection non-trivial in real-world problems:
  - State non-observable.
  - Action non-deterministic.
  - Require dynamic performance.

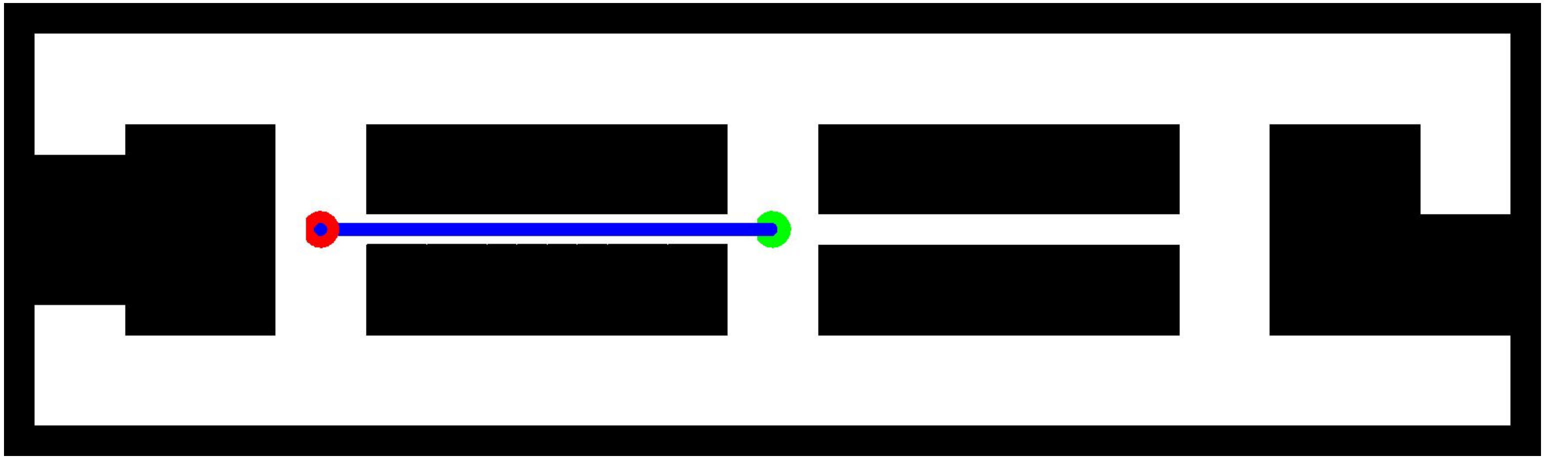
Problem Classes

- Deterministic vs. stochastic actions.
  - Classical approaches assume known action outcomes. Actions typically non-deterministic. Can only predict *likelihoods* of outcomes.

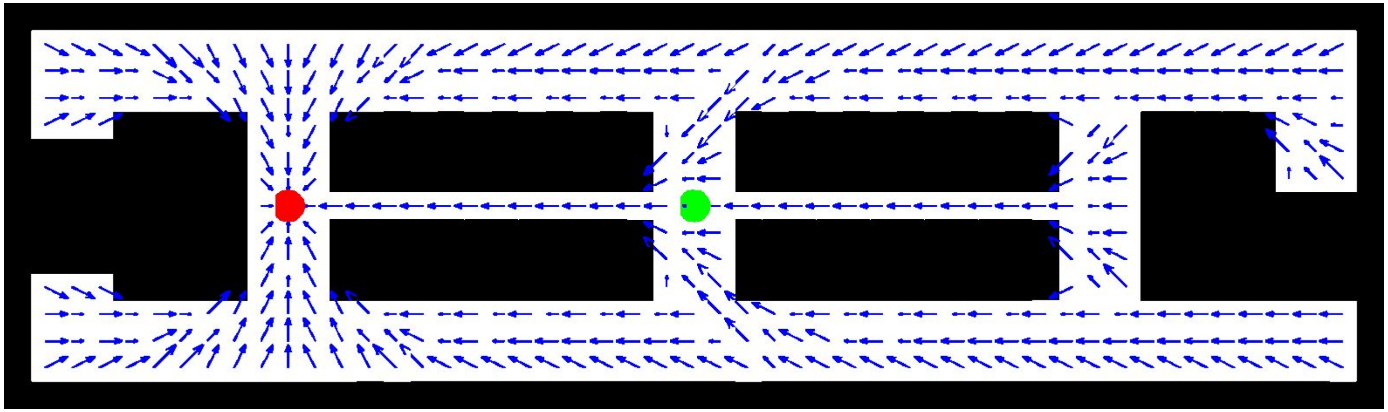
- Full vs. partial observability.
  - State of the system completely observable – never happens in real-world applications.
  - Build representation of the world by performing actions and observing outcomes.

