CS 687
Jana Kosecka

Markov Decision Processes

Ch. 17

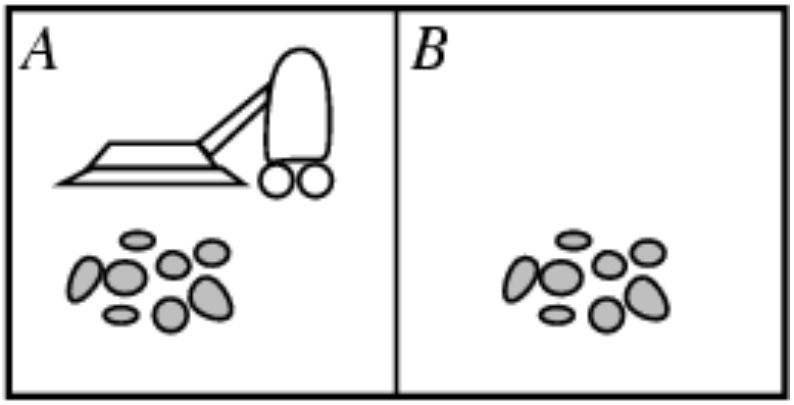
Planning and seach

- Previously, problem solving by search A*
- Idea – get a state space, initial stage, goal state
  come up with a plan
- Everything is off-line
- What if things change ? (e.g. walking blind folded)

- Need to interleave planning and executing

- Environment is stochastic (driving example)
- Partially observable, there can be many agents
- Model of the world is uknown, plans are hierarchical

Problem characteristics



Room 2 locations – 8 possible states
3 possible actions – move left, right, suck the dirt

What are some of the challenges of solving even this simple version in the real world?

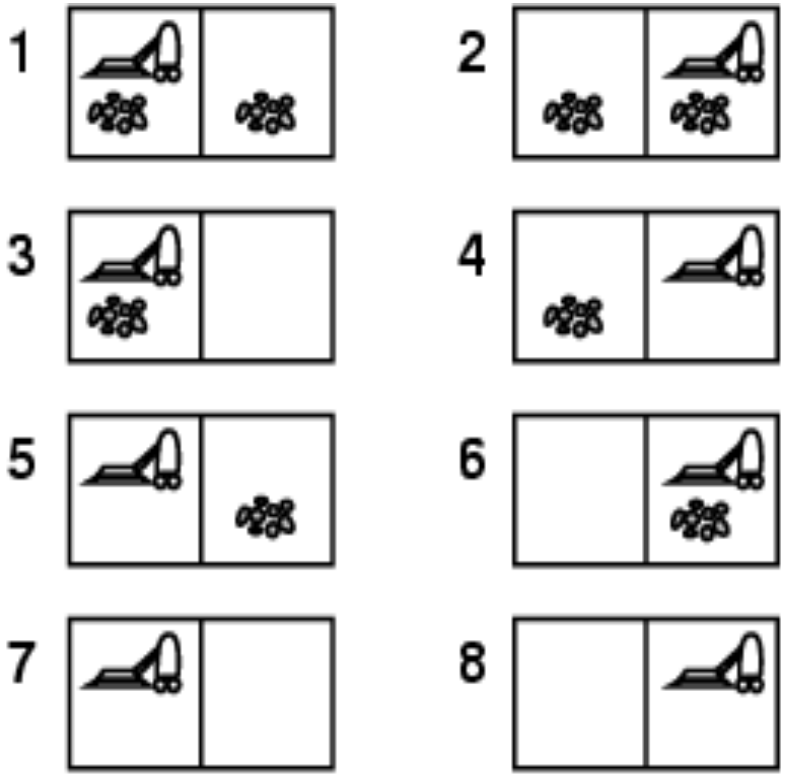
Problem characteristics

- Fully observable vs. partially observable
  - do we have access to all of the *relevant* information
  - noisy information, inaccurate sensors, missing information
- Deterministic vs. non-deterministic (stochastic)
  - outcome of actions are not always certain
  - probabilistic sometimes
- Known/unknown environment
  - Do we know a priori what the problem space is like (e.g. do we have a map)

