

---

# Announcements

---

- ❖ Mid-term exam: June 25, 12:10pm-1:50pm
  - ❖ Closed book, 2 A4-sized cheatsheets
  - ❖ No electronic device, no communication
- ❖ HW5 on CSP
  - ❖ Early release today
  - ❖ Due June 30 at 11:59pm
- ❖ P3 on MDP and RL
  - ❖ Released later today
  - ❖ Due July 5 at 11:59pm

# Ve492: Introduction to Artificial Intelligence

## Mid-term Review



Paul Weng

UM-SJTU Joint Institute

Slides adapted from <http://ai.berkeley.edu>, AIMA, UM, CMU

# What have we learned so far?

- ❖ Search and planning

- ❖ Define a state space, goal test; Find path from start to goal

- ❖ Game trees

- ❖ Define utilities; Find path from start that maximizes utility

- ❖ Decision theory and game theory

- ❖ Foundation for MEU; Basic concepts in game theory

- ❖ MDPs

- ❖ Define rewards, utility = (discounted) sum of rewards
  - ❖ Find policy that maximizes utility

- ❖ Reinforcement learning

- ❖ Just like MDPs, only T and/or R are not known in advance

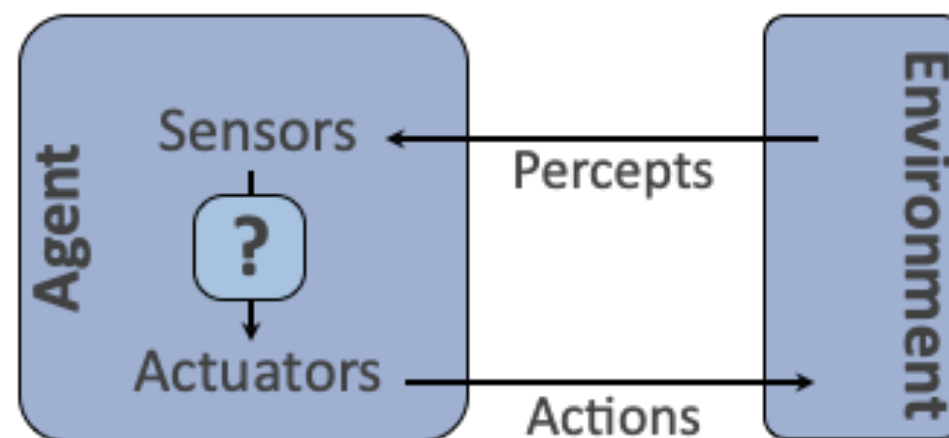
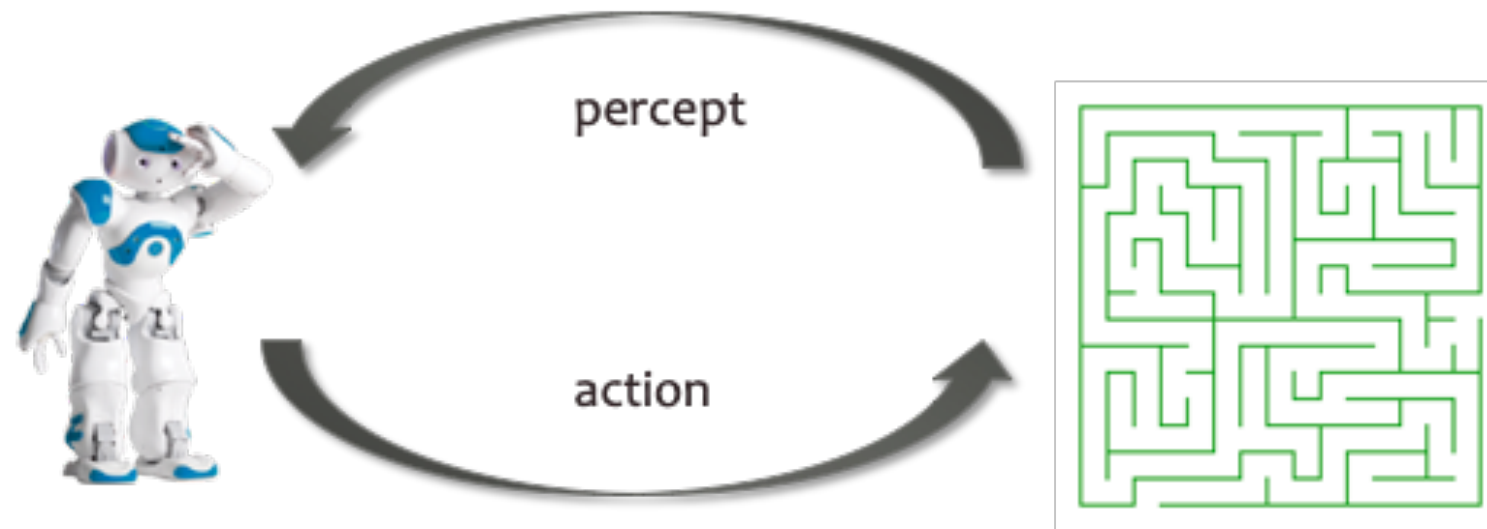
- ❖ Constraint satisfaction

- ❖ Find solution that satisfies constraints; Not just for finding a sequential plan



# High-Level Framework

- ❖ How to build AI system?



