Adversarial Search :-

minimax → brute force (searched through all the nodes)

Alpha-Beta Pruning →

For a leaf node ⇒

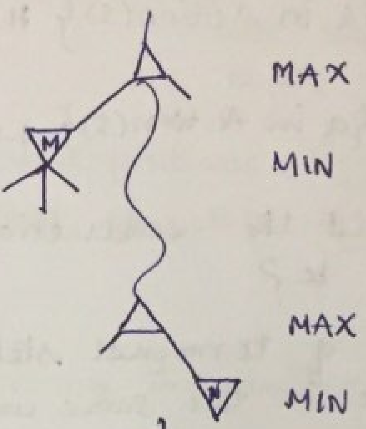
$\alpha = \beta =$ utility

At max, α = largest child utility

$\beta = \beta$ of parent

At min, $\alpha = \alpha$ of parent

β = smallest child utility (children explored till then)



If M is a better choice, then N wont be searched.

$\boxed{\alpha \leq \text{utility} \leq \beta}$

If $\alpha > \beta$, Pruning happens. as this condition fails.

Good move ordering $\equiv O[b^{m/2}]$ → improves the effectiveness of pruning

Dynamic move ordering → from historical data (data in the past)

can be used by implementing "Iterative Deepening Search"

Transpositions $\equiv$ different move sequences (multiple paths) → end up in same board position [final state]

→ repeated states

Alpha-Beta algorithm → has to search till terminal states

[though pruning is allowed for large parts of it]

→ Cut off the search + & use a heuristic function evaluation.

Heuristic Minmax →

* H-Minimax (S,d) ≡

- EVAL(S) if cut off-Test (S,d)
- Max_{a in Action(S)} H_Minimon (Result (S,a), d+1) if player(s) = MAX
- Min_{a in Action(S)} Minimax (Result (S,a), d+1) if Player(s) = MIN

what should the evaluation function be?

- ordering of terminal states should be in the same way as the true utility function
- No long computations
- Highly correlated

Features ≡ define categories or equivalence classes of states i.e state in each category have same values for all features.

↓ some states ≡ wins, draws, losses.

Ex: No. of pawns on board } = 4 (for many no. of sequences of moves)

→ too many categories (Not easy to estimate the probabilities of winning)

↓

Weighted Functions

|||

linear comb^n of func^ns { Eval(S) = w1. f1(s) + w2 f2(s) . . . .

weight feature

These features, weights → not part of the rules of chess

→ Domain Experts (Deep learning)

→ Weights of evaluation

≀

Machine learning Techniques

Modifying the alpha-beta search →

- Pick an appropriate depth "d" — allocated time
  - ↳ Iterative Deepening Search [if you run out of time, you still have the best solution for depth "d-1"]

- Quiescence search — "avoid positions (Searching the states) that result in wild swings"
  - ↓ heuristics

- Singular extensions — storing the better moves.

- Forward Pruning — Pruning at that instant without expanding further.
  - └ Beam Search → breadth of the tree might be increasing but make sure the breadth "b" is fixed with the best actions and leave the pruned nodes.
  - disadvantage ↓ a chance of pruning out the best moves

- PROBCUT (Probabilistic cut algorithm) ≡ Alpha beta search ↓ prune if you guarantee that the solution doesnt lie in the window "provably"
  - ↓ it is a heuristic → (prior experience (Large data set of previous games)) probability for pruning
  - "probably" — pruning of nodes ~~the~~ outside the window
  - ↓ Shallow search

≡ uses past experience to compute how likely a score of v at depth d would be outside of (α, β)

v ↓ computed value of the node

outside ↓ probability

= For node of value "v" — if it results in loss for more no. of games in past

⇓ Probability (depends on past data sets)

↓ ProbCut likely prunes the node "v".

Good Evaluation function + Reasonable cut off test

+ quiescence search + Large transposition table

↓

searching for visibility.

→ Search — you dont want to find every more using search from scratch (for every position)

↓

better to do.

<u>Table Lookup</u> (from the database) ≡ mapping from every possible state to the best move in that state.

Stochastic Games ≡ involving a random element

<u>ForEx</u> : throwing a dice.

⟹ DECISION THEORY & PROBABILITY THEORY

Decisions in the face of uncertainity [Dont use adhoc techniques]

what is uncertainity?
& how to reason with it?

Ex Logical Agent

- slow(Route 1), fast(Route 2)
- slow(x) => Avoid(x)
- Avoid(x) ^ fast(y) => select(y)

Avoiding the route x and route y being fast

lead to selecting the path y

- Agent selects Route 2

Not a good way to logically reason ~~this~~ situation — why?

- take ~~into~~ the distance into the account
- "speed" of the agent
- look at the trade offs

Uncertainity ≡

→ both qualitative & quantitative information

Ex: if we know it is slow, how slow?