- Current and anticipated uncertainty.

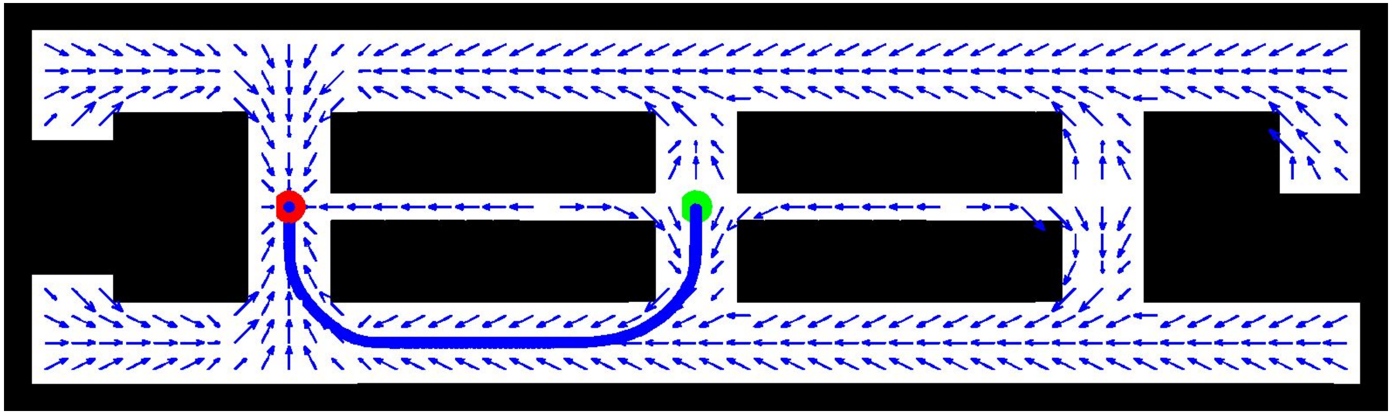
Deterministic, Fully Observable



Stochastic, Fully Observable

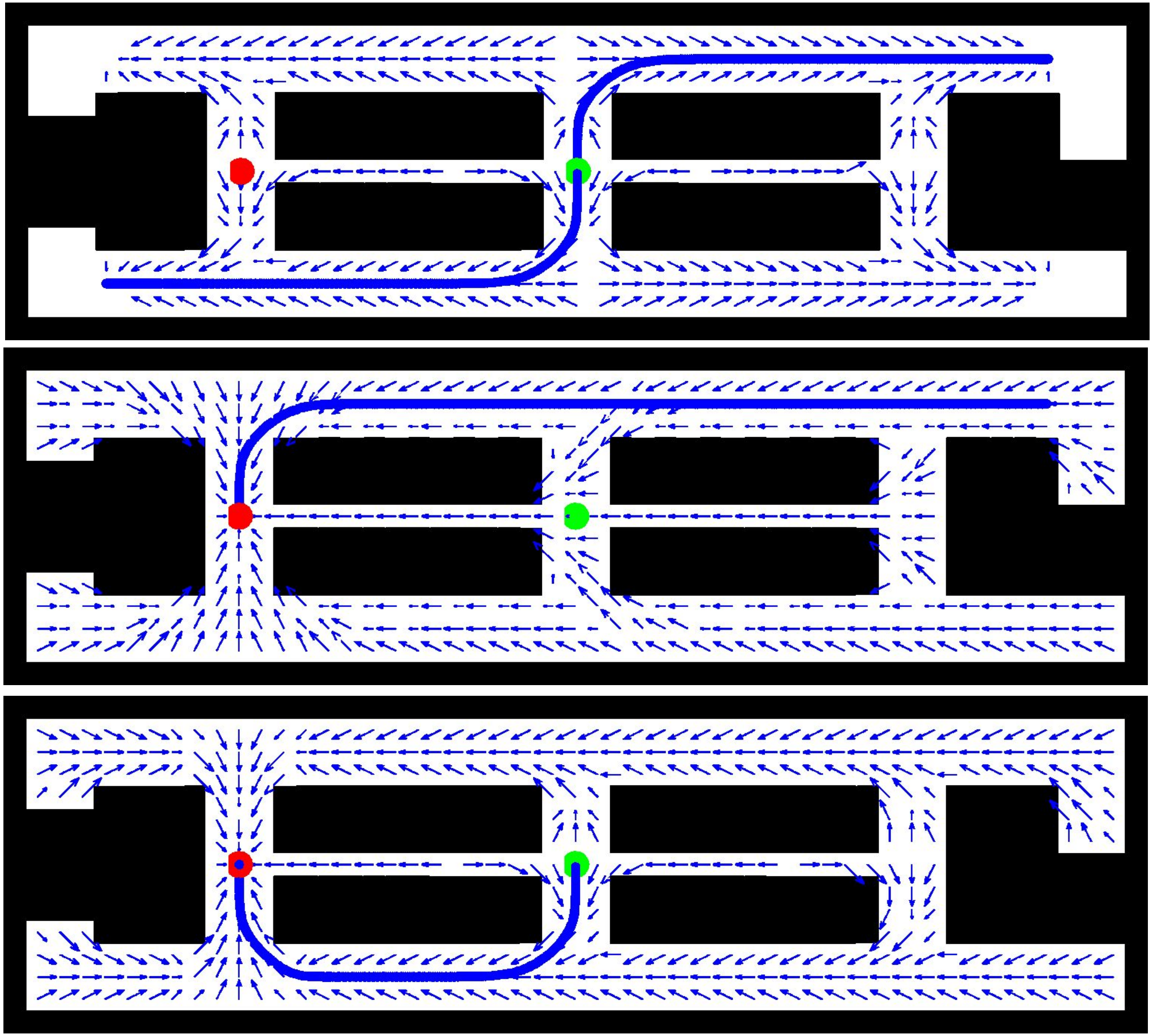


- No noise in Motion models.

