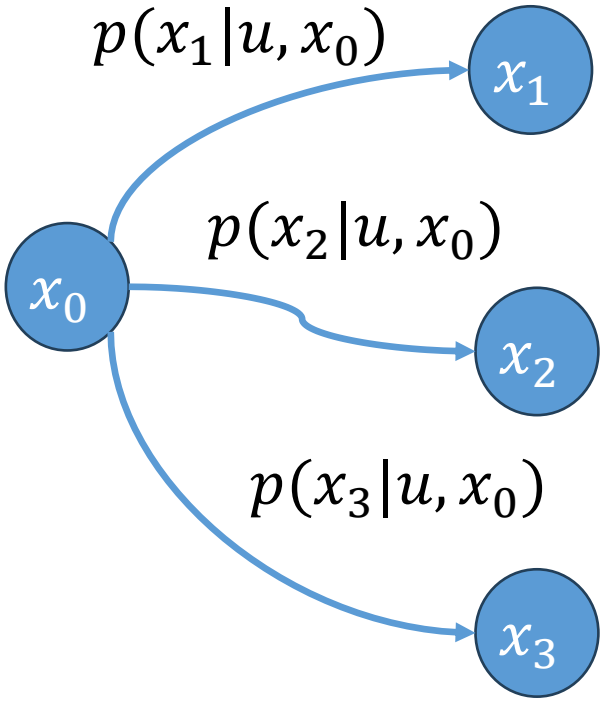
COLUMBIA UNIVERSITY EEME E6911 FALL ‘25

TOPICS IN CONTROL : PROBABILISTIC ROBOTICS

# MARKOV DECISION PROCESS

Instructor: Ilija Hadzic

# Fully Observable MDP




- Present state is known:
  - No uncertainty
- Transitions are uncertain:
  - Given the input
  - Randomness in transition
- Example:
  - Ground robot skidding
  - Drone disturbed by the wind
- Fully observable
- Partially controllable

# Reward, Value, Horizon, Discount

- $r(x, u)$: reward/penalty of moving
  - Typically, small negative for just moving around.
- $\mathrm{V}(x)$: value of the state
  - Typically, high positive for goal state(s).
  - Typically, high negative for lethal states.
  - Calculated for transitional states.
- $T$: planning horizon
  - How far into the future do we calculate the reward.
- $\gamma$: discount
  - Decay factor for future rewards.

# Value Iteration

- Main idea:
  - Action rewards → some states more valuable than others
  - Infinite planning horizon → state values converge
  - Certain state → pick action with highest value gain
  - Uncertain action → use *expected* value gain
- Application:
  - Use as global planner
  - One time state value calculation per goal: high computation
  - Policy (action) extraction: low computation
  - Approximations for partially-observable MDPs:
    - QMDP, AMDP, MC-MDP

# Value Iteration

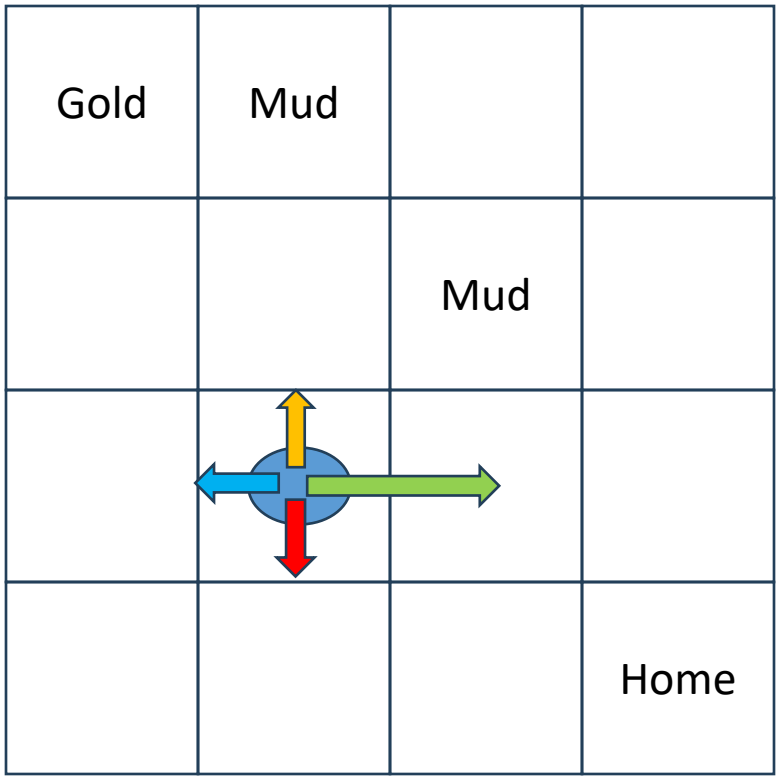
- Assign value to each state
  - Initialize: $V_0(x) = 0, \forall x$ (usually zero, but can be anything)
  - Set the iterative update:
    - $V_{n+1}(x) = \gamma \max_u \left( r(x,u) + \int V_n(x^+) p(x^+|u,x) dx^+ \right)$
  - Let $n \to \infty$:
    - $V(x) = \gamma \max_u \left( r(x,u) + \int V(x^+) p(x^+|u,x) dx^+ \right)$
  - This is known as Bellman equation
- Pick the motion (policy extraction):
  - Maximize the *expected* value of the next state
    - $u(x) = \operatorname{argmax}_u \left( r(x,u) + \int V(x^+) p(x^+|u,x) dx^+ \right)$