Changes the way an agent makes a decision.

— Cant take each factor/probability into account

may be unknown [cognitive overload]

Rational Decision

→ relative importance of various goals (with uncertainities)

→ Degree of success of the goals.

there can be ~~goals~~ decisions with or without uncertainities.

without uncertainity, it doesnt mean it is an ideal choice as it may be having the least degree of success (end up never reaching the goal)

Ex: Being a prime minister (reward ≡ 1000) (probability = 0.5%) or a software engineer (reward = 10) (probability = 99%).

*consider the probability of success also (understanding the trade offs)

Probabilities of each action => can determine the correct decision involving these actions

Ex:	Route 1	P	Route 2	P
without Accident	20min	80%	25min	70%
when Accident occurs	50min	20%	40 min	30%

→ DECISIONS UNDER UNCERTAINITY

Making a
- choice among actions or plens:
  → chance of success / failure
  → Consequence or cost or reward of success or failure.

Decision Theory = Probability Theory + Utility Theory
(making decisions) (deals with chance) (deals with outcomes)

→ Maximum Expected Utility (MEU)
- choosing the action that yields the highest expected utility
- weigh the outcome.

ex: Trading a 100Rs note for a pen.
(So the value of pen is higher than money at times and viceversa)

* Conditional Probability = if Agent has some evidence.

$P(A/B)$ = probability of A given that all we know ~~abo~~ about is B.

Joint probability Distribution => table

For Ex: P(Weather, Cavity) => P(Weather ∧ Cavity)

Weather ↓ 4 values {sunny, rain, cloudy, snow}

Cavity ↓ 2 values {true, false}

↓ 4 * 2 joint probability distribution table.

$$\Rightarrow P(Y) = \sum_{z \in Z} \underbrace{P(Y,z)}_{\text{Ydepending on z}} = \sum_{z} P(Y/z).P(z)$$

$P(\text{Cavity}) = \sum_{z=\{\text{Catch, Toothache}\}} P(\text{Cavity}, z)$ ∴ Summing out (or) Marginalization.

Idependent variable → is converted into a conditioned.

→ Conditioning cavity on {Cautch, Toothache}

↓ Issue with this: For a domain with n Boolean variables table size goes up to $[2^n]$.

↑ building a table itsef is hard.

↓ Full joint distribution in tabular form — Not a practical tool. for building reasoning systems.

Independence → If variables are independent, you dont need to build any table

ForEx: x and y variables (10 values each) — dependent $= 10 * 10 = 100$

independent $= 10 + 10 = 20$.

Ex:

$P(\text{toothache, catch, cavity, cloudy}) =$ $2*2*2 = 8$ entries

$\Uparrow$ $P(\text{cloudy} \mid \text{toothache, catch, cavity}) *$

$P(\text{toothache, cetch, cavity})$

↓

$P(\text{cloudy} \mid \text{toothache, cetch, cavity}) = P(\text{Cloudy}) = 4$ entrie

↑ Cloudy doesnt depend on all these factors.

Instead of modelling all the $2*2*2*4 = 32$ entries

↓ we can just do $\underbrace{2*2*2+4}_{\text{due to the independence}} = 12$ entries

Baye's Rule →

- $P(A \wedge B) = P(A|B) * P(B)$
- $P(A \wedge B) = P(B/A) * P(A)$
- $P(B/A) = (P(A/B) * P(B))/P(A)$.

$P(\text{effect}/\text{cause})$ — causal Knowledge.
↓
"cause is given"

$P(\text{cause}|\text{effect})$ — diagnostic Knowledge.
↓
figuring out cause from the given effects.

S: Patient has stiff neck
M: Patient has meningitis

→ Meningitis causes stiff-neck, 70% of the time.

$P(S|M) = 0.7$ (causal Knowledge, does not change) — "model based"

$P(M) = 1/50{,}000$

$P(S) = 0.01$

$P(M/S) = (0.7 * 1/50{,}000)/0.01 = 0.0014$

$(P(S/M) * P(M))/P(S)$

stiff neck causing meningitis.

probability is too low because the probability of meningitis itself is very low.

Computing Relative likelihoods — If meningites & Whiplash are two possible explanations for stiff neck

$$P(M/S) = \frac{P(S/M) * P(M)}{P(S)}$$

$$P(W/S) = \frac{P(S/W) * P(W)}{P(S)}$$

$$P(M/S)/P(W/S) = P(S/M) * P(M) \Big/ P(S/W) * P(W)$$

$P(M/S) + P(\sim(M/S)) = 1$

$P(\sim(M/S)) = P(S/_{\sim M}) * P(\sim M) / P(S)$

$\Rightarrow P(S/M) * P(M) + P(S/_{\sim M}) * P(\sim M) = P(S)$

→ Baye's rule for multi-valued vaiiables

$\rightarrow P(Y|X,e) = P(X|Y,e) * P(Y|e) / P(X,e)$

e, background evidence.