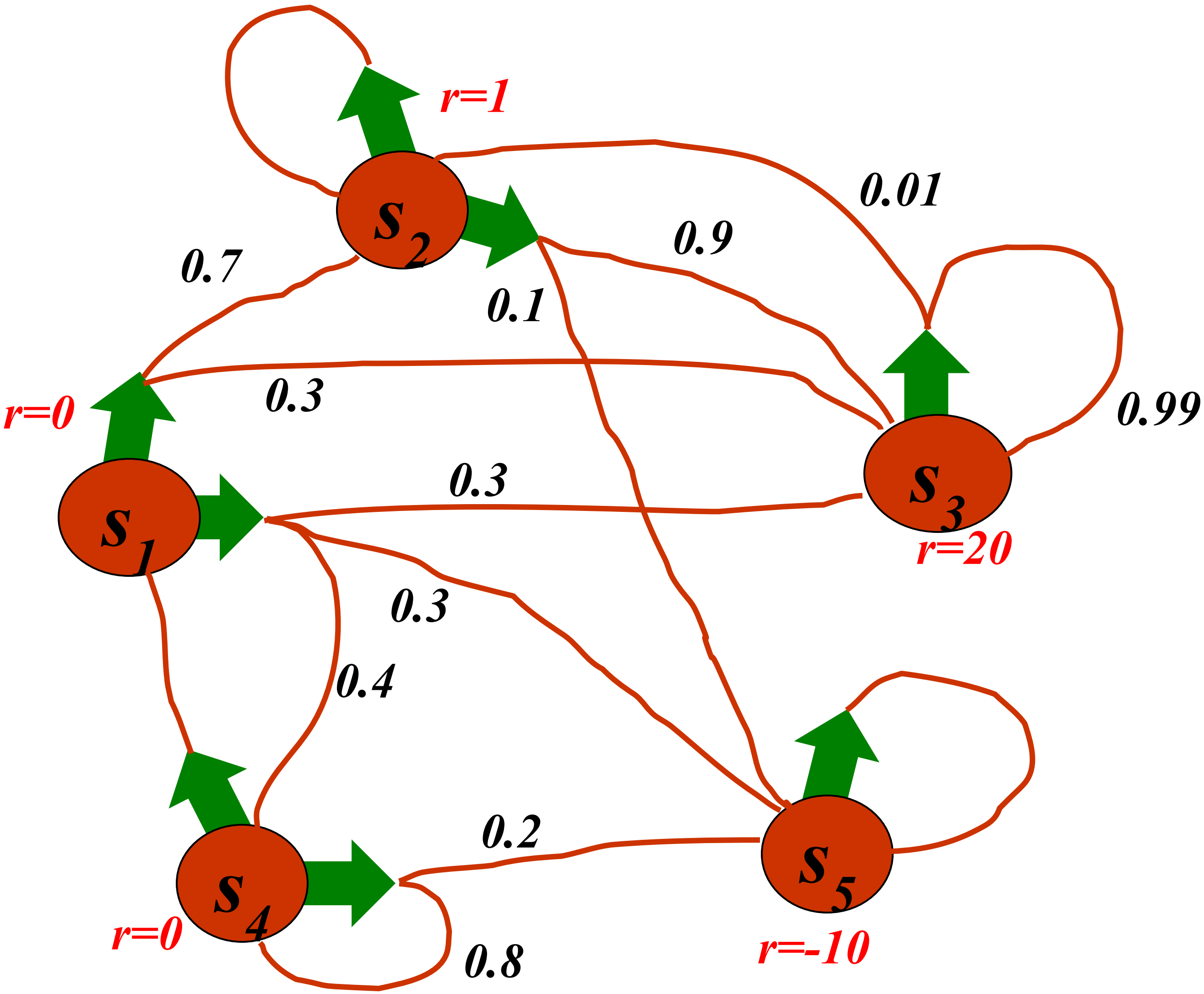


- Noisy motion models.

Stochastic, Partially Observable



Markov Decision Process (MDP)



Markov Decision Process (MDP)

- **Given:**
  - States: $X = \{x_1, x_2, \ldots, x_N\}$
  - Actions: $A = \{u_1, u_2, \ldots, u_N\}$
  - Transition probabilities: $p(x_{t+1} = x' \mid x_t = x, u_t = u)$
  - Reward / payoff function:
    $r(x, u, x') = E\left[r_{t+1} \mid x_t = x, u_t = u, x_{t+1} = x'\right]$
- **Wanted:**
  - Policy π that maximizes future expected reward.

Policies

- Policy (general case):
  - All past data mapped to control commands.

$$\pi : z_{1:t-1}, u_{1:t-1} \longrightarrow u_t$$

- Policy (fully observable case):
  - State mapped to control commands.

$$\pi : x_t \rightarrow u_t$$

Rewards

- Expected cumulative payoff:

$$R_t = E\left[\sum_{\tau=0}^{T} \gamma^{\tau} r_{t+\tau+1}\right]$$

  - Maximize sum of future payoffs!
  - Discount factor $\gamma \in [0,1]$: future reward is worth less!

  - T=1: greedy policy. Discount does not matter as long as $\gamma > 0$!

  - 1<T<∞: finite horizon case, typically no discount.

  - T=∞: infinite-horizon case, finite reward if $\gamma < 1$:

$$|r| < r_{\max}, \quad R_{\infty} = r_{\max} + \gamma r_{\max} + \gamma^2 r_{\max} + \ldots = \frac{r_{\max}}{1-\gamma}$$

Optimal Policies

- Expected cumulative payoff of policy:

$$R_t^{\pi}(x_t) = E\left[\sum_{\tau=0}^{T} \gamma^{\tau} r_{t+\tau+1} \mid u_{t+\tau} = \pi(x_t)\right]$$

- Optimal policy: $\pi^* = \underset{\pi}{\operatorname{argmax}}\, R_t^{\pi}(x_t)$

- 1-step optimal policy: $\pi_1(x) = \underset{u}{\operatorname{argmax}}\, r(x,u)$