Example: vacuum world

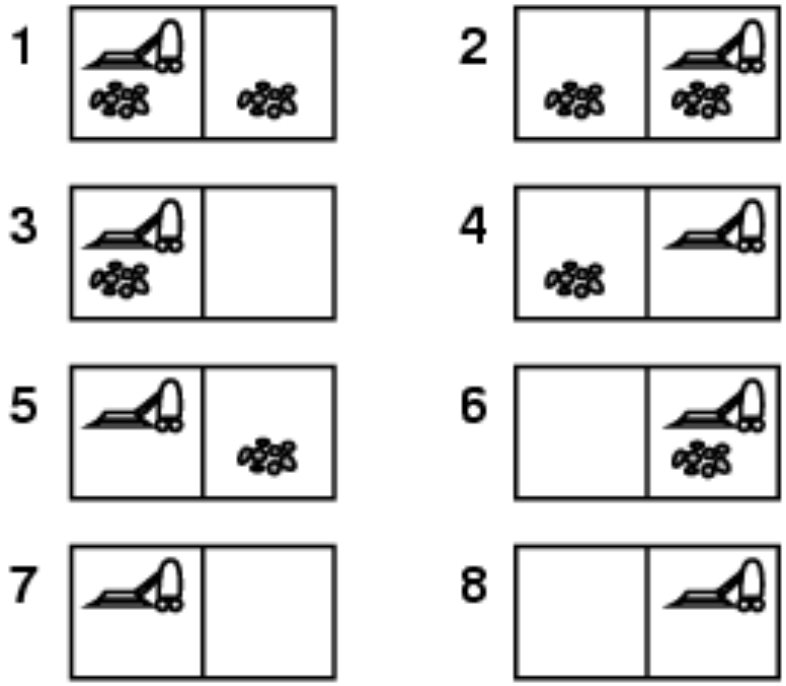
- Deterministic, fully observable
- Actions L, R, S (left, right, suck)
- start in #5. Solution?

- Graph search to Get to the goal state 8
- Graph corresponding to actions edges not drawn here



Example: vacuum world

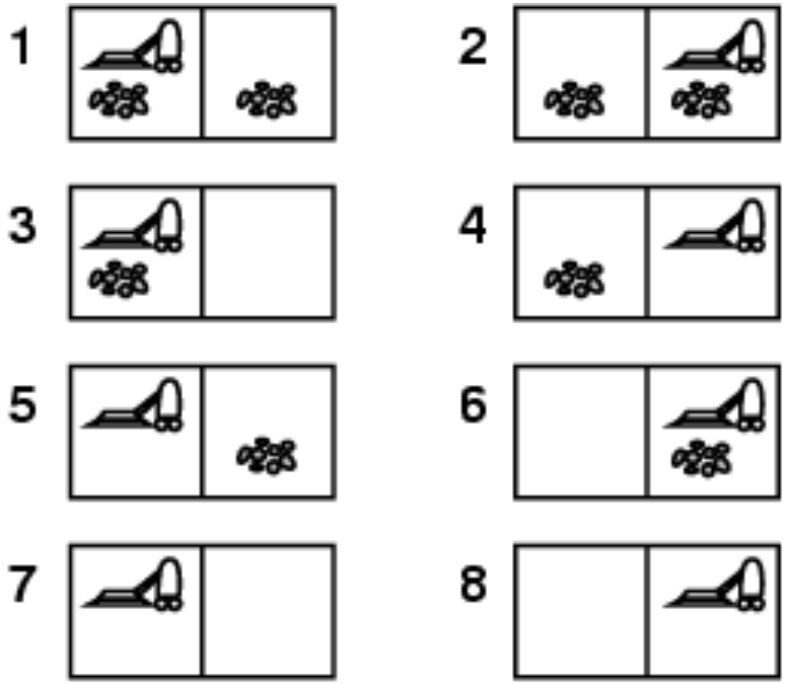
- Sensorless – don't know where you are, not sure if there is dirt
- start in {*1,2,3,4,5,6,7,8*}
  Solution?