# Example: Quest on a Grid

| | | | |
|---|---|---|---|
| Gold | Mud | | |
| | | Mud | |
| | | | |
| | | | Home |

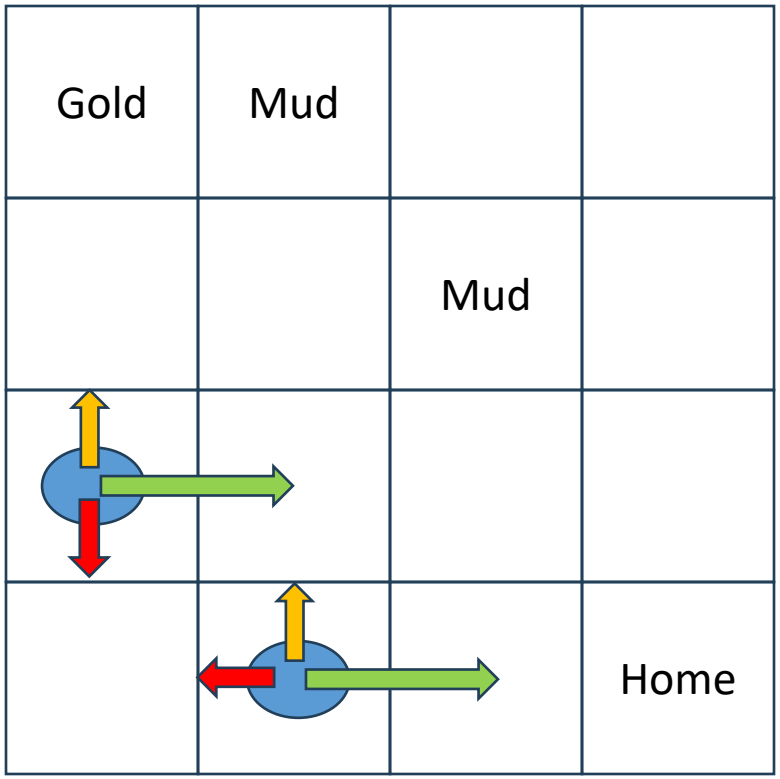
- Start: Home
- Goal: Find Gold
- Constraints:
  - Avoid stepping in mud
- Reward for gold: +50
- Penalty for mud: -100
- Cost of moving: -1
- Discount factor: 0.9

