

CS 121

Course Review

Chris Archibald
August 14, 2009

Announcements

- ▶ Homework 6 problems due now
- ▶ Solutions should be posted later today
- ▶ Homework 6 programming due Wednesday at midnight
- ▶ The Final will be Friday, August 14, 2009 from 12:15 - 3:15 in this room
- ▶ Practice problems were released this morning on Coursework

Purpose

- ▶ Today we will review (very quickly) the topics that we have covered this quarter
- ▶ The goal today is to remind you of all the topics that could be on the final
- ▶ If you have questions about a specific area, or problem we have seen, ask!
- ▶ These slides will hopefully provide a good study resource

Outline

- ▶ Agents and Environments
- ▶ Search Problems
- ▶ Blind Search
- ▶ Informed Search
- ▶ Local Search
- ▶ Constraint Satisfaction Problems
- ▶ Action Planning
- ▶ Adversarial Search
- ▶ Non-deterministic Uncertainty
- ▶ Probabilistic Uncertainty
- ▶ Game Theory
- ▶ Learning
- ▶ Motion Planning

Agents

- ▶ Focus on designing **rational agents**
- ▶ PEAS
 - ▶ P = Performance measure
 - ▶ E = Environment
 - ▶ A = Actuators
 - ▶ S = Sensors

Environment Types

- ▶ Observable?
- ▶ Deterministic?
- ▶ Episodic?
- ▶ Static?
- ▶ Discrete?
- ▶ Single-agent?

Search

- ▶ We formulate a **goal** for the agent
- ▶ We formulate the search problem
 - ▶ States
 - ▶ Actions
- ▶ Find a solution = **Search**

Basic search environment

- ▶ Typically assume that the environment is:
 - ▶ Observable, deterministic, static, discrete and single-agent
- ▶ Later in the class we relaxed some of these assumptions

Search problem

A **search problem** is defined by the following:

- ▶ State space
- ▶ Initial state
- ▶ Successor function
- ▶ Goal test
- ▶ Path cost

Search problem formulation

- ▶ Problem must be abstracted
- ▶ No one right way to do this
- ▶ Formulation can have big impact on feasibility or optimality of resulting solution

Configuration space

- ▶ n -dimensional space that describes completely the robot's configuration
- ▶ We must map the obstacles into this space
- ▶ Then we search for an obstacle-free path from start state to goal state

Probabilistic Road-Map

- ▶ Configuration spaces are continuous and can be high-dimensional
- ▶ How to handle?
- ▶ Probabilistic road maps:
 - ▶ Randomly sample points in the configuration space
 - ▶ Test for obstacles
 - ▶ Join points to other points
 - ▶ This gives us a skeletonization of the space
 - ▶ Connect start and goal states, now we have a normal search problem
- ▶ Very influential and widely-used approach and idea

Blind Search

- ▶ Blind search is the problem of searching given only the search problem formulation
- ▶ Basic approaches:
 - ▶ Breadth-first : Complete, optimal, large space / time
 - ▶ Depth-first : Incomplete, suboptimal, less space, larger time
 - ▶ Iterative-deepening search: Complete, optimal, less space, better time
 - ▶ Bidirectional search : Can be used with BFS
 - ▶ Uniform cost search: Same as BFS but in the cost space, instead of step space

Informed search

- ▶ Informed search orders nodes in fringe by an evaluation function
- ▶ There are different evaluation functions we can use
- ▶ $f(n) = g(n) \Rightarrow$ Uniform-cost search
- ▶ $f(n) = h(n) \Rightarrow$ Greedy-best-first search
- ▶ $f(n) = g(n) + h(n) \Rightarrow$ A* search

Heuristic functions

- ▶ Arbitrary function which estimates distance to the goal
- ▶ $h(n) = 0$ if n is a goal node
- ▶ **Admissible** heuristics never overestimate the distance to the goal
- ▶ **Consistent** heuristics satisfy the triangle inequality
- ▶ Heuristics are often solutions to a simplified version of the problem
- ▶ If $h_1(n) \geq h_2(n)$ then h_2 **dominates** h_1

A* search

- ▶ Complete
- ▶ Optimal
- ▶ With a consistent heuristic then nodes are expanded in non-decreasing f -value
- ▶ With consistent heuristic dominating heuristic always does better than dominated (i.e. expands fewer nodes)
- ▶ A* is **optimally efficient** = no optimal algorithm can expand fewer nodes
- ▶ Extensions of A*
 - ▶ IDA*
 - ▶ Memory bounded A*

Local Search

- ▶ When the path to the goal doesn't matter, but only the goal reached, then local search is a good alternative
- ▶ Local search works on a complete state representation
- ▶ Successor function generates “neighboring” complete states
- ▶ Objective function tells us how good a complete state is
- ▶ Generally try to go to best neighboring state
- ▶ Can be very effective when state space is large

Local search algorithms

- ▶ Hill-climbing
- ▶ Stochastic hill-climbing
- ▶ First-choice hill-climbing
- ▶ Random-restart hill-climbing
- ▶ Simulated annealing
- ▶ Local beam search
- ▶ Stochastic beam search
- ▶ Genetic algorithms

Constraint Satisfaction Problems

- ▶ Many search problems have **constraints** between the variables in the state
- ▶ We need a way to explicitly represent the constraints, and a method of dealing with them

CSP

- ▶ Set of **variables** $\{X_1, X_2, \dots, X_n\}$
- ▶ Each variable X_i has a (finite) **domain** D_i of possible values
- ▶ Set of **constraints** $\{C_1, C_2, \dots, C_p\}$
- ▶ Each constraint relates a subset of variables by specifying the valid combinations of their values
- ▶ **Goal:** assign a value to every variable such that all constraints are satisfied