- Cannot do anything – state is unknown
- Strategy represent belief state

Example: Vacuum world

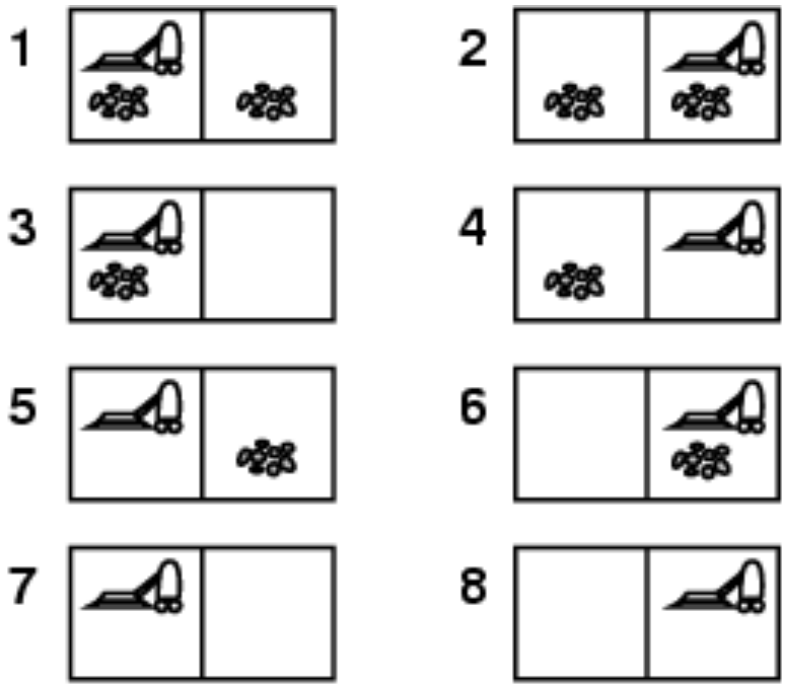
- Deterministic world and partially observable
  - Partially observable: location,
    dirt at current location – local sensing
  - Outcomes of actions are deterministic
  - Percept: *[L, Clean],*
    i.e., start in #5 or #7
    Solution?
  - Planning in belief space
  - each belief state corresponds
  - to a set of worlds stages



Example: Vacuum world

- Non-deterministic and/or partially observable
  - Nondeterministic: *Suck* may dirty a clean carpet
  - Partially observable: location, dirt at current location.
  - Percept: *[L, Clean],* i.e., start in #5 or #7

Solution?



Uncertainty and decisions

- Previously how to do state estimation under uncertainty
- Uncertainty can affect how the robot makes decisions
- How to encode preferences, between different outcomes of the plans (sequences of actions)

- Utility theory – reasoning about preferences
- Every state has some utility

- Decision theory = probability theory + utility theory

- Principle of maximum expected utility – agent is rational if it chooses an action with the highest expected utility

Markov Decision Process

- Formal definition
- 4-tuple (S, A, T, R)
- Set of states S - finite
- Set of actions A - finite
- Transition model $T : S \times A \times S \rightarrow [0,1]$

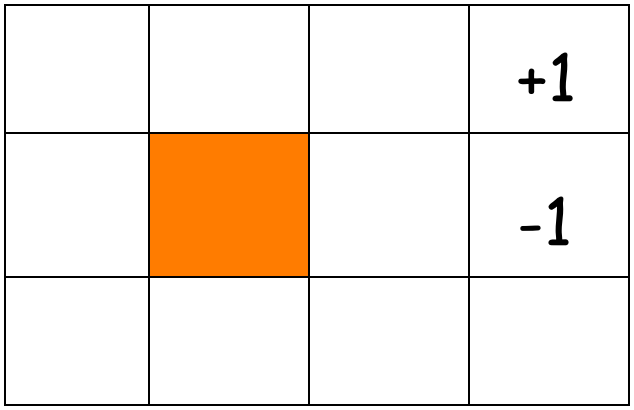
  Transition probability for each action, state
- Reward model $S \times A \times S \rightarrow R$

  $S \rightarrow R$

Example

- Robot navigating on the grid
- 4 actions – up, down, left, right
- Effects of moves are stochastic, we may end up in other state then indented with non-zero probability
- Reward +1 for reaching the goal
- Goal: find the policy sequence of actions $\pi: \quad s_t \rightarrow \quad a_t$

e.g.



