ECE 448 cheat-sheet
Midterm
sp2023 2JUI

Expert system = knowledge base + logical rules
"Fuzzy logic"

Failure: The law of out-of-vocabulary word fragility

rational: maximizing expected utility

pros: Generality; Practicality; Solvability

AI: think and act humanly & rationally

E ← world states

agent: PEAS A ← transition model

Actuators

Types of agents: RIGU Utility-Directed

Reflex; Internal-state; Goal-Directed;

Properties of Enviroments: ODSEC Continuous

Observable; Deterministic; Episodic; Static;

P ← utility $U_t = f(E_t, A_{1:(t-1)})$

- Fully observable vs. partially observable
- Deterministic vs. stochastic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multi-agent
- Known vs. unknown

Spam Filter:

$E_t = \{M_1, \ldots, M_t, L_1, \ldots, L_t\}$

L_t=1 if the email is spam, A_t=1 if reject it.

$$U_t = -\sum_{\tau=1}^{t} C_{FR} A_\tau (1 - L_\tau) + C_{FA} L_\tau (1 - A_\tau)$$

Performance measure: An objective criterion for success of an agent's behavior

RIGU: I(past) G(past+future)

U(past+future+value) sensors → AI → effectors

strategic: the enviroment is deterministic except for the actions of other agents

---

Search

State space: the initial state, actions and transition model

(directed graph) unvisited set: Frontier

to handle repeated states: use a explored set. $O(|S|)$ $(O(|S|) < O(b^m))$

DFS: Frontier is a LIFO stack

T_n and S_n are measured by

b: maximum branching factor of the search tree

d: depth of the optimal solution

m: maximum length of any path in the state space (possibly infinite) $O(b^m)$

$|S|$: number of distinct states

$O(bm)$

BFS: frontier is FIFO queue

complete and optimal

Time: $O(b^d)$ a b-ary tree of depth

Space: $O(b^d) >> O(bm)$

Dijkstra's algorithm: Uniform-cost search

frontier is a priority queue ordered by the path cost

Time: $O\{b^d\}$ Priority queue is $O\{\log_2 d\}$/node

Space: $O\{b^d\}$

Heuristics: accurate if $h(n) \approx d(n)$

cheap if $\ll O\{b^d\}$

Manhattan distance: $h(n) = |x_n - x_a| + |y_n - y_a|$

Greedy: min $h(n)$

A*: $f(n) = g(n) + h(n)$ $g(n)$: path cost $c(n)$

$h(n)$ is admissible if $d(n) \geq h(n)$

Bad interaction between A* and es.

To solve it, ① Nodes on the explored set are tagged by $c(n)$; ② use a heuristics not only admissible but also consistent

$h(n)$ is consistent if $d(n) - d(p) \geq h(n) - h(p)$

Dijkstra = A* with $h(n) = 0$

Relaxed problem: fewer restrictions

Admissible heuristics $h_1(n)$ and $h_2(n)$

if $h_2(n) \geq h_1(n)$ for all n, then

h_2 dominates h_1

CSP: D possible values with N variables

commutative $N!\, D^N$

Recursive-backtracking search

(DFS with heuristics that

① choose the next variable to asign

Least Remaining Values (LRV)

Most Constraining Variable (MCV)

② choose a value for that var

Least Constraining Assignment (LCA)

③ Early dectection of failure

Forward checking: Terminate if any illegal

Arc Consistency: Constraint propagation

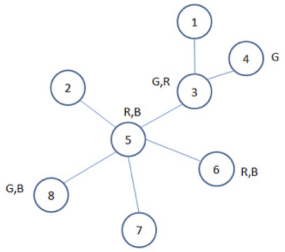
check and re-check

```
function RECURSIVE-BACKTRACKING(assignment, csp)
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp)
    if value is consistent with assignment given CONSTRAINTS[csp]
      add {var = value} to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove {var = value} from assignment
  return failure
```

Apply *inference* to reduce the space of possible assignments and detect failure early

```
function AC-3(csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
  local variables: queue, a queue of arcs, initially all the arcs in csp
  while queue is not empty
    (Xi, Xj) ← REMOVE-FIRST(queue)
    if REMOVE-INCONSISTENT-VALUES(Xi, Xj) then
      for each Xk in NEIGHBORS[Xi] do
        add (Xk, Xi) to queue
```

```
function REMOVE-INCONSISTENT-VALUES(Xi, Xj) returns true iff succeeds
  removed ← false
  for each x in DOMAIN[Xi]
    if no value y in DOMAIN[Xj] allows (x,y) to satisfy the constraint Xi ↔ Xj
      then delete x from DOMAIN[Xi]; removed ← true
  return removed
```



Solution:

1. Sort the nodes into a search order with no crossed edges, e.g., 2,8,7,6,5,3,1,4.
2. Perform arc consistency on each pair of nodes, from right to left. With the ordering specified above, the following changes are made: 3 cannot be G. 5 cannot be R. 2, 8, 7, 6 cannot be B.