# Motion probabilities – center square



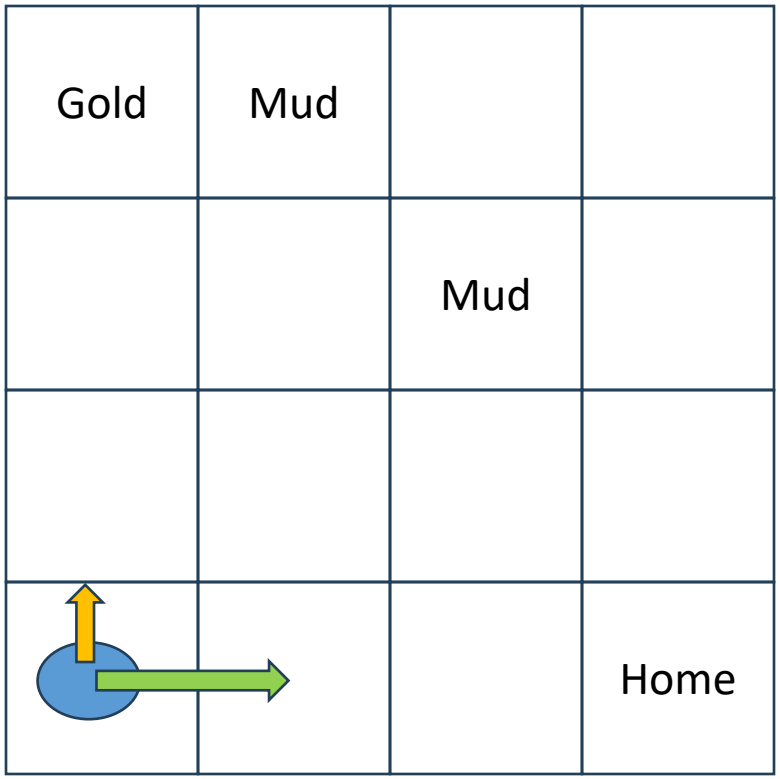

- Requested direction:
  - p=0.7
- Bounce:
  - p=0.1
- Skid:
  - p=0.1
- Skid:
  - p=0.1