→ Conditional Independence :-

variables becoming independent in the presence of ^ other variables

- toothache and cetch are independent given the presence of cavity

cetch is not being perfomed to check for toothache but for catching cevity

$\Rightarrow P(\text{toothache} \wedge \text{cetch}/\text{cavity}) =$

$P(\text{toothache} \mid \text{cetch, cevity})\, P(\text{catch} \mid \text{cavity})$

reduces the table size. presence of information about cavity

Ex:

S: Patient has stiff neck
H: Patient has headache
M: Patient has meningitis

$P(S/M) = 0.7$

$P(H/M) = 0.5$

& P(M)?
how to combine both the symptoms

$P(M \wedge S \wedge H) = P(S \mid M \wedge H) * P(M \wedge H)$

$= P(S/M) * P(M \wedge H)$

$= P(S/M) * P(H/M) * P(M)$

stiff neck and headache are not related but only over meningitis

Scanned by CamScanner

→ Conditional Independence ↓

↓ decomposing larger tables ⇝ small tables

common

"they become independent in the presence of other variables"

UTILITY THEORY

Utility Theory ≡ deals with outcomes.

$EU(plan1) = P(home\text{-}early / plan1) * U(home\ early) + P(stuck1 / plan1) * U(stuck1)$

↓ quantifying the probabilities with "utility"

RISK AVERSE —

$EU(Choice\ 1) = 10$
$EU(Choice\ 2) = 9$

• Convex function ≡ Slope of utility function is continuously decreasing

$$U(x+c) - U(x) < U(x) - U(x-c)$$

↓ why you should not play the game?

$$U(x+c) + U(x-c) < 2U(x)$$

$$\left[U(x+c) + U(x-c)/_2 \right] < U(x).$$

↳ EU(playing the game) < EU(not playing the game)

RISK SEEKER —

[concave] $EU(choice\ 1) = 10$
✓ $EU(Choice\ 2) = 25$
(he'll definitely play the game)

⇓

Problem statement — Choice 1 ≡ without playing, you'll get 1 million dollars

Choice 2 ≡ On playing, if you get head ≡ 3 milliondollars
tail ≡ 0 (nothing)

↓ which is a rationally better choice?

Markov Decision Process (MDP) ≡ Sequential Decision Making Problems.

↓ factor the transition uncertainities

→ Making complex Decisions —

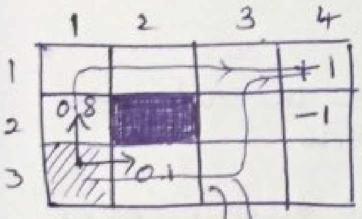
⟨S, A, P, R⟩

S – State of the agent
A – Actions of the agent
P – Transition function — $P(s' | s, a)$
R

↓ probability of "s'", on performing action a on state "s"

↓ it may not definitely end up. in "desired state". (probability is present)

might end up in an "undesired state".



start

$P((4,3) | (3,3), N) = 0.1$ [Going "N ≡ North" from (3,3) to reach (4,3)]

→ $P((3,2) | (3,3), N) = 0.8$

How to solve? — 1. Simple search — X Not Deterministic "stochastic".

↓ each plan doesnt account all the other states.

↓ 2. Apply "Maximum Expected Utility" to an entire sequence of actions.

↓ but there's uncertainity is present at every step.

≡ In the top plan (set of Actions)
- what if we reached east and not ~~ea~~ north.
↓ this state is not even present in the (action sequence).

⇒ *Markov Assumption ≡ Transition probabilities from any given state depend only on the "current state" and not on previous history. X.

Decision Epoch → Points at which decisions are made

√d Finite horizon MDPs ≡ finite , Time Dependent Policy

No. of decision epochs

Infinite horizon MDPs ≡ infinite, Time Independent Policy.

[agent's gonna live forever]

Absorbing State ≡ Goal state.

Reward Function ≡ $R(S, A)$ → Reward associated with a state and an action.

[default] $R(S)$ → if all actions ~~rewards~~ ~~are associated~~ ~~we~~ have same rewards

$R(S, A, J)$ ≡ associated with a state, action and destination_state.

$$\Rightarrow R(S,A) = \sum R(S,A,J) * P(J/S,A)$$

Decision Rule ≡ Procedure to choose action in each state for a given decision epoch.

Policy ≡ Decision Rule to be used at all decision epochs

Ex: (finite horizon, $T=4$) $\{D_1, D_2, D_3, D_4\}$.

Stationary and Deterministic Policies ⇒

- Stationary ≡ same [decision rule] in every epoch

  Stationary policy ≡ $\{D, D, D, ..\}$

  Non-stationary policy ≡ changes with time $\{D_1, D_2, D_3 .. D_n\}$

- Deterministic ≡ choosing an action with certainity.