Backtracking algorithm

- ▶ Pick a variable X
- ▶ Pick an ordering on the domain of X
- ▶ For each value of X
 - ▶ Assign that value to X
 - ▶ Recurse on the remaining variables

Forward checking

- ▶ Remove inconsistent values from other domains
- ▶ If any domain is empty, inconsistent assignment

Heuristics in CSPs

- ▶ Major impact of variable decision and value ordering
- ▶ Heuristics to use:
 - ▶ Most-constrained variable : smallest remaining domain
 - ▶ Most-constraining variable : break ties by variable tied to most others
 - ▶ Least-constraining value: Most likely to lead to valid assignment

AC-3

- ▶ Algorithm for checking that domains are consistent with each other
- ▶ For each variable X , check that all variables Y connected with X and ensure that for each value in Y 's domain, there is a value in X 's domain that is valid with it. Otherwise remove value. Reinsert Y into queue if removal occurred.

Action planning

- ▶ STRIPS representation
- ▶ States, goals, actions
- ▶ Actions have precondition, add list, delete list

Action planning

- ▶ Forward planning, planning graphs
- ▶ Backward planning, regression

Adversarial search

- ▶ Now consider opponent
- ▶ Game tree where a player acts at each node
- ▶ Basic algorithm works from leaves backward
- ▶ When game is **zero-sum** then this is called: Minimax algorithm
- ▶ We assume that our opponent is rational, and knows that we are rational, etc.

Alpha-beta pruning

- ▶ Often game trees are large
- ▶ Alpha-beta pruning ignores portions of search tree that cannot affect decision
- ▶ Basic idea: keep track of each player's best option and if an option is worse, ignore rest of that option
- ▶ Key point: Alpha-beta pruning will make exactly the same decision as minimax, just looks at fewer nodes
- ▶ We can apply similar ideas to games with chance nodes

Non-deterministic uncertainty

- ▶ Each action returns a set of possible resulting states
- ▶ No additional information about which is more likely or less likely
- ▶ Need to ensure that goal is reached

And-Or Tree

- ▶ Action nodes and state nodes
- ▶ Label tree from leaves backward
 - ▶ State is solved if it is a goal state or if one of successors is solved
 - ▶ Action node is solved if all of its successors are solved
- ▶ OR sub-tree corresponds to a path in a classical search tree
- ▶ Careful treatment of revisited states

Uncertainty in sensing

- ▶ Now we have a **belief state**, which is the set of all states we consider possible
- ▶ We now search the space of belief states, instead of the space of states
- ▶ An agent knows that something is true when it is true in each state of his belief state
- ▶ Sensory actions: agent chooses when to make observations

Probabilistic uncertainty

- ▶ Use probability to say something about how likely certain states are and what the results of actions will be
- ▶ Markov Decision Process (MDP)
 - ▶ Agent gets rewards in certain states
 - ▶ Some states are terminal
 - ▶ Goal is to choose a **policy** that will maximize the expected amount of rewards collected
 - ▶ Markov assumption: The results of an action do not depend on what happened before reaching the state where that action was taken

MDP approaches

- ▶ **Value iteration**: initialize each state with a value, update values using best action available, repeat until utilities converge
- ▶ **Policy iteration**: compute utility of each state given policy, then compute optimal policy given utility, if no change, then we are done
- ▶ Discount factor measures how much we prefer short term rewards to long-term ones

Bayesian networks

- ▶ Probabilistic belief state is distribution over all states that an agent thinks is possible
- ▶ Bayes' nets provides an explicit representation of the independencies of the situation, and take less space to encode than full belief states
- ▶ They still represent full belief states and can be used to answer any questions about the joint belief state, (e.g. given evidence)

Game theory

- ▶ Mathematical study of agent interaction
- ▶ Assume common knowledge of rational agents
- ▶ Normal-form games
- ▶ Pure strategies vs. mixed strategies

Game theory topics

- ▶ Pareto optimal outcomes
- ▶ Dominant strategies
- ▶ Best response
- ▶ Nash equilibrium: strategy profile in which each player's strategy is a best response to the other players

Other multiple agent topics

- ▶ Voting: Arrow's Impossibility theorem – no “fair” voting rule
- ▶ Auctions: different rules lead to different agent strategies
- ▶ Incentives are very important

Learning

- ▶ Types of learning
 - ▶ Supervised
 - ▶ Unsupervised
 - ▶ Reinforcement
- ▶ Supervised learning / Inductive learning
 - ▶ Given example inputs and outputs decide on a hypothesis h
 - ▶ Keep-It-Simple bias

Decision trees

- ▶ Used for learning a goal predicate from observable predicates
- ▶ Greedy algorithm
- ▶ Find attribute that maximizes information gain
- ▶ Repeat ...

Summary

- ▶ We have covered many topics
- ▶ Many more topics within AI, many more classes to explore

Other Stanford AI courses

222

Rational Agency
and Intelligent Interaction

224M

Multi-Agent
Systems

224N 224S 224U

Natural Language Processing
+ Speech Recognition and Synthesis

227B

General
Game Playing

227

Reasoning
Methods in AI

221

121

228

Structured
Probabilistic Models

226

Statistical Techniques
in Robotics

223A 225A 225B

Intro. to Robotics + Experimental Robotics

229

Machine Learning

223B

Intro. to
Computer Vision

Final exam

- ▶ Final will be this friday, from 12:15 to 3:15 pm in this room
- ▶ One sheet of handwritten notes (both sides) allowed
- ▶ Final will be comprehensive
- ▶ If final grade is higher than midterm, I will replace your midterm grade
- ▶ On wednesday, half the class will be open for your questions
- ▶ Programming assignment due wednesday at midnight
- ▶ Good luck and see you then!