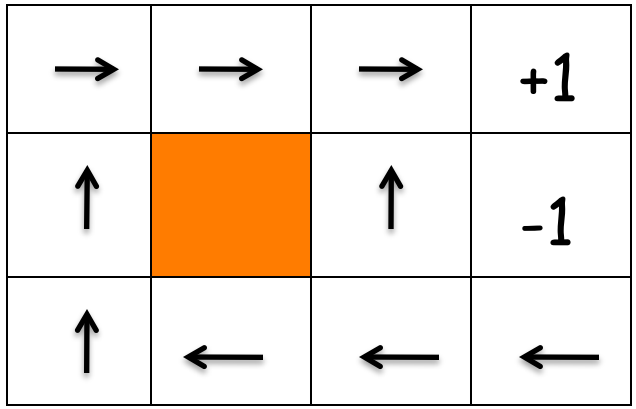
T(s,a,s')

Up = 0.8 up 0.1 left 0.1 right
Left = …
Right = …
Down = …

Policy Example

- Goal of MDP is to find optimal policy $\pi: \; s_t \rightarrow a_t$

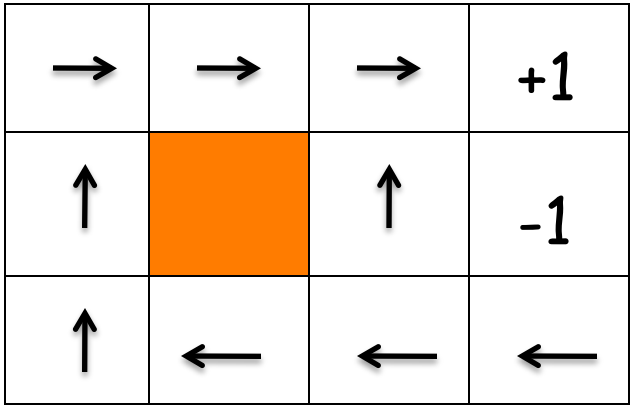
e.g.



Example

- Robot navigating on the grid - up, down, left, right
- Reward +1 for reaching the goal, -1 for going to (4,2)
- R(s) = -0.04 small negative reward for visiting non-goal states (penalize wandering around0
- Goal: find the policy sequence of actions
- Solution

e.g.



- Utility of the sequence – total payoff $0 \le \gamma \le 1$

$$U(s_0,\cdots,s_n) = R(s_0) + \cdots + R(s_n)$$

$$U(s_0,\cdots,s_n) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$

Markov decision processes

- Infinite horizon – since there is infinite time budget the optimal action depends only at the current state
- Discounted rewards are preferable
- Goal how to chose among policies
- Note that given policy it generates not one state sequences but whole set of them each with some probability determined by the transition model
- Hence value of a policy is expected sum of (discounted rewards)

$$E[\sum_{t=0}^{\infty} \gamma^t R_t(s_t) \mid \pi]$$

Types of rewards

- Reward structure: additive rewards

$$U(s_0, s_1, \ldots, s_n): \quad R(s_0) + R(s_1) + \ldots + R(s_n)$$

- Discounted rewards

$$U(s_0, s_1, \ldots, s_n): \quad R(s_0) + \gamma R(s_1) + \ldots + \gamma^n R(s_n)$$

- Preference for current rewards over future rewards (good model for human and animal preferences over time)
- How to deal with the infinite rewards ? Make sure that the utility of the infinite sequence is finite
- Design proper policies which are quaranteed to reach the final state
- Compare policies based on average reward per step

Utility of the state

- The goodness of the state is defined in terms of utility
- Utility of the state is expected utility of sequences which may follow that state

$$U^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^{t} R\ (s_t) \mid \pi, s_0 = s]$$

- Distinction between reward and utility
- Goal: Find the best policy (which will maximize expected utility)

$$\pi^{*}: \quad S \rightarrow \quad A$$

Calculate optimal policies

- Value iteration: algorithm for calculation utility of a state
- Given utility of a state under certain policy

$$U^{\pi}(s) = E(\sum_{t=0}^{\infty} \gamma^t R(s_t))$$

- Reward in current state + value function for next state

$$U^{\pi}(s) = E[R(s) + \gamma(R(s_1) + \gamma R(s_2) + \cdots)]$$

- Bellman equation (rewrite above in a recursive manner)

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} T(s,a,s')U(s')$$

- Reward in current state + (sum over all possible next states (probability of going to next state s' when action a is executed * utility of that next state)

Optimal Payoff

- Belman equation: set of linear constraints

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} T(s,a,s')U(s')$$

- One equation per state, n states n equations, solve for U
- Find such policy which maximizes the payoff

$$U^{*}(s) = \max_{\pi} U^{\pi}(s)$$

- We know how to compute value function
- How to compute optimal policy – there are exponentially many sequences of actions