- Value function of 1-step optimal policy:

$$V_1(x) = \max_u r(x,u)$$

Value Functions

- Value function for specific policy (Bellman equation for V^π)

$$V^\pi(x) = E_\pi\left[\sum_{\tau=0}^{\infty} \gamma^\tau r_{t+\tau+1} \mid x_t = x\right]$$

$$= E_\pi\left[r_{t+1} + \gamma \sum_{\tau=0}^{\infty} \gamma^\tau r_{t+\tau+2} \mid x_t = x\right]$$

$$= \sum_u \pi(x,u) \sum_{x'} p(x' \mid x,u)\left[r(x,u,x') + \gamma\, E_\pi\left\{\sum_{\tau=0}^{\infty} \gamma^\tau r_{t+\tau+2} \mid x_{t+1} = x'\right\}\right]$$

$$= \sum_u \pi(x,u) \sum_{x'} p(x' \mid x,u)\left[r(x,u,x') + \gamma V^\pi(x')\right]$$

$$Q^\pi(x,u) = E_\pi\left[\sum_{\tau=1}^{\infty} \gamma^\tau r_{t+\tau} \mid x_t = x, u_t = u\right]$$

Optimal Value Functions

- Optimal policy:

$$V^*(x) = \max_{\pi} V^{\pi}(x)$$

$$Q^*(x,u) = \max_{\pi} Q^{\pi}(x,u) = E\left[r_{t+1} + \gamma V^*(x_{t+1}) \mid x_t = x, u_t = u\right]$$

- Bellman optimality equations.

$$\begin{aligned} V^*(x) &= \max_{u \in u(x)} Q^{\pi^*}(x,u) \\ &= \max_{u} E\left[r_{t+1} + \gamma V^*(x_{t+1}) \mid x_t = x, u_t = u\right] \\ &= \max_{u} \sum_{x'} p(x' \mid x,u)\left[r(x,u,x') + \gamma V^*(x')\right] \end{aligned}$$

$$\begin{aligned} Q^*(x,u) &= E\left[r_{t+1} + \gamma \max_{u'} Q^*(x_{t+1},u') \mid x_t = x, u_t = u\right] \\ &= \sum_{x'} p(x' \mid x,u)\left[r(x,u,x') + \gamma \max_{u'} Q^*(x',u')\right] \end{aligned}$$

- Necessary and sufficient condition for optimal policy.

Value Iteration – Discrete Case

- For all *x* do

$$V(x) \leftarrow r_{\min}$$

- EndFor

- Repeat until convergence
  - For all *x* do

$$V(x) \leftarrow \max_u \sum_{x'} p(x' \mid x, u) \left[r(x, u, x') + \gamma V(x') \right]$$

  - EndFor
- EndRepeat

- Action choice:

$$\pi(x) = \operatorname*{argmax}_u \sum_{x'} p(x' \mid x, u) \left[r(x, u, x') + \gamma V(x') \right]$$