Computing Optimal Policies:

~~Choose such a path~~

— Value Iteration —

- Iterate — update utility of state "i" using old utility of neighbor states "j", given actions "A"

At ($t-1$) states ⟶ At (t)

Reward (Step Cost).

$$U_{t+1}(I) = \max\left[R(I,A) + \sum P(J|I,A) * U_t(J)\right]$$

[Bellman update equation]

max of Values for all the actions possible (A)

Can reach any of the new states with action A. are summed up.

$$\boxed{U_{t+1}(I) \leftarrow \max(A) \underset{\text{actions possible}}{=} t}$$

immediate reward

longerterm reward

Algorithm ⟹

- $U_0(I) = 0$.
- Iterate: (A) $$U_{t+1}(I) = \max_A\left[R(I,A) + \sum_J P(J|I,A) * U_t(J)\right]$$

— Until Close Enough (U_{t+1}, U_t)

⇓

→ determine close enough:

- δ = max change in utility of any state in an iteration.
- $\delta <= \text{BOUND}$

- At the end of iteration,

$$\text{Policy}(I) = \text{argmax}(R(I,A) + \sum P(J/I,A) * U(J)]$$

Linear Programming Model

Model-1 –

Let $x_1, x_2, \ldots x_n$ = decision variables.

Z = objective function or linear function.

$$\int z = c_1x_1 + c_2x_2 + c_3x_3 \ldots c_nx_n.$$

Maximization of the lineer function, Z. under various constraints.

Model 2: – (Efficient Notation).

Maximize , $Z = \sum_{j=1}^{n} c_jx_j$

subject to $\sum_{j=1}^{w} a_{ij}\,x_j \leq b_j$

where $i = 1, 2, \ldots m$

and $x_j \geq 0$.

where $j = 0, 1, \ldots n$.

Examples of LP Problems →

- Product Sell
- Transpotation (minimizing the cost of transpotation)
- Flao Capacity (maximizing the flao capacity)

For Ex: Product Sell: ↓

$$Z = 13x_1 + 11x_2$$

Maximize

$$Z = 13x_1 + 11x_2$$

subject to constraints ↓

$$4x_1 + 5x_2 \leq 1500$$

$$5x_1 + 3x_2 \leq 1575$$

$$x_1 + 2x_2 \leq 420.$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

13 products-x_1

11 products-x_2

$4x_1 + 5x_2 \leq 15000$ (other constraints)

$5x_1 + 3x_2 \leq 1575$

$x_1/_{60} + x_2/_{30} \leq 7$

Constraint Table

	I	II	
Storage	4	5	150
Raw	5	3	157
Product Rate	60	30	
Selling	13	11	

Linear Programming for MDPs

Maximize, $\sum_{i \in S} V_i$ (S is the set of states, V_i = cost of state)

such thet : $V_i <= \Big[\underbrace{R(I,A)}_{\text{immediate reward}} + \gamma \sum P(J|I,A)^* V_j\Big]$

$\sum_A$ (all possible actions)

Linear programming polynomial time formulation. → slower than Value Iteration.

→ Simplex Method – Dantzig's

Popular Formulation ≡

x_{ia} → flow out of state i on performing action "a"

$$\max \sum_i \sum_a x_{ia} r_{ia}$$

r_{ia} → reward for taking action a in state i

j – destination

i – origin

$$\sum_a x_{ja} - \sum_i \sum_a x_{ia} p_{ij}^a = \alpha_j$$

$x_{ia} \geq 0$

p_{ij}^a → probability of reeching state j when action a is teken in state i

α_j → initial probability of being in state j

Flow generated by the system ≡ initial probability of being in state j (except for the start states)

$$\sum_i \sum_a (\delta_{ij} - p_{ij}^a) x_{ia} = \alpha_j, \quad x_{ia} \geq 0.$$

δ_{ij} → Kronecker delta, $\delta_{ij} = 1$ $(j = i)$, else 0.

Artificial Intelligence →

$Ax = \alpha$ (Solving MDP using LP.)

building "A" matrix { flowing out = +ve for the state whereas flowing in = −ve for that state.

CMDP → to model constraints
↓
Addition of such constraints result in "Constrained MDP" $\boxed{\text{CMDP}}$

to include a Constraint → very easy when done using LP.

→ goal: OPTIMISE THE TEAM REWARD.

MMDP → extension over MDPs for multiple agents.

↓ joint action set, $M = \langle S, \{A_i\}_{i \in m}, T, R \rangle$

R → team reward function

$\{A_i\}$ ↓ set of joint action $\langle a_1, a_2, \ldots a_m \rangle$ where $a_i \in A_i$.

↓ state is fully observable by each agent.

→ Dec-MDP — you may not view your