Optimal Policy

- Optimal payoff – choose action which maximizes the future utility

$$U^*(s) = R(s) + \arg\max_a \sum_{s'} T(s,a,s')U^*(s')$$

- Then the definition of optimal policy is

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')U(s')$$

- Always choose an action which maximizes expected utility of the next state

- Next Value of a state and Utility of a state are used interchangably

Value iteration

- Compute the optimal value function first, then the policy
- N states – N Bellman equations, start with initial values, iteratively update until you reach equilibrium

1. Initialize V; $U(s) = 0$
2. For each state x

$$U'(s) = R(s) + \gamma \max_a \sum_{x'} T(s,a,s')U(s')$$

3. If $\left| U'(s) - U(s) \right| > \delta$ then $\delta \leftarrow \left| U(s') - U(s) \right|$
4. until

$$\delta < \varepsilon(1-\gamma)/\gamma$$

- Optimal policy can be obtained before convergence of value iteration

Policy Iteration

- Alternative Algorithm for finding optimal policies
- Takes policy and computes its value
- Iteratively improved policy, until it cannot be further improved

  1. Policy evaluation – calculate the utility of each state under particular policy π_i

  2. Policy improvement – Calculate new MEU policy, using one-step look-ahead based on π_{i+1}

  1. Initialize policy

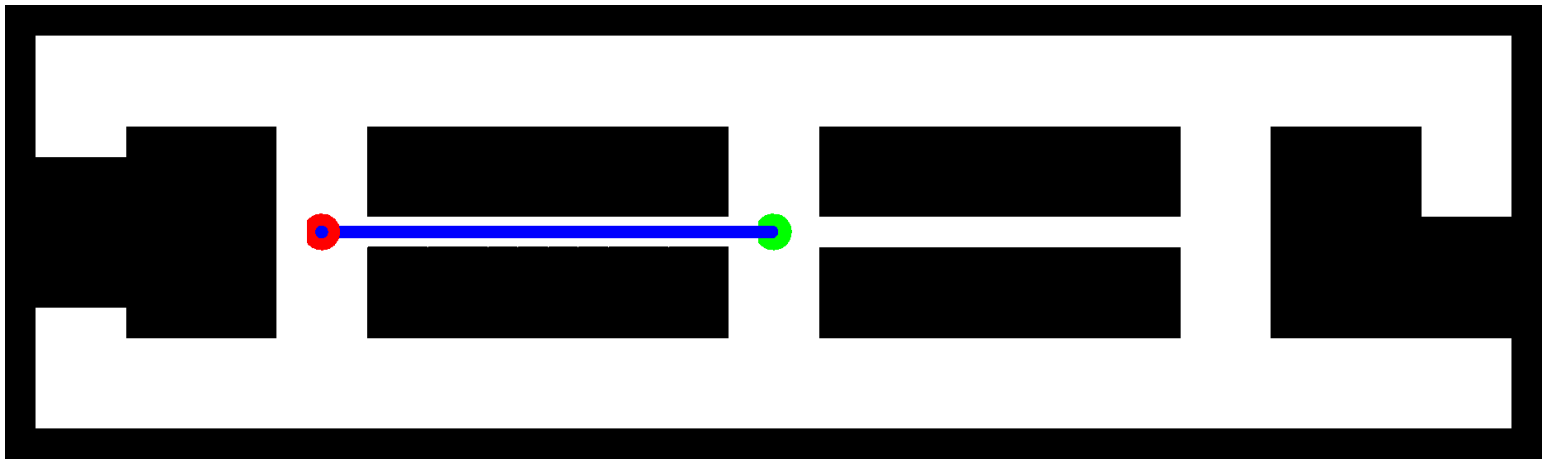
  2. Evaluate policy get V; For each state do if