Value Iteration

- For all *x* do

$$V(x) \leftarrow r_{\min}$$

- EndFor

- Repeat until convergence
  - For all *x* do

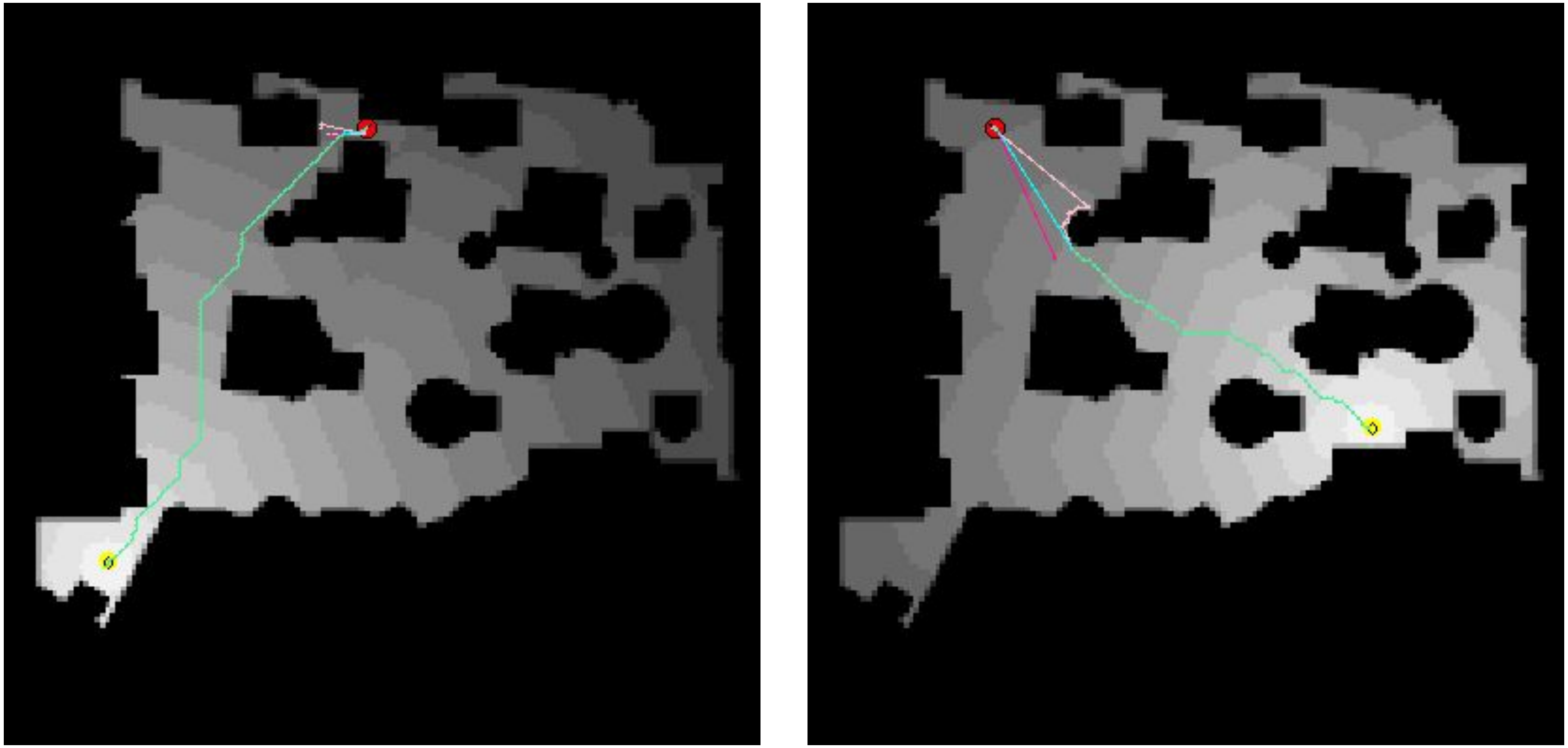
$$V(x) \leftarrow \max_u \int p(x' \mid x, u)\left[r(x, u, x') + \gamma\, V(x') \right] dx'$$

  - EndFor
- EndRepeat

- Action choice:

$$\pi(x) = \operatorname*{argmax}_u \int p(x' \mid x, u)\left[r(x, u, x') + \gamma\, V(x') \right] dx'$$

