CS 440
Introduction to Artificial Intelligence

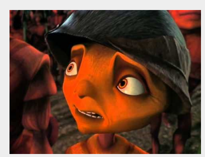
Lecture 19:

Partially Observable Markov Decision Processes

March 31, 2020

Hidden Markov Model Example 2

- **Food at corner of grid**
  - **Don't know where food is**
  - **And has .3 probability of moving towards food and .1 probability of moving away**

		.1		
	.1		.3	
		.3		
				

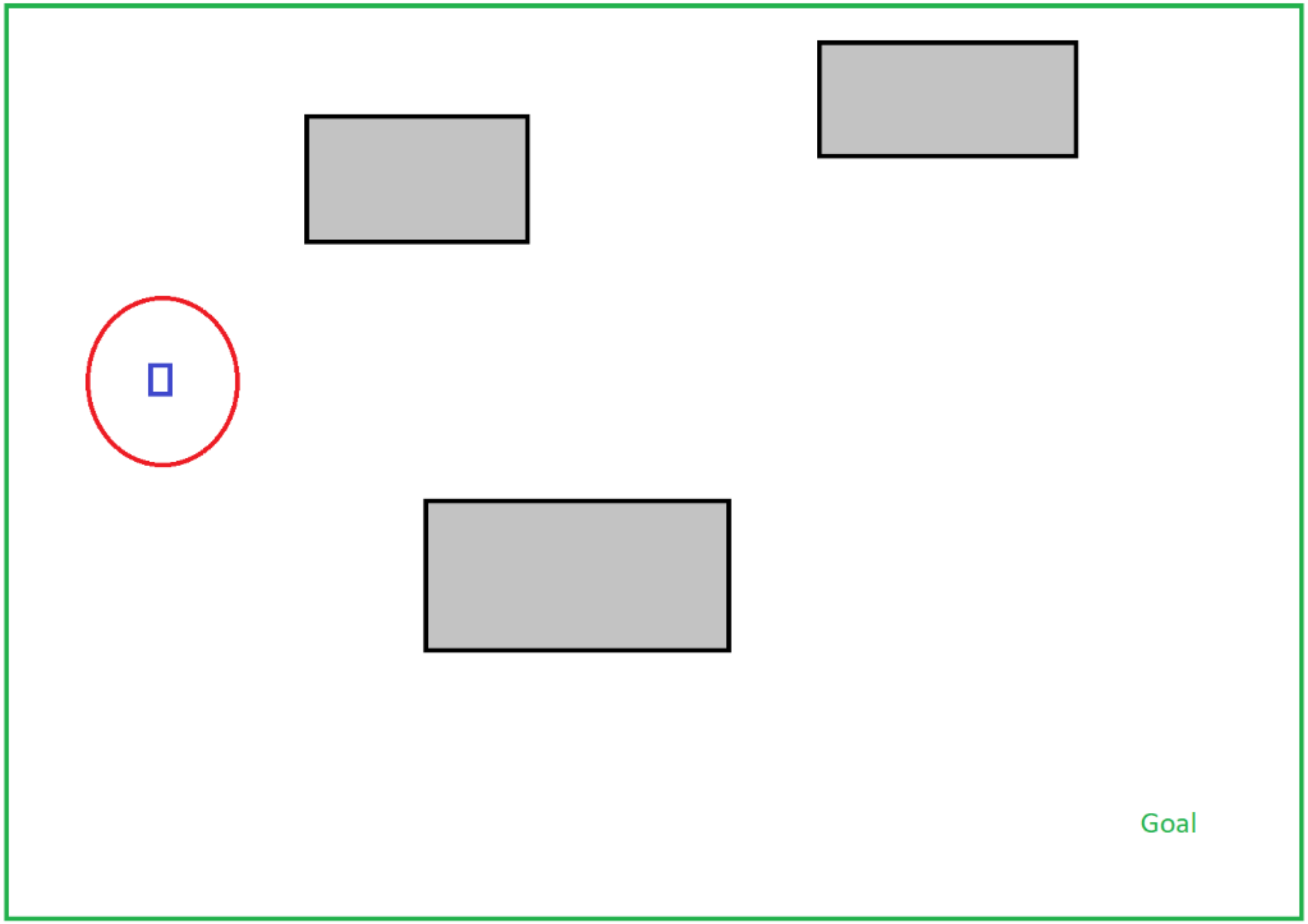
POMDP

- **Agent cannot directly observe state of environment**
  - **Observations agent makes are limited and noisy**
- **Agent's actions can influence state of environment**
- **Results of actions not deterministic**
  - **Defined by probability distribution**