# Motion probabilities – edge square




- Requested direction:
  - p=0.8
- Skid:
  - p=0.1
- Skid (or bounce):
  - p=0.1

# Motion probabilities – corner square



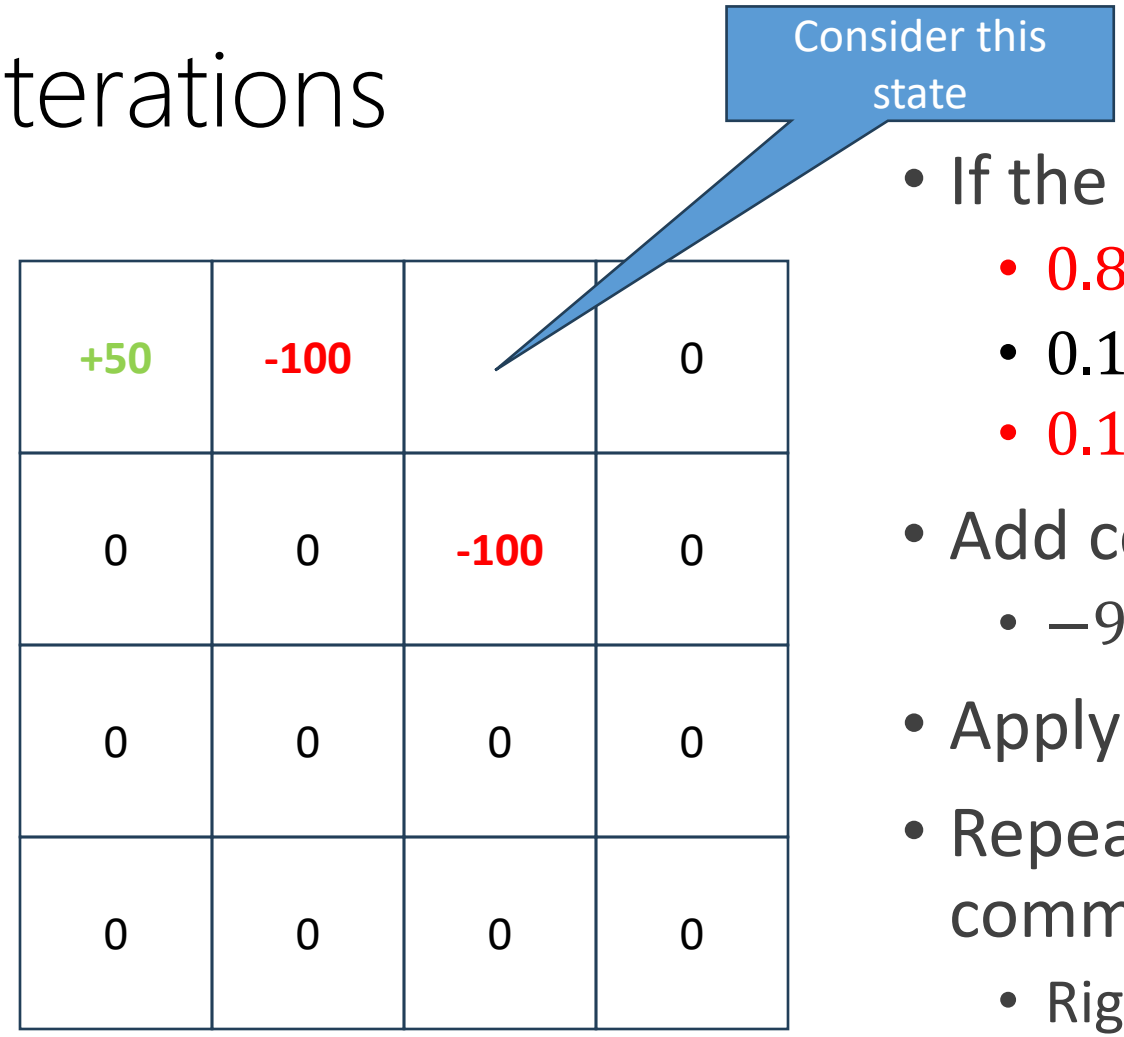

- Requested direction:
  - p=0.9
- Skid:
  - p=0.1

# Initial score

| | | | |
|---|---|---|---|
| +50 | -100 | 0 | 0 |
| 0 | 0 | -100 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

- Mud and gold:
  - Terminal states
  - No motion from them
  - Value set to +50 and -100
- All other states:
  - Initialize to 0

# Iterations




- If the command is “left”:
  - $0.8 \cdot -100$
  - $0.1 \cdot 0$
  - $0.1 \cdot -100$
- Add cost of motion:
  - $-90 - 1 = -91$
- Apply discount: $-81.9$
- Repeat for other two commands:
  - Right: $-18.9$
  - Down: $-81.9$
- Max: $-18.9$ → New value

# Iteration 1

| | | | |
|---|---|---|---|
| +50 | -100 | -18.9 | -0.9 |
| 35.1 | -18.9 | -100 | -9.9 |
| -0.9 | -0.9 | -9.9 | -0.9 |
| -0.9 | -0.9 | -0.9 | -0.9 |

- Continue the process for other states.
- Careful:
  - Do not mix up present and updated values
  - Pick only the maximum
  - Watch the sign when comparing

# Iteration 2

| | | | |
|---|---|---|---|
| **+50** | **-100** | -19.55 | -10.62 |
| 33.32 | 3.13 | **-100** | -10.63 |
| 24.21 | -4.14 | -10.63 | -3.33 |
| -1.71 | -1.71 | -2.52 | -1.71 |

- Repat the process until convergence

# Iteration 3

| | | | |
|---|---|---|---|
| **+50** | **-100** | -26.55 | -11.27 |
| 37.56 | 1.72 | **-100** | -13.25 |
| 22.56 | 13.52 | -12.16 | -4.04 |
| 18.56 | -2.73 | -3.24 | -3.24 |

- Repat the process until convergence

# Convergence

| | | | |
|---|---|---|---|
| +50 | -100 | -23.53 | -6.43 |
| 38.57 | 7.37 | -100 | -4.22 |
| 31.21 | 21.92 | 6.16 | 8.7 |
| 26.32 | 21.49 | 16.3 | 13.09 |

- Converges after 29 iteration for $\epsilon = 0.001$

# Policy Extraction

| | | | |
|---|---|---|---|
| +50 | -100 | -23.53 | -6.43 |
| 38.57 | 7.37 | -100 | -4.22 |
| 31.21 | 21.92 | 6.16 | 8.7 |
| 26.32 | 21.49 | 16.3 | 13.09 |