3. Assign the variables to any remaining value, from left to right. For example, 2=G, 8=G, 7=G, 6=R, 5=B, 3=R, 1=B, 4=G.

Algorithm	Complete?	Optimal?	Time complexity	Space complexity	Implement the Frontier as a...
BFS	Yes	If all step costs are equal	O(b^d)	O(b^d)	Queue
DFS	No	No	O(b^m)	O(bm)	Stack
IDS	Yes	If all step costs are equal	O(b^d)	O(bd)	Stack
UCS	Yes	Yes	Number of nodes w/ $g(n) \leq C^*$	Number of nodes w/ $g(n) \leq C^*$	Priority Queue sorted by g(n)
Greedy	No	No	Worst case: O(b^m) Best case: O(bd)	Worse case: O(b^m) Best case: O(bd)	Priority Queue sorted by h(n)
A*	Yes	Yes	Number of nodes w/ $g(n)+h(n) \leq C^*$	Number of nodes w/ $g(n)+h(n) \leq C^*$	Priority Queue sorted by h(n)+g(n)

$O\{2m\} = O\{m\}$ DFS if many sol

Universal Instantiation: $\forall x, F(x)$

Existential Instantiation: $\exists x, F(x)$

Forward Chaining: generates a new sentences by combining many diff states.

Backward chaining: apply {set of known sentences} to generate {set of desired sentences} in order to the target sentence

Diff between Planning and Theorem Proving: planning may negate some of its preconditions

1st h(n): number of goal sentences left to achieve; 2st h(n): planning graph upper bound Ant S_{n+1} + Mutex links until no mutex Convergent h(n) can be stages that no longer mutex links

Planning is PSPACE-complete

QBF: $\exists x_1 \forall x_2 \exists x_3 \forall x_4 (\Lambda_1 \vee \neg x_3 \vee x_4) \wedge$ $(\neg x_2 \vee x_3 \vee \neg x_4)$

quantified boolean formular (games)

SAT: $\exists x_1 \exists x_2 \exists x_3 \exists x_4 (x_1 \vee \neg x_3 \vee x_4) \wedge$ $(\neg x_2 \vee x_3 \vee \neg x_4)$

puzzles

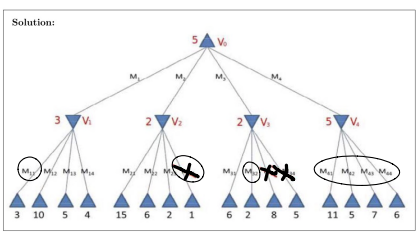
Two player's game: tic-tac-toe

Max-min

me my opponent

zero-sum game

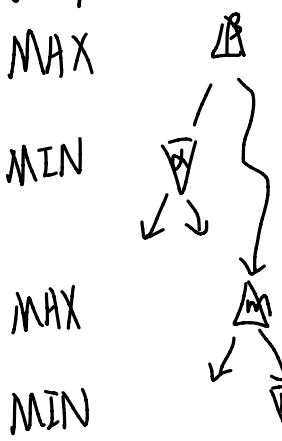
=> minimax tree



Alpha-beta pruning ($O(b^{\frac{m}{2}})$ from $O(b^m)$)

β is the value of the best choice for the MIN player found so far (lowest number)

α is the value of the best choice for the MAX player found at any choice above node n.



we want to compute

① MIN value at n, loop over n's children, it decreases

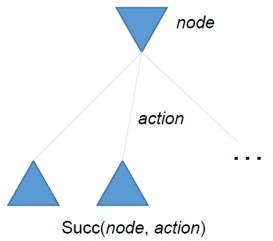
② MAX value at m, loop over m's children, it increases

$\alpha \leq \beta$

Function *action* = **Alpha-Beta-Search**(*node*)
  v = **Min-Value**(*node*, $-\infty$, ∞)
  return the *action* from *node* with value v