Formalization

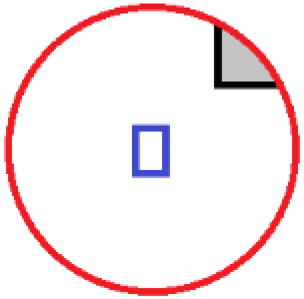
- State space S
  - Agent does not know what state it is in
- Set of actions A
  - Actions can be noisy
  - Results of actions a probability distribution over other states
- Transition function T(s,a,s')
  - T(s,a) result of taking action a while in state s
    - Noisy actions:
      - T(s,a) is a probability distribution over state space
      - $T(s,a) = \{p(s'_1), p(s'_2), \ldots, p(s'_n)\}$
        - where $p(s'_i)$ is the probability that performing action a while in state s will result in state s'
    - Defined for all combinations of $s \in S$, $a \in A$
- Reward function R
- Immediate objective: Agent must determine what action to take in order to maximize future expected reward

Example:Robot with limited sensor range

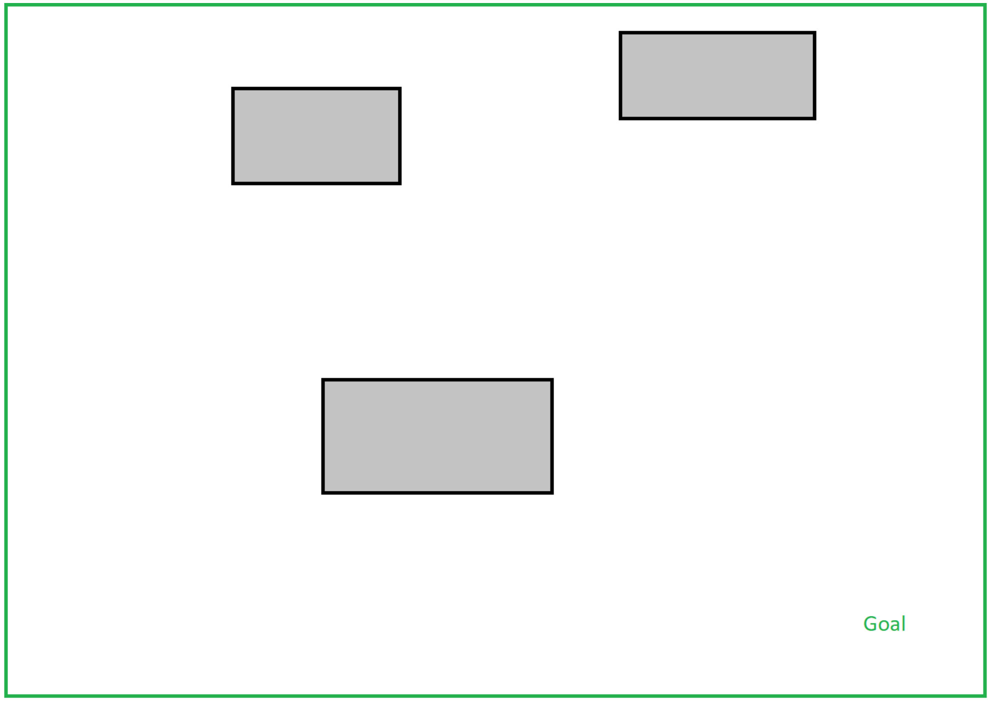


- **Robot (blue) can only see region inside of red circle**
  - **Needs to localize**
  - **Needs to find path to goal**