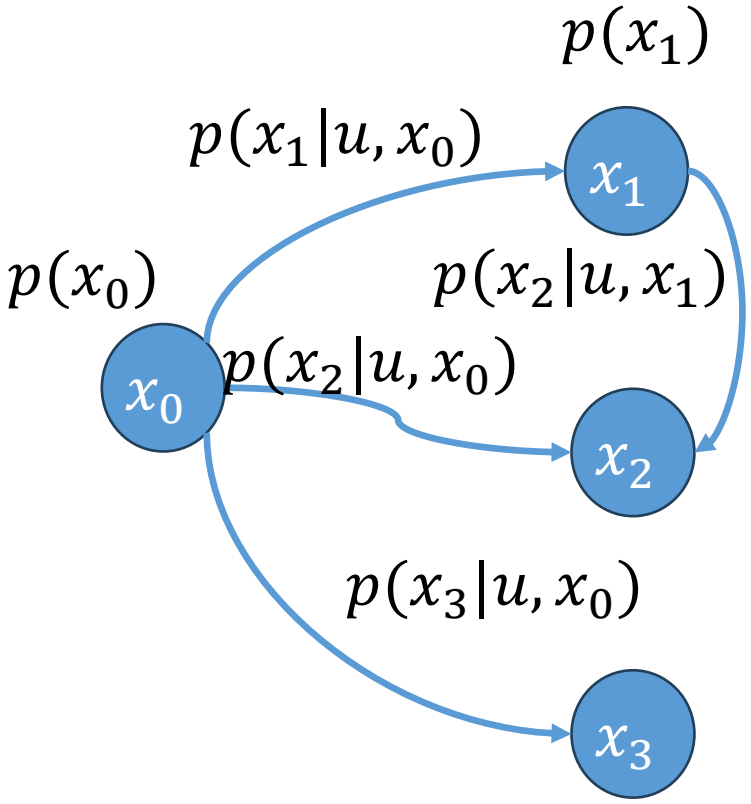
- Where will the robot go?
- Left? Add this up:
  - $0.7 \cdot 31.21$
  - $0.1 \cdot 7.37$
  - $0.1 \cdot 21.49$
  - $0.1 \cdot 6.16$
- Less the cost of motion
  - $25.35 - 1 = 24.35$
- Down?
  - $18.52 - 1 = 17.52$
- Up? Right?
- Left maximizes the gain!

# Policy Extraction

| | | | |
|---|---|---|---|
| +50 | -100 | -23.53 | -6.43 |
| 38.57 | 7.37 | -100 | -4.22 |
| 31 | .92 | 6.16 | 8.7 |
| 26.32 | 21.49 | 16.3 | 13.09 |

- It may be tempting to just pick the highest-value neighbor
- Must factor motion uncertainty!
- Intuitive policy:
  - Steer clear of mud-pits!
  - Value Iteration has learned that!

# Partially Observable MDP




- Present state is uncertain:
  - Randomness in state belief
- Transitions are uncertain:
  - Given the input
  - Randomness in transition
- Example:
  - Ground robot skidding
  - Drone disturbed by the wind
  - Probabilistic localization
- Fully observable
- Partially controllable

# Partially Observable MDP

- Must consider full state space with associated PDF
- Exact solution not tractable (see Chapter 15)
- Practical solutions are approximate:
  - QMDP
  - AMDP
  - MC-MDP

# Partially Observable MDP

- Must consider full state space with associated PDF
- Exact solution not tractable (see Chapter 15)
- Practical solutions are approximate:
  - QMDP
  - AMDP
  - MC-MDP

We will only discuss this

# QMDP

- Calculate state values: same as MDP

QMDP

- Consider full state space
  - Associated with belief PDF
- Calculate expected future values for all states

$$Q(x_i, u) = r(x_i, u) + \sum_{j=1}^{N} V(x_j) p(x_j | u, x_i)$$

- Pick control

$$u = \underset{u}{\operatorname{argmax}} \sum_{i=1}^{N} p_i Q(x_i, u)$$

MDP

- Focus on present state $x$
  - Known with certainty
- Pick control
  - Maximize expected future value

$$u = \underset{u}{\operatorname{argmax}} \left[ r(x, u) + \sum_{j=1}^{N} V(x_j) p(x_j | u, x) \right]$$

# QMDP

- Calculate state values: same as MDP

QMDP

- Consider full sta
  - Associated wit
- Calculate expecte ure values for all states

$$Q(x_i, u) = r(x_i, u) + \sum_{j=1}^{N} V(x_j) p(x_j | u, x_i)$$

Mean over state-space transitions (future states)

- Pick control

$$u = \underset{u}{\operatorname{argmax}} \sum_{i=1}^{N} p_i Q(x_i, u)$$

Mean over state belief (present state)

MDP

- Focus on present state $x$
  - Known with certainty
- Pick control
  - Maximize expected future value

$$u = \underset{u}{\operatorname{argmax}} \left[ r(x, u) + \sum_{j=1}^{N} V(x_j) p(x_j | u, x) \right]$$

Only one present state