Study Sheet AJC 22031290 cs188

Agents/ Search/Uninformed Search/Informed Search

* UCS- expand cheapest node, complete and optimal but nothing focused on getting to goal specifically, focuses on path cost/backward cost g(n) (LEAST TOTAL COST TO ROOT)
* DFS Time-o(b^m) Space-o(bm)BFS Time-o(b^s)Space-o(b^s) Optimal IFF set cost
* Iterative deepening gets best of DFS space while BFS shallow solution (setting over deeper and deeper limits for DFS)
* Greedy search- badly guided DFS (best first) focuses on goal proximity/forward cost h(n)

Time complexity- nodes expanded Space complexity- size of fringe

A\* Search and Heuristics

* A\*- combination of UCS and Greedy (hedges bets, doesn’t check nodes again after being popped) f(n) = g(n) + h(n), is optimal with ADMISSIBLE tree AND CONSISTENT HEURISTICS for graph… heuristic design is key h(n) –h(l) <= path(n->l)
* Admissibility, consistency- admissible if h(n) <= true cost to nearest goal and CONSISTENT if <= actual cost for each ARC (not to goal) h(n) –h(l) <= path(n->l)
* General heuristics
* Tree search is dangerous- can fail to detect repeated states, don’t expand a state twice

CSPs- nodes are variables and arcs show constraints

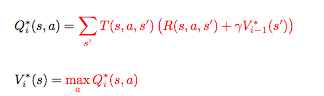
* + Goal test defined by constraints
  + Applying arc consisten to tree-structured csp guarantees no backtracking
  + MRV and LCV are helpful but don’t guarantee to reduce backtracking2
  + Solution: Backtracking search
  + Speedups: ordering, filtering, structure
  + Local search quick and no fringe but not optimal or complete (hill climbing)
  + Simulated Annealing… allow downward movement (with thought of a better max later) concept of temperature with higher temp at maxes that allows more risky moves

Game Trees

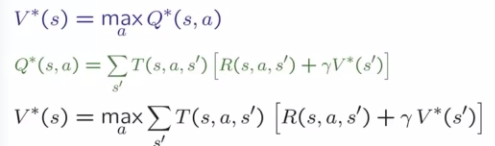
* Alpha beta pruning – alpha is best option for max on path to root, beta is best option for min on path to root
* Tree structured csps
  + Take cutset if you need to make tree structured and then linearize and check arc consistency tail to head (remove backwards and then assign forward)

Risk averse- concave Risk seeking- convex

Utilities/MDP

* Optimal quantities
  + Utility / value of a state (V\*)
    - Expected Utility startin in s and acting optimally
  + Utility/value of a q state (Q **) = Sum over s**\*Transition(s,a,s\*)[R(s,a,s\*) + gammaV\*(s\*)]
    - Epected utility starting out having taken action a from state s and acting optimally
  + The optimal policy (pi \*)
    - Optimal action from state s
    - 

Bellman Equations



value iteration is guaranteed to converge with gamma less than 0



Policy extraction (policy can converge faster than values)

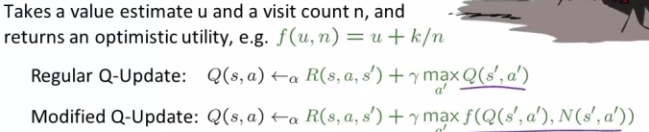
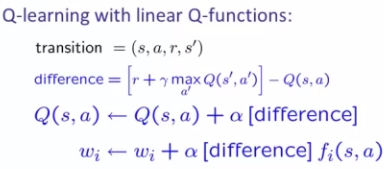
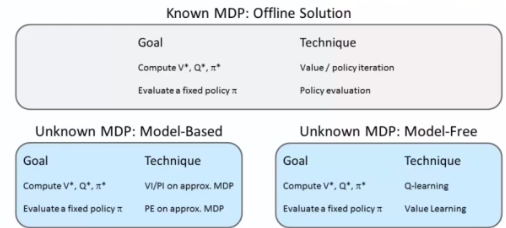
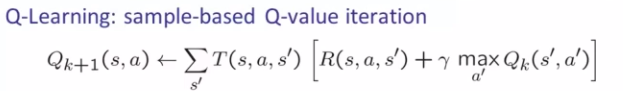