Example:Robot with limited sensor range



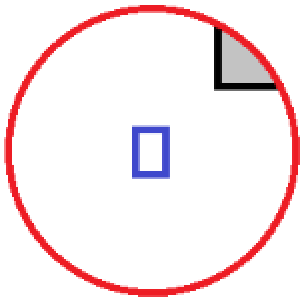
What robot sees



environment

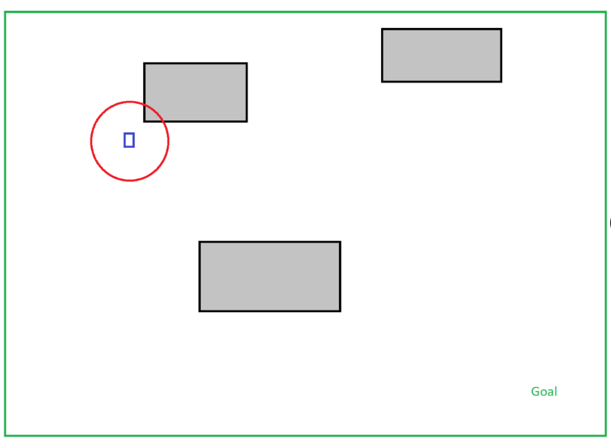
Where is the robot located?

Example:Robot with limited sensor range

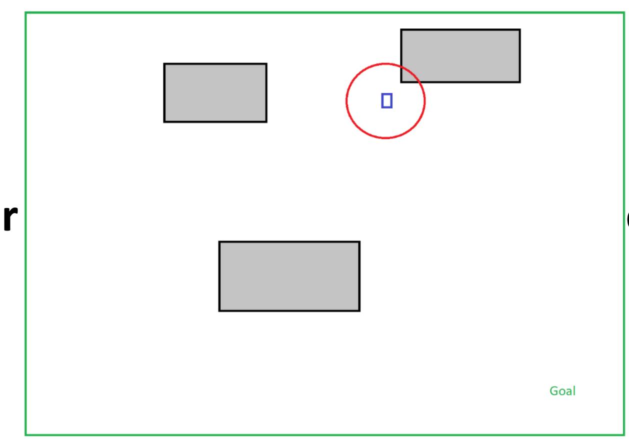


What robot sees

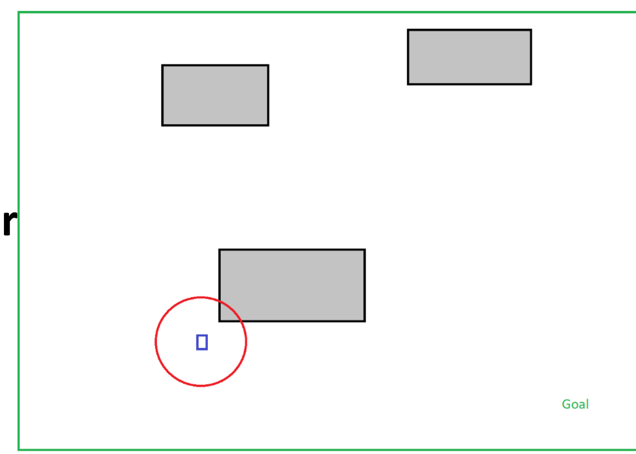
Where is the robot located?



