CS 171 F24 Final Exam Reference Sheet

Created by Steven He, 2024

Uninformed Search

A **state** represents particular configuration of environment. **Nodes** in search tree can encode details like state, parent, action, path cost, depth, etc.
Tree Search doesn't remember visited nodes: slow search, low memory
Graph Search remembers visited nodes: fast search, high memory
Basic search scheme maintains explored and frontier nodes. Process the frontier in some order based on some policy/search strategy, marking as visited if graph search, then expanding node.

Strategies evaluted by:

- *completeness* - does it guarantee a solution if one exists
- *optimality* - does it guarantee min-cost solution
- *time/space complexity* - # of nodes generated / max # of nodes in memory. In terms of max branching factor b, depth of min-cost solution d, maximum depth of state space m.

Uninformed search strategies:

- *Breadth-first Search* - Frontier is FIFO queue. Goal test node before push. Always complete; only optimal if cost is non-decreasing function of depth.
- *Uniform-cost Search* - Frontier is priority queue, ordered by total path cost $g(n)$. Goal test after popping from queue. Complete if b is finite; optimal if all step costs $\geq \epsilon > 0$ (non-zero).
- *Depth-first Search* - Frontier is LIFO stack. Defined as Depth-limited search with limit $l = \infty$. Goal test after pop from stack. Incomplete if depth of states is infinite; not optimal.
- *Iterative Deepening Search* - Perform Depth-limited search, while iteratively increasing depth limit. Memory advantage of DFS, completeness & optimality conditions of BFS.

Heuristic Search

A *heuristic* estimates best possible cost left to the solution, denoted as $h(n)$.

$$S \underbrace{\longrightarrow \cdots \longrightarrow}_{g(n)} n \underbrace{\longrightarrow \cdots \longrightarrow}_{h(n)} G$$

$$\begin{aligned} g(n) &: \text{known path cost so far to state } n \\ h(n) &: \text{estimate of optimal cost to goal from } n \\ f(n) = g(n) + h(n) &: \text{estimate of total cost to goal through } n \end{aligned}$$

A heuristic is **admissible** iff $\forall n, h(n) \leq h^*(n)$, where $h^*(n)$ is true optimal cost to goal. Admissible heuristics never overestimate cost to the goal.
A heuristic is **consistent** iff $\forall n, f(n') \geq f(n)$ where n' is any successor of n. $f(n)$ is non-decreasing along any path.

$$\begin{aligned} \textbf{consistent} &\implies \textbf{admissible} \\ \neg\textbf{admissible} &\implies \neg\textbf{consistent} \\ \textbf{admissible} &\not\Longrightarrow \textbf{consistent} \end{aligned}$$

Heuristic search strategies:

- *Greedy Best-first Search* - Frontier is priority queue, ordered by heuristic $h(n)$. Only graph version complete in finite spaces; not optimal even with perfect heuristic $h = h^*$.
- *A^* Search* - Frontier is priority queue, ordered by heuristic + path cost $f(n)$. Complete unless infinite nodes with $f < f(G)$. Optimal with tree search if h is admissible; graph search if h is consistent. No other algorithm with the same consistent heuristic is guaranteed to expand fewer nodes.

Game Search

todo

Constraint Satisfaction

todo

Logic

todo

Probability & Uncertainty, Bayesian Networks

todo

Intro to ML, Linear Regression, kNN

todo

Decision Trees and Neural Networks

todo

Reinforcement Learning

Markov Property - Future is independent of past given present:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \ldots, S_t] \text{ where } S_t \text{ is state at time } t$$

We use matrix $\mathcal{P}$ to define transition property from state s to s', denoted as probability in row s, column s'.

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{11} & \ldots & \mathcal{P}_{1n} \\ \vdots & & \vdots \\ \mathcal{P}_{n1} & \ldots & \mathcal{P}_{nn} \end{bmatrix}, \mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

Markov Process/Chain - Sequence of states $S_1, S_2, \ldots$ satisfying Markov property. Formally defined as tuple $\langle \mathcal{S}, \mathcal{P} \rangle$ i.e. (set of states, prob matrix)
Episode - Some sequence of traveresed states in a MP
Markov Reward Process - Give states in MP some reward. We "gain" reward R_{t+1} when transitioning from states $S_t \to S_{t+1}$
placeholder intermediary stuff goes here

	Evaluate Policy, π	**Find Best Policy, π^***
MDP Known (Planning probs)	Policy Evaluation	Policy/Value Iteration
MDP Unknown	MC and TD Learning	Q-Learning