Value Iteration for Motion Planning

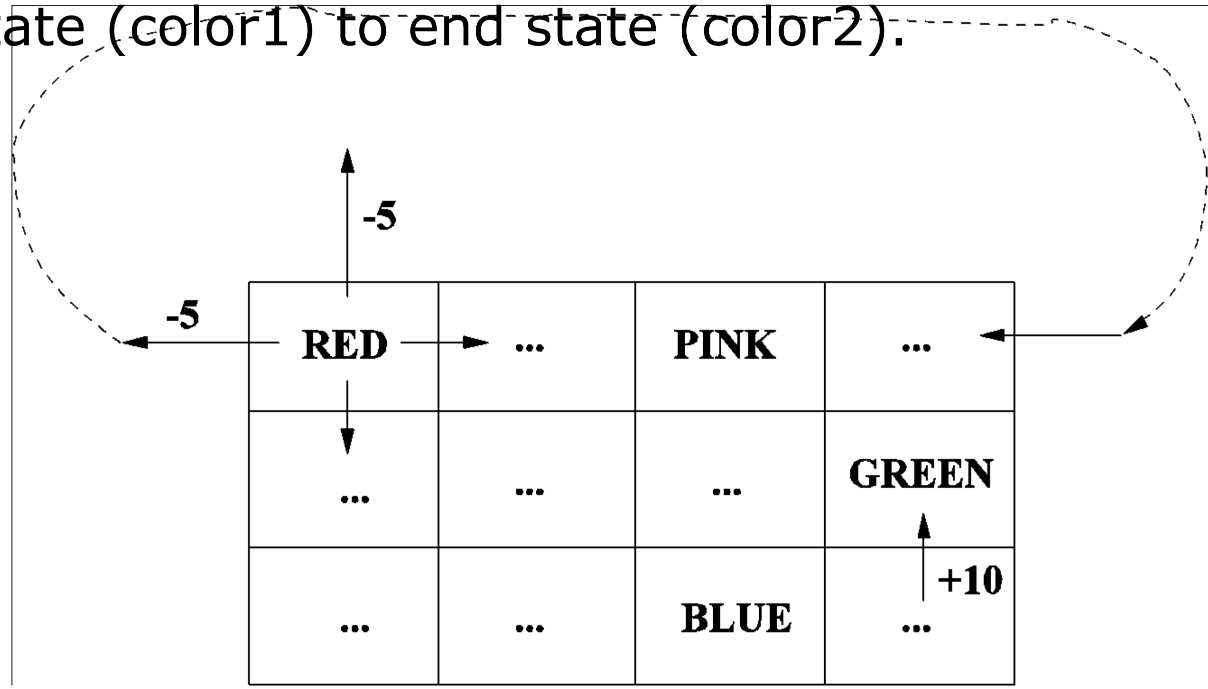


Value Function and Policy Iteration

- Often the optimal policy has been reached long before the value function has converged.
- Policy iteration calculates a new policy based on the current value function and then calculates a new value function based on this policy.
- This process often converges faster to the optimal policy.

Value-Iteration Game ☺️

- Move from start state (color1) to end state (color2).
- Maximize reward.

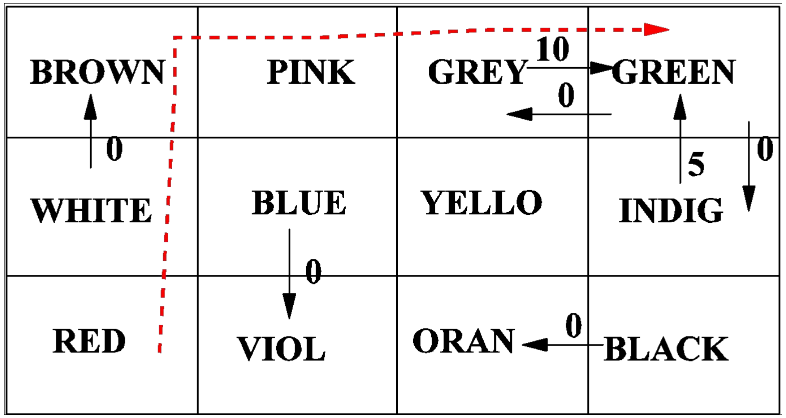


- Four actions.
- Twelve colors.

- Exploration and exploitation.

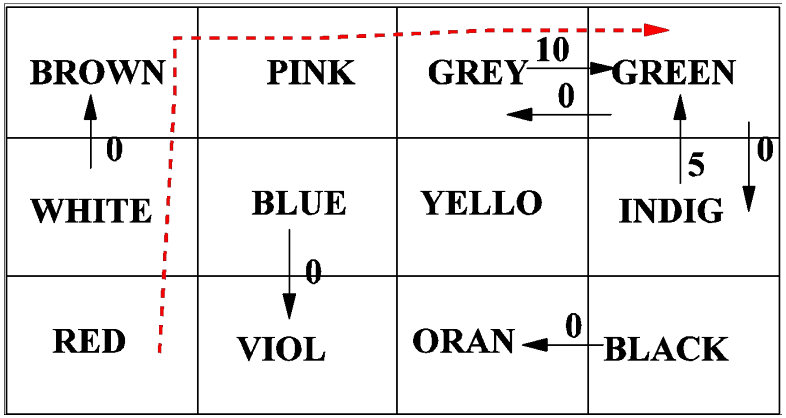
Value Iteration

- Explore first and then exploit!



- One-step look ahead values?
- N-step look ahead?
- Optimal policy?

Value Iteration – A few steps...