α: *best alternative available to the Max player*
β: *best alternative available to the Min player*

Function v = **Min-Value**(*node*, α, β)
  if Terminal(*node*) return Utility(*node*)
  $v = +\infty$
  for each *action* from *node*
    v = Min(v, **Max-Value**(Succ(*node*, *action*), α, β))
    if $v \leq \alpha$ return v
    β = Min(β, v)
  end for
  return v



branch-factor: number of children nodes for each parent

Function *action* = **Alpha-Beta-Search**(*node*)
  v = **Max-Value**(*node*, $-\infty$, ∞)
  return the *action* from *node* with value v

α: *best alternative available to the Max player*
β: *best alternative available to the Min player*

Function v = **Max-Value**(*node*, α, β)
  if Terminal(*node*) return Utility(*node*)
  $v = -\infty$
  for each *action* from *node*
    v = Max(v, **Min-Value**(Succ(*node*, *action*), α, β))
    if $v \geq \beta$ return v
    α = Max(α, v)
  end for
  return v

③ Transposition table

④ Forward Pruning

⑤ Lookup table

Limited-horizon search

$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$

Horizon effect: the opponent's move effect delayed but no avoided

Remedies: ① Quiescence search; ② singular extension

Nash equilibrium: A pair of strategies such that no player can get a bigger payoff by switching strategies, provided that the othe player sticks with the same strategy.

Dominant strategy

Pareto optimal outcome

For better: ① Super rationality: Categorical Imperative

② Repeated Games:

↑ cooperation

↳ in stag Hunt

Sealed-bid second-price auction: the winner pays the price of the second-highest bid (bid your V)

truth revealing mechanism

Probability: $P(A|B) = \frac{P(A,B)}{P(B)}$ Conditional distribution

$P(\neg A, B) = P(B) - P(A,B)$

A, B is independent iff $P(A,B) = P(A) \cdot P(B) \Rightarrow P(A|B) = P(A)$

$P(B|A) = P(B)$

computational complexity limitation

environment: unknown / partially observable

stochastic

expected utility:

$E[\text{Utility}|\text{Action}] = \sum_{\text{outcomes}} P(\text{outcome}|\text{action}) \cdot \text{Utility}(\text{outcome})$

Solution: The stag hunt is a coordination game. The payoff matrix is

	Player 1: Cooperate	Player 1: Defect
Player 1: Cooperate	2, 2	0,1
Player 2: Defect	1,0	1,1

There are no dominant strategies. The pure-strategy Nash equilibria are (Cooperate,Cooperate) and (Defect,Defect). The Pareto-optimal solution is (Cooperate,Cooperate).

The game of Chicken is an anti-coordination game. The payoff matrix is

	Player 1: Chicken	Player 1: Drive
Player 1: Chicken	0, 0	-1,1
Player 2: Drive	1,-1	-10,-10

There are no dominant strategies. The Pareto optimal solutions are (Chicken,Chicken), (Chicken,Drive), and (Drive,Chicken). The pure-strategy Nash equlibria are (Chicken,Drive) and (Drive,Chicken).

Solution: Minimax solution maximizes, over all of your possible actions, the minimum, over all of your opponent's possible actions, of your reward.

- Prisoner's Dilemma: Defect. Result is also the Nash equilibrium.
- Stag Hunt: take the Hare. Result is one of the two Nash equilibria.
- Game of Chicken: chicken out. Result is not a Nash equilibrium.

Nash equilibrium assumes that you know what the other player will do, and can respond appropriately. Minimax makes more sense if you want to limit your losses, and have no way to predict the other player's behavior.

utility theory is used to represent and infer preference

decision theory = probability theory + utility theory

Frequentism, Subjectivism

$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

Atomic events: mutually exclusive and exhaustive

joint distribution: $\sum P_i = 1$

marginal distribution: from $P(X,Y)$ to get $P(X)$ and $P(Y)$

$P(A,B) = P(A|B) \cdot P(B)$ product rule

$= P(B|A) \cdot P(A)$ chain rule

$P(A, B \ldots N)$

mutually exclusive $P(A \vee B) = P(A) + P(B)$

conditionally independent

$P(A \wedge B | C) = P(A|C) \cdot P(B|C)$

ExpectMinimax

max (min (E (Reward))

SL & RL

Monte Carlo Tree Search