---

# Search

---

- ❖ Environment: single-agent, fully-observable state, deterministic transition, sequential, model known
- ❖ Search problem
  - ❖ States, transition model, goal test, initial state
  - ❖ Search tree
- ❖ Algorithms
  - ❖ Uninformed search
    - ❖ BFS, DFS, UCS
  - ❖ Informed search
    - ❖ Greedy search, A\*
- ❖ Properties
  - ❖ Complete, optimal
  - ❖ Space and computational complexities

# Search in Games

---

- ❖ Environment: multi-agent, fully-observable state, deterministic or stochastic transition, turn-taking, model known
- ❖ Multi-agent search problems as games
  - ❖ States, players, transition model, terminal test/ values, initial state
  - ❖ Game tree
- ❖ Algorithm for adversarial agent (zero-sum game)
  - ❖ Minimax search algorithm
  - ❖ Alpha-beta pruning
  - ❖ Depth-limited search, iterative deepening
- ❖ Algorithm for random agent
  - ❖ Expectimax
- ❖ Algorithm for multi-agent search
  - ❖ Expectiminimax

# Decision Theory and Game Theory

---

- ❖ Axiomatization of Expected Utility
  - ❖ Completeness, Transitivity, Independence, Continuity
  - ❖ Unicity of utility function up to positive affine transformation
  - ❖ Preference elicitation
- ❖ Game theory
  - ❖ Extensive form vs normal form
  - ❖ Best response, dominant/ dominated strategies
  - ❖ Nash equilibrium (pure or mixed)
  - ❖ Pareto optimal, correlated equilibrium

# Markov Decision Process

---

- ❖ Environment: single-agent, fully-observable state, stochastic transition, sequential, model known
- ❖ Model
  - ❖ States, actions, transition function, reward function
- ❖ Algorithms
  - ❖ Policy evaluation
  - ❖ Policy extraction
  - ❖ Value iteration
  - ❖ Policy iteration



# Reinforcement Learning

---

- ❖ Environment: single-agent, fully-observable state, stochastic transition, sequential, model unknown
- ❖ MDP Model, but unknown!
  - ❖ States, actions, transition function, reward function
- ❖ Algorithms
  - ❖ Policy evaluation with TD learning
  - ❖ Policy learning with Q-learning
  - ❖ Approximate Q-learning
  - ❖ Action selection with  $\epsilon$ -greedy or exploration function

# Constraint Satisfaction

---

- ❖ CSP

- ❖ Set of variables, set of domains, set of constraints
- ❖ Find assignments to variables such that all constraints are satisfied

- ❖ Algorithms

- ❖ Backtracking search
  - ❖ Filtering, forward-checking, arc consistency, k-consistency
  - ❖ Ordering of variables and values
- ❖ Structure of constraint graph
  - ❖ Two-pass algorithm for tree-structured constraint graph
  - ❖ Cutset conditioning
- ❖ Iterative improvement

- ❖ Local search

# Quiz: Search

❖ Consider a graph search problem where for every action, the cost is at least  $\epsilon$ , with  $\epsilon > 0$ . Assume the used heuristic is consistent.

 ❖ Greedy graph search is guaranteed to return an optimal solution.

 ❖ A\* graph search is guaranteed to return an optimal solution.

 ❖ A\* graph search is guaranteed to expand no more nodes than depth-first graph search.

 ❖ A\* graph search is guaranteed to expand no more nodes than uniform-cost graph search.

# Quiz: A\* Heuristics

❖ Let  $H_1$  and  $H_2$  both be admissible heuristics.

⌊ ❖  $\max(H_1, H_2)$  is necessarily admissible

⌊ ❖  $\min(H_1, H_2)$  is necessarily admissible

⌊ ❖  $(H_1 + H_2)/2$  is necessarily admissible

✱ ❖  $\max(H_1, H_2)$  is necessarily consistent



# Quiz: Search under Uncertainty

- ❖ You are given a game tree for which you are the maximizer, and in the nodes in which you don't get to make a decision an action is chosen uniformly at random amongst the available options. Your objective is to maximize the probability you win \$10 or more (rather than the usual objective to maximize your expected value).
- F ❖ Running expectimax will result in finding the optimal strategy to maximize the probability of winning \$10 or more.
- F ❖ Running minimax, where chance nodes are considered minimizers, will result in finding the optimal strategy to maximize the probability of winning \$10 or more.
- T ❖ Running expectimax in a modified game tree where every pay-off of \$10 or more is given a value of 1, and every pay-off lower than \$10 is given a value of 0 will result in finding the optimal strategy to maximize the probability of winning \$10 or more.
- F ❖ Running minimax in a modified game tree where every pay-off of \$10 or more is given a value of 1, and every pay-off lower than \$10 is given a value of 0 will result in finding the optimal strategy to maximize the probability of winning \$10 or more.