0	0	0	0
0	0	0	0
0	0	0	0

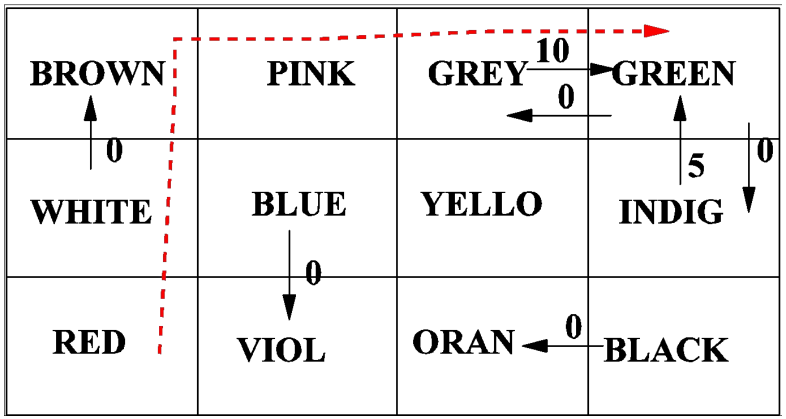
			5
-1	-1	-1	-1

1	2	3	4

11	12	13	14

21	22	23	24

Value Iteration – A few steps...



0	0	0	0
0	0	0	0
0	0	0	0

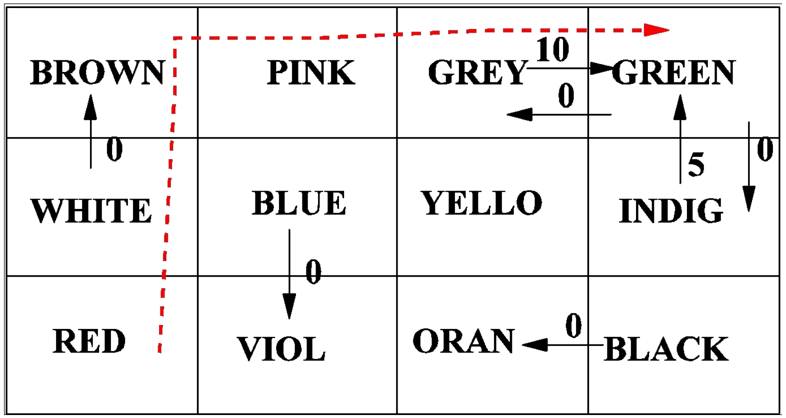
1	2	10	10
2	3	4	5
-1	-1	-1	-1

11	12	20	20
12	13	14	15
1	2	3	4

21	22	30	30
22	23	24	25
11	12	13	14

31	32	40	40
32	33	34	35
21	22	23	24

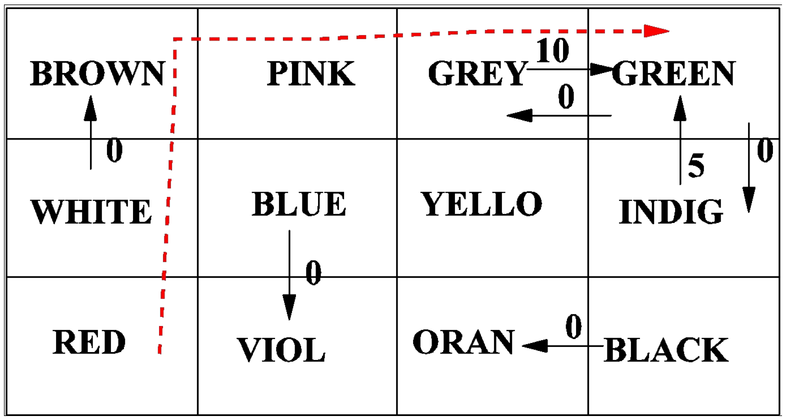
Value Iteration – A few steps...



0	0	0	0
0	0	0	0
0	0	0	0

-1	-1	10	10

Value Iteration – A few steps…



0	0	0	0
0	0	0	0
0	0	0	0

-1	-1	10	10
-1	0	9	15
-1	-1	8	14

5	9	20	20
10	8	19	25
9	7	18	24

15	19	30	30
20	18	29	35
19	17	28	34

25	29	40	40
30	28	39	45
29	27	38	44

More Information

- Chapters 3-4 of Reinforcement Learning textbook by Sutton and Barto (second edition).

  http://incompleteideas.net/book/the-book-2nd.html