

Small aside: last time I mentioned NP-completeness in reference to playing chess. In fairness, I should point out that playing perfect chess is actually even harder than NP-complete (PSPACE-complete), though we won't talk about those subjects in detail for a while yet.

Reading for this lecture: Chapter 3

1 AI as Optimization

- Last time, we discussed only very high-level ideas about what constitutes AI or intelligence in general. Now we'll pin down our goals somewhat more specifically.
- In this class, we'll take an operational, "weak sense" view of AI
- Our goal, then, is not to produce fully "intelligent" systems, so much as systems that "do useful things"
- Overall framework: agent interacting with environment (the world)
- **pic of agent interface here**
 - robot
 - softbot
 - game player (chess, unreal tournament, etc)
 - spam filter
 - etc
- This view actually contains a lot of sub-parts, each of which is an entire sub-field of AI:
 - Perception: vision, speech recognition, sensor interpretation, etc.
 - actuation: translating internal commands to actions on the world (easy for softbots, hard for robots), could be simple as speakers to generate spoken text
 - representation: how to model world, different levels of resolution, etc.
 - inference: figuring out (a) what the current state of the world is or (b) what the state of the world *will be* after action
 - learning: developing model of (a) the world or (b) the effects of actions
- Our target, now: program agent to achieve some *goal*: perform series of actions that result in world attaining goal state
 - robot: deliver package, explore mars, defuse bomb, build car, etc.
 - chess 'bot: win game
 - spam filter: decide yes or no on msg
- more generally: want agent to "act well" continuously for indefinite time
 - robot: don't fall off cliff; don't run over people

- chatbot: keep person on other end guessing

This is a view we'll return to later – for the moment, we'll just stick with the discrete, “achieve particular goal state” view

- first: we need some way to *describe* the world – have to be able to differentiate one config of world from another
- CS geeks are used to this:
 - strings: very descriptive, hard to manipulate, totally unstructured
 - arrays/vectors of measurements: easier to manipulate, semi-structured, semi-descriptive
 - algebraic statements: somewhat less descriptive, great deal more structured, easier to manipulate
 - graphs: easy to manipulate, strongly structured, moderately descriptive
 - general programs: very expressive, pretty hard to manipulate, almost totally unstructured.
- Generally, there's a tradeoff between **expressiveness** and **tractability**
- leads to field of *knowledge representation*: how to choose a “language” for describing world
- Luger jumps right into predicate calculus (Chapter 2) – we'll get to that in a bit, but for the moment, we'll just use simpler langs
- Note: this is an *internal* description language – only represents *part* of the environment
- Big question in KR is what parts of world we can effectively ignore in interest of tractable computation
- for the moment, we'll think about problems that can be represented with graphs or vectors
 - mobile robot decelerating to a stop
 - 15-puzzle/8-puzzle
 - TSP
- What are the state spaces of each of these? How big are they?
- next thing we need is a definition of our *goal state*:
 - robot: pair of (a, t) such that $x = x_{\text{goal}}$ (or, in a bit more general: $|x - x_{\text{goal}}|$ is minimized)
 - 8-puzzle: sorted configuration
 - TSP: *minimal* path
- note that two of these are minimizations and one is single goal

- in general, we'll talk about an *optimality criterion*
- **definition** optimality criterion: function $f : \mathcal{S} \rightarrow \mathbb{R}$
- next component we need is some way to manipulate these structures – the set of “legal operations”
- more formally, we'll call these inference rules:
- **definition** an inference rule is an operator for moving from one state in state space to another: $r_i : \mathcal{S} \rightarrow \mathcal{A}$
- what are inference rules/operators for these problems?
 - mobile robot: pick acceleration (deceleration) and time to apply it
 - 8-puzzle: directions for moving space around
 - TSP: extending route to date
- now the question we have to answer is how to *find* the goal state
- In some cases, soln is analytic (e.g., for mobile robot)
- more often, we have to *search* for it
- **definition** search is procedure for examining state space to locate goal state (or optimum)
- (A lot of AI people/books write entirely in terms of search; I prefer to think about more general problem of *optimization*)

2 Intro to Search

- For discrete state spaces (TSP, 8-puzzle), we can think of the space as being a graph w/ each state at a node and each edge defined by an operator
- now the problem is to systematically traverse the graph so as to find optimum
- **Breadth-first search** expands the frontier of search until goal is found. Note: guaranteed to find the shortest path to goal state.
- **Depth-first search** expands one child of a node fully before moving on to next node
- **Greedy search** (a.k.a., hill-climbing)

Next time: heuristics and heuristic search. . .