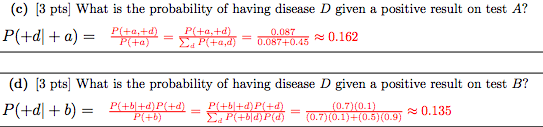
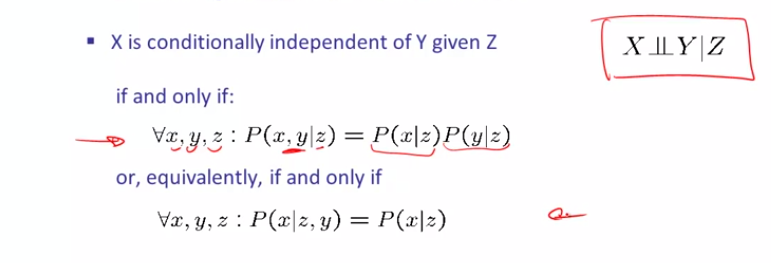
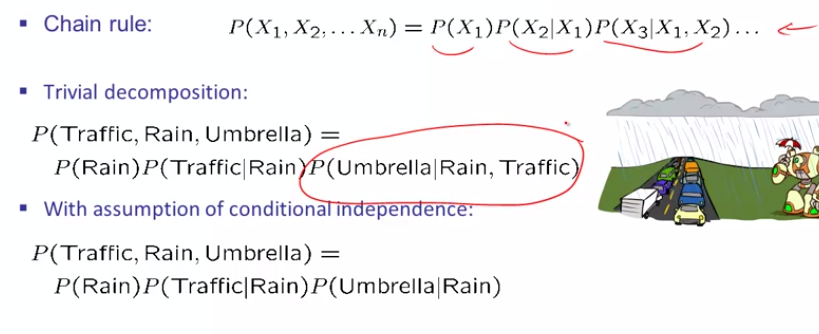
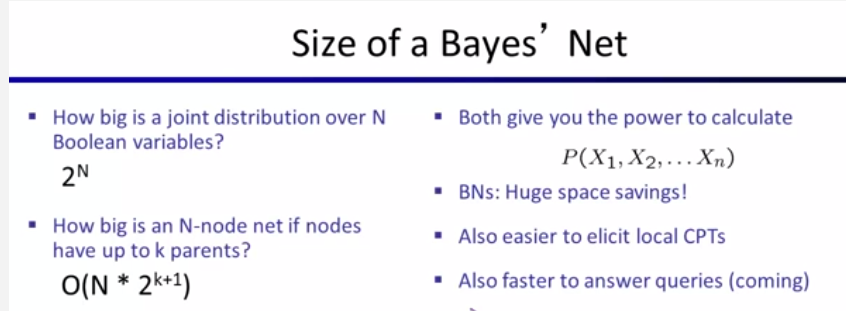
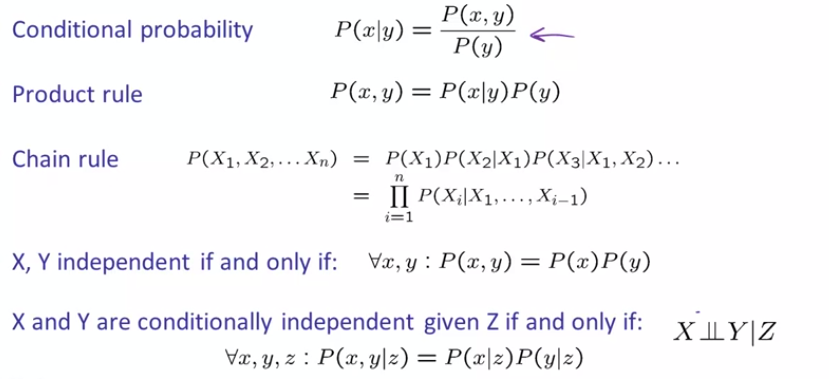
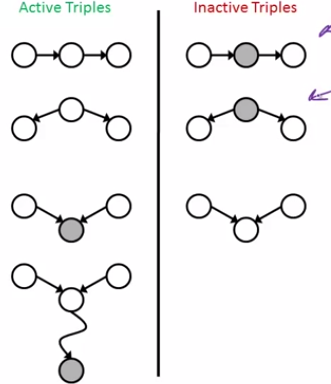
Temporal difference (same for Vi+1 except no max statement)





Reinforcement Learning



**Variable Elimination**