---

# Quiz: Adversarial Search

---

- ❖ In the context of adversarial search,  $\alpha$ - $\beta$  pruning
  - ❖ can reduce computation time by pruning portions of the game tree
  - ❖ is generally faster than minimax, but loses the guarantee of optimality
  - ❖ always returns the same value as minimax for the root of the tree
  - ❖ always returns the same value as minimax for all nodes of the tree





# Game Theory: Zero-Sum Game

- ❖ Two players choose simultaneously a coin of 10 cents, 50 cents or 1 dollar, which they show to each other.
- ❖ If they chose the same coin, player I wins. Otherwise, player II wins.
- ❖ Write this game in normal form. Is there any pure NE? *N*
- ❖ Express a system of inequalities to find a mixed NE.



# Quiz: MDP

❖ For Markov Decisions Processes (MDPs), we have that:

-  ❖ A small discount (close to 0) encourages shortsighted, greedy behavior.
-  ❖ A large, negative living reward ( $\ll 0$ ) encourages shortsighted, greedy behavior.
-  ❖ A negative living reward can always be expressed using a discount  $< 1$ .
-  ❖ A discount  $< 1$  can always be expressed as a negative living reward.



# Quiz: MDP

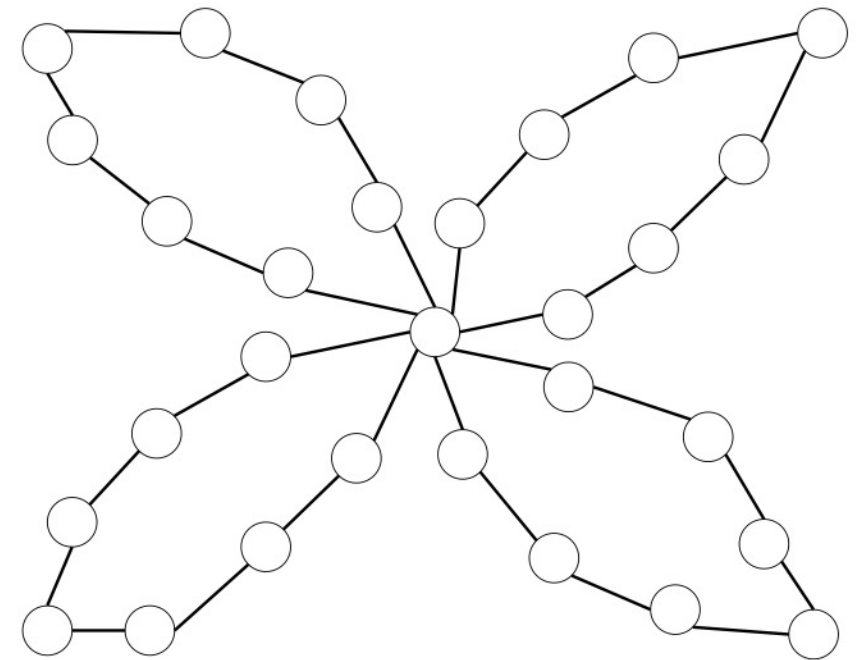
- ❖ Value iteration can converge only if the discount factor ( $\gamma$ ) satisfies  $0 < \gamma < 1$ .
- ❖ Policies found by value iteration may be superior to policies found by policy iteration.
- ❖ Policies found by policy iteration may be superior to policies found by value iteration.
- ❖ In some problems, value iteration can converge even though policy iteration may not.

# Quiz: Reinforcement Learning

- ❖ Assume that the agent observes the true reward with some Gaussian noise  $\mathcal{N}(0,1)$ , Q-learning would still converge
- ❖ Q-learning can learn the optimal Q-function  $Q^*$  without ever executing the optimal policy.
- ❖ If an MDP has a transition model  $T$  that assigns non-zero probability for all triples  $T(s, a, s')$  then Q-learning will fail.
- ❖ In Q-learning, we decide to explore every  $k$  steps, i.e., if  $t = 0 [k]$  we choose a random action with a uniform distribution, otherwise we choose the greedy action. This version would still converge.

# Quiz: CSP

- ❖ Assume given a CSP whose constraint graph is given below and that all the variables have the same domain.
- ❖ What is the complexity of solving it with a direct application of backtracking search?
- ❖ Which efficient strategy could you apply to solve it? What would be the complexity?



# CSP Problem: Job Scheduling

## ❖ When can I move in?

Task	Description	Duration	Predecessor
a	Erecting walls	7	none
b	Carpentry for roof	3	a
c	Roof	1	b
d	Installations	8	a
e	Facade painting	2	c & d
f	Windows	1	c & d
g	Garden	1	c & d
h	Ceilings	3	a
i	Painting	2	f & h
j	Moving in	1	i