$$\max_a \sum_{x'} T(s,u,s')U(s') > \sum_{s'} T(s,\pi(s),s')U(s')$$

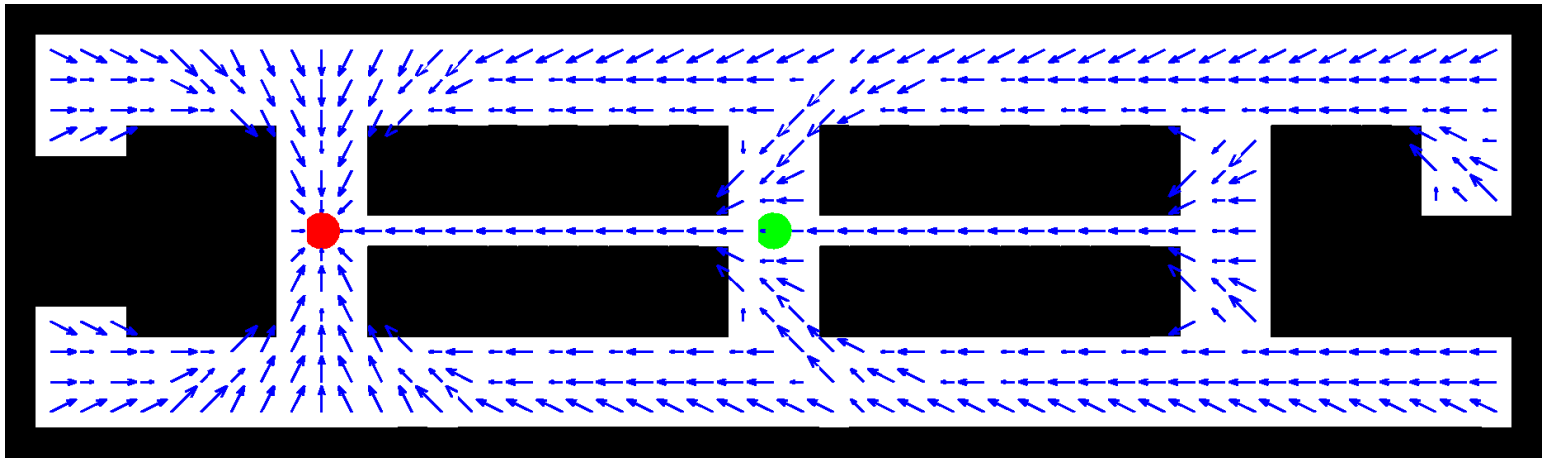
$$\pi(s) \leftarrow \arg\max_a \sum_{x'} T(s,u,s')U(s')$$

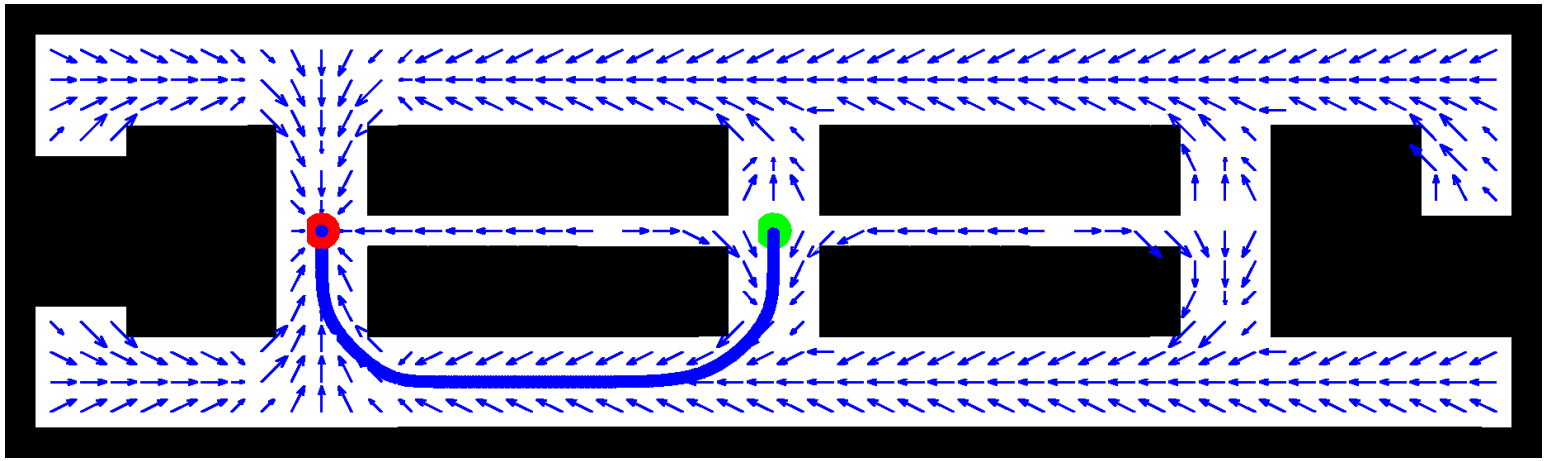
- Until unchanged

Deterministic, fully observable



Stochastic, Fully Observable





Stochastic, Partially Observable