uninformed search methods

- DFS
- BFS
- UCF

informed search A*

Admissible heuristics

h is admissible if:

$$0 \le h(n) \le h^*(n)$$

$h^*(n)$ is true cost at node n

optimal in tree search

Dominance: $h_a \ge h_c$ if

$$\forall n: h_a(n) \ge h_c(n)$$

consistent heuristic

$$\forall A, C: h(A) - h(C) \le \text{Cost}(A \text{ to } C)$$

A* graph search is optimal

CSPs

Variables Domain Goal test is a set of constraints

Variables: WA, NT, Q, ...

Domain: D = {Red, green, blue}

implicit: WA ≠ NT

explicit: (WA, NT) ∈ {(Red, green) {Red, blue}...}

Solution:

Backtracking = DFS + variable-ordering + fail-on-violation

0. for each value in order domain values
1. Only consider assignment to a single variable at each step. 2. check assignment, if good, add to assignment, if no, remove value from assignment.

Filtering: improve backtracking by forward checking

filtering: Keep track of domains for unassigned variables and cross off bad options

Forward checking: Cross off values that violate a constraint when added to the existing assignment

Consistency of a single arc, An arc $\overset{tail}{X} \to \overset{head}{Y}$ is consistent iff every X, there's some Y which could be assigned without violate a constraint

Delete from tail, neighbors of X need to be rechecked

Ordering: MRV: minimum Remaining values

choose values with the fewest legal left values in its domain (variable)

(the most constrained variable)

LCV Lease constraining value heuristic

B: (1,3), (2,3) A: (2,4) (2,5) (1,4) ... E: (3,4) (4,3)

We choose B by MRV, and for each (1,3), (2,3), enforce arc consistency, if (1,3) leaves A: (2,4) E: ∅ We chose (2,3)

(2,3) leaves A: (2,4) (2,5) E: (3,4) because 3 > 2

D: (1,2)

Independent subproblems are connected components of constraint graph

Worse case $O(\frac{n}{c}(d^c))$ n variables d subproblems each c variables

(CSPs worse case: $O(d^n)$ complete assignment (check every assignment))

CSPs Constraint graph has no loop, can be done in $O(nd^2)$

Tree-structured CSPs $O(nd^2)$

D: d_1, d_2, ..., d_n

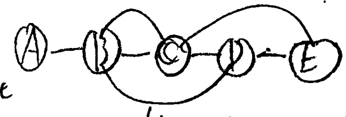
E: d_1, d_2, ..., d_n

(Remove subset set a tree)

↑

Cutset, local search (iterative improvements)

min conflict heuristic



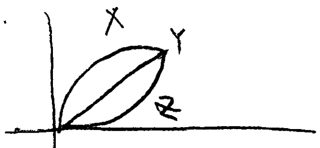
linear in n, and check both domains d^2

lottery preference $A \succ B$

$L = [p, A; (1-p); B]$

Utility of this lottery:

$U(L) = pU(A) + (1-p)U(B)$



X risk averse

Y risk neutral

Z risk taker

---

minimax efficiency

Like DFS Time $O(b^m)$

Space $O(bm)$

α max's best option on path to root

β min's best option on path to root

max-value: if $v \geq \beta$ return β

min-value: if $v \leq \alpha$ return α

MDPs $T(s,a,s') = P(s' \mid s,a)$

$V^*(s)$ the value (utility) starting in s and acting optimally

$Q^*(s,a)$ expected utility starting from s taking action a, acting optimally thereafter

$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \lambda V^*(s')]$

$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \lambda V^*(s')]$

complexity of each iteration $O(S^2 A)$

S state $\to A_1, A_2, \ldots, A_n$ action, $\to s_1, s_2, \ldots, s_n$ $O(S \cdot A \cdot S)$

policy Evaluation fix π

$V_0^\pi(s) = 0$

$V_{k+1}^\pi(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^\pi(s')]$

$O(S^2)$

Exploration vs Exploitation

Exploration: unknown, to explore unknown state

Exploitation: using known knowledge to act

Direct Evaluation

input policy

A →	D
B	C

observed Episodes

Episode 1: A, right, D, 1 ⋮

Episode 2 ⋮

Output value

4	10
2	8

Waste information

---

active Reinforcement learning

goal: learning the optimal policy/values

$Q_{k+1}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$

random action (ϵ-greedy)

with small prob ϵ, act random

with large $1-\epsilon$, act on current policy

Better idea: explore area whose badness is not estiblished, eventually stop exploring

Visit count n, $f(u,n) = u + \frac{k}{n}$

$Q(s,a) =_\alpha R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$

policy extraction

$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$

$\pi^*(s) = \arg\max_a Q^*(s,a)$

One step look-ahead

$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi_i}(s')]$

Model-Based learning

1. Learn empirical MDP model $\hat{T}$ and $\hat{R}$
2. solve learned MDP

Temporal difference learning (model free)

fixed policy sample $= R(s, \pi(s), s') + \gamma V^\pi(s')$

Update V(s) $V^\pi(s) = (1-\alpha)V^\pi(s) + \alpha\,\text{sample}$

Can't turn values into a new policy

Q-learning

sample $= R(s,a,s') + \gamma \max_{a'} Q(s',a')$

$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[\text{sample}]$

Approximate Q-learning

$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots$

adjust weights

Transition: (s, a, r, s')

difference $= [r + \gamma \max_{a'} Q(s',a')] - Q(s,a)$

$Q(s,a) = Q(s,a) + \alpha[\text{difference}]$

$w_i = w_i + \alpha[\text{difference}] f_i(s,a)$

Model-based learning: estimate Transition and Reward functions

Model-free learning: 1.off-policy learning: q-learning

2.on-policy learning:TD learning and direct evaluation