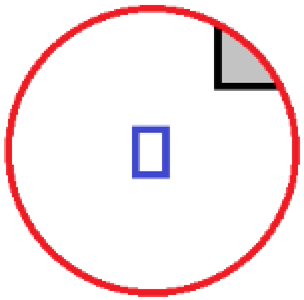
or



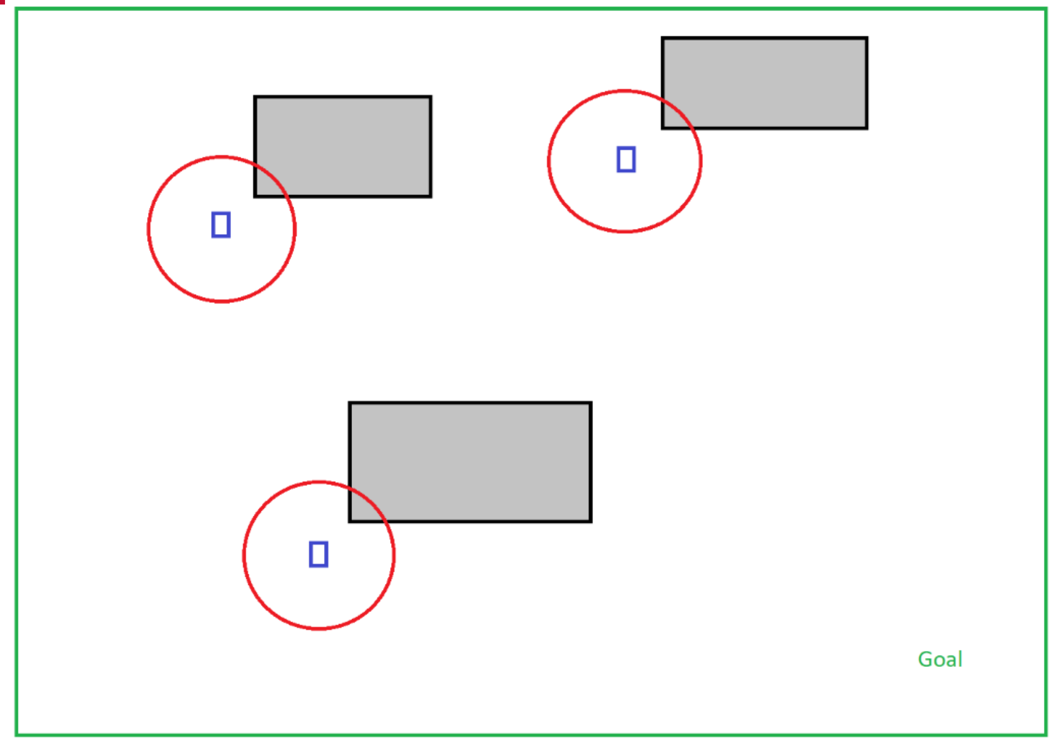
or



Example:Robot with limited sensor range



What robot sees

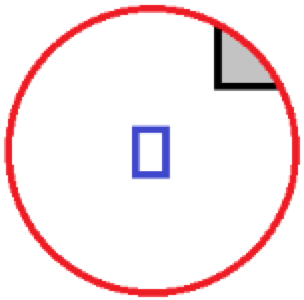


What move should the robot make?

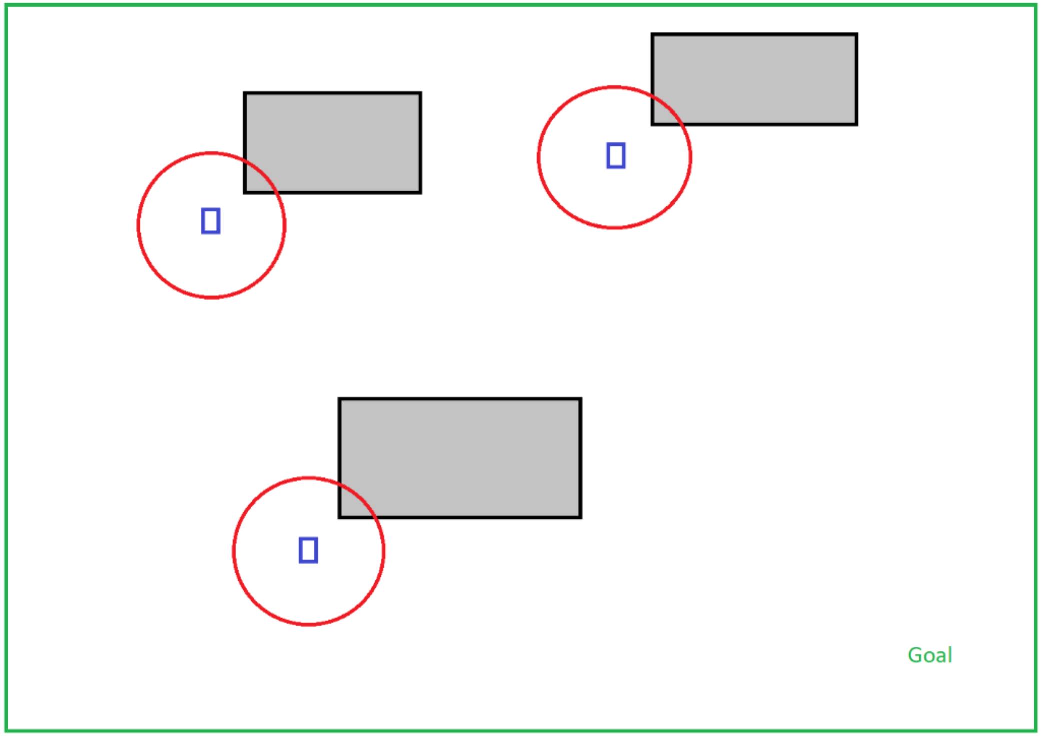
Belief

- Belief defined as a probability distribution over state space
- $S = \{s_1, s_2, \ldots, s_n)$
- $b = \{p(s_1), p(s_2), \ldots, p(s_n)\}$
  - $p(s_i)$ is the agent's estimate of the likelihood it is in state s_i
- For discrete state space consists of a probability for each state
  - Often times probability of most states will be 0
- For non-discrete problem consist of a probability density function
  - Example: Gaussian

Example:Robot with limited sensor range



What robot sees

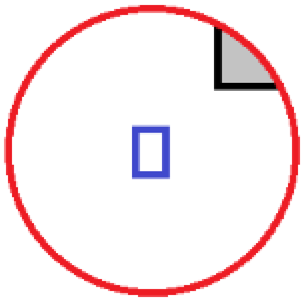


What would belief be here?

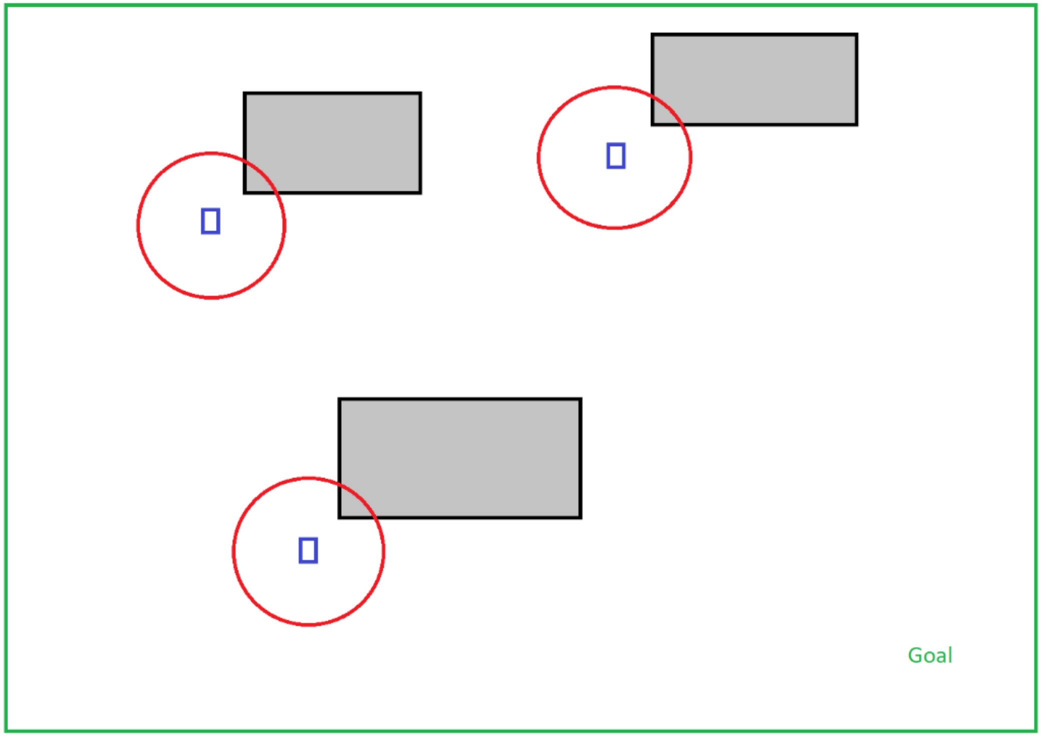
Exploration vs Exploitation

- Exploration
  - Gain information about environment
  - Reduce uncertainty of belief
- Exploitation
  - Find a solution to the problem
    - Reach the goal
    - Get to a high reward state
- A solution to the pomdp needs to balance exploration and exploitation

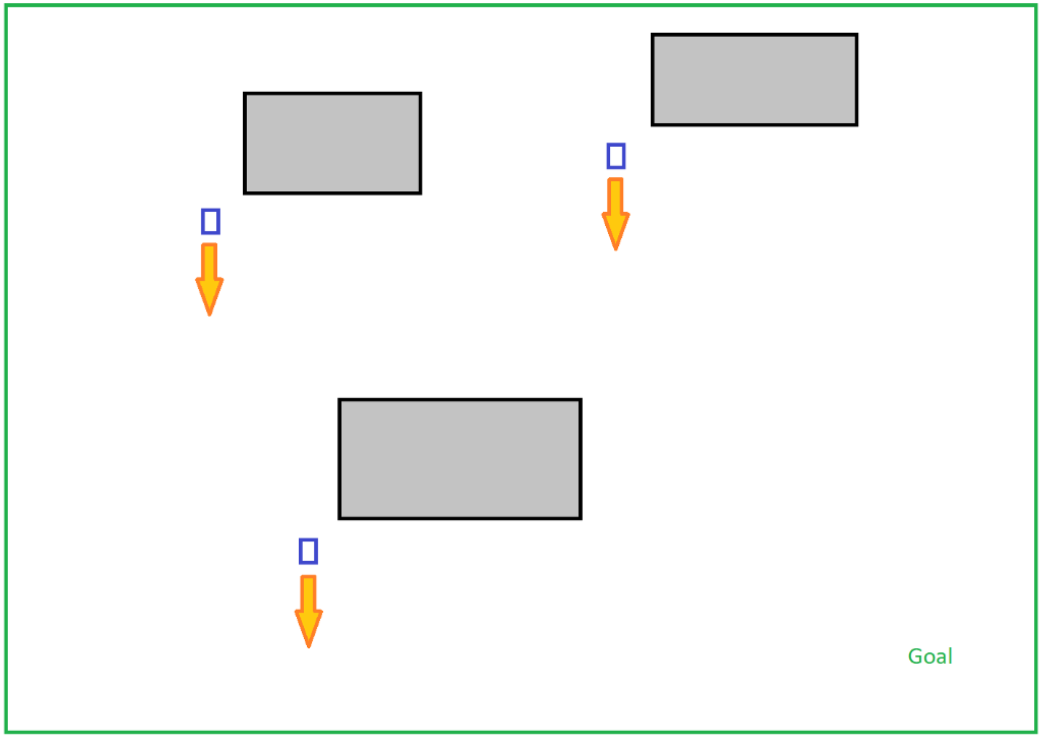
Example:Robot with limited sensor range



What robot sees

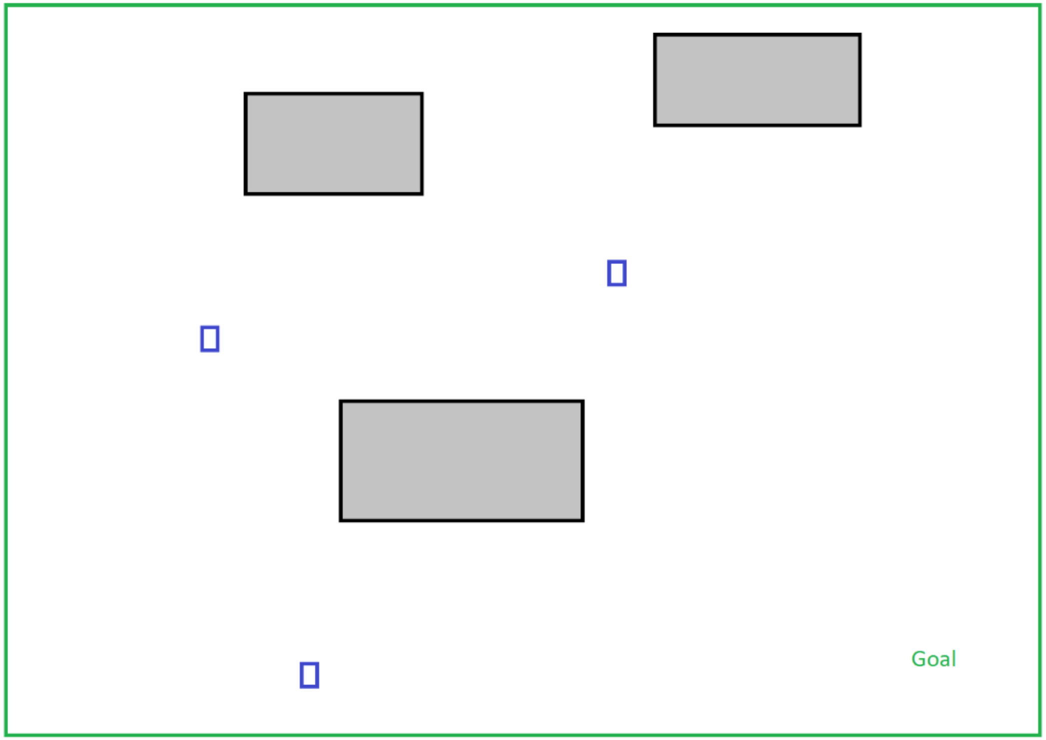


Example:Robot with limited sensor range

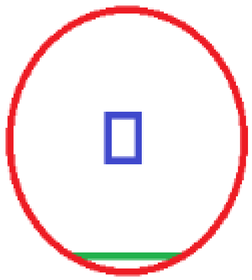


RUTGERS

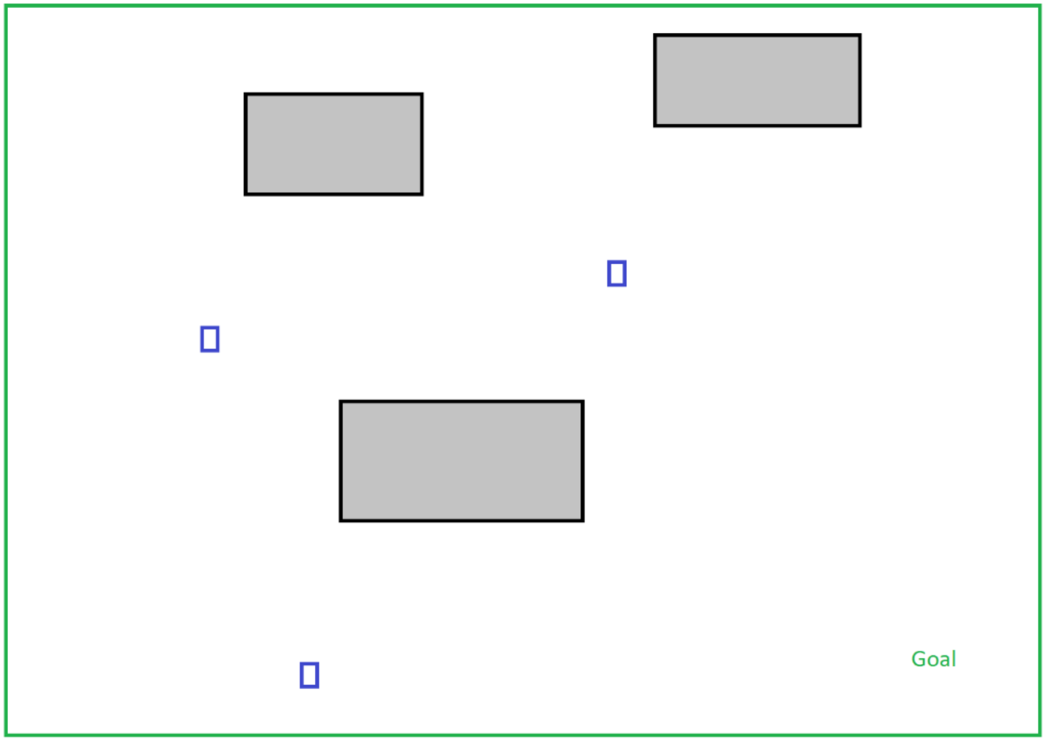
Example:Robot with limited sensor range



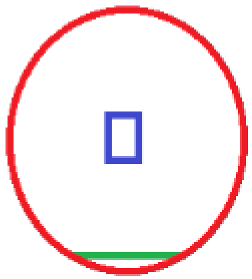
Example:Robot with limited sensor range



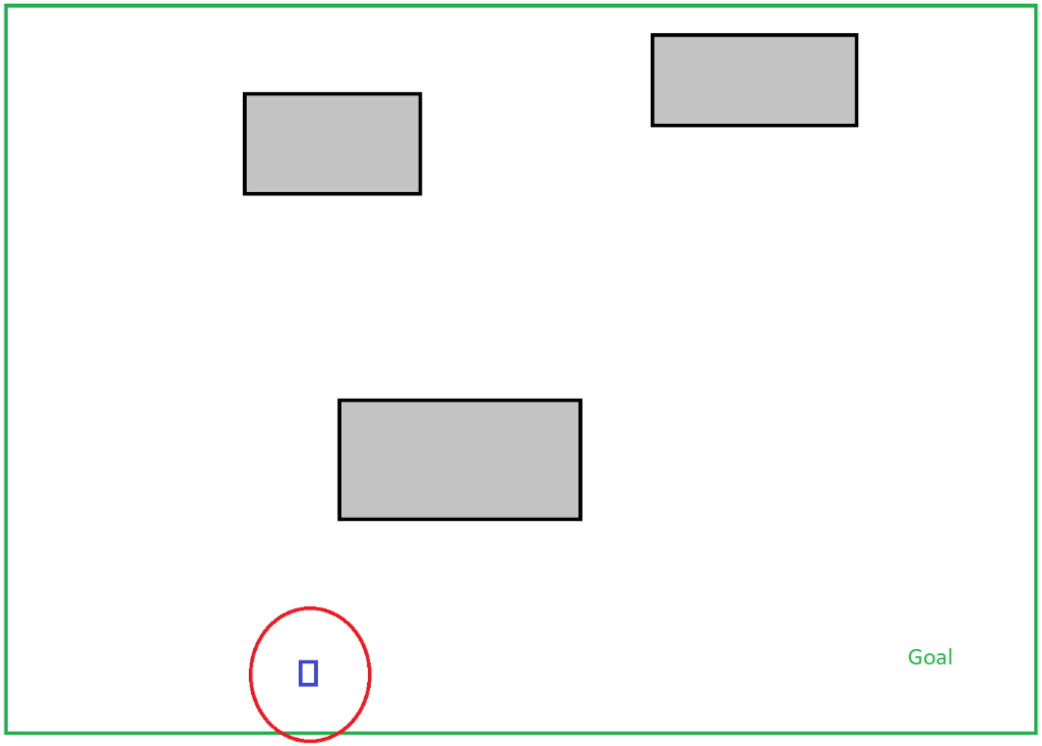
What robot sees



Example:Robot with limited sensor range



What robot sees



Solution to POMDP

- Policy
  - For MDPs a policy was a mapping of states to actions
  - For PRMDPs a policy is a mapping of beliefs to actions
    - Belief: Probability distribution over states
    - $\Pi(b) \Rightarrow a$
    - Action the agent should perform given its belief

POMDP Solution

	Step i-1	Step i	Step i+1
Observation	o_{i-1}	o_i	o_{i+1}
Action	a_{i-1}	a_i	a_{i+1}
Belief	b_{i-1}= $p(x_{i-1,0}),p(x_{i-1,1}),\ldots$	b_i= $p(x_{i,0}),p(x_{i,1}),\ldots$	b_{i+1} $p(x_{i+1,0}),p(x_{i+1,1}),\ldots$

- If we know what actions the agent will take we can compute in same manner as with hidden Markov model
- Intermediate step
  - $p(x_i') = \sum_{x_{i-1} \in X} p(x_{i-1}) * p(x_i' | x_{i-1}, a_{i-1})$
    - $p(x_i' | x_{i-1}, a_{i-1})$ = transition probability given action a
- $p(x_i) = \sum_{x_i' \in X} p(x_i') * p(x_i | x_i', o_i) \;/\; \sum_{x_i' \in X} p(o_i | x_i')$
  - Bayes theorem
- Compute for all combinations of actions to find best action?
  - Computationally infeasible
    - Branching factor A^n where n is the number of steps you need to search
- Use heuristics to decide what action sequences to consider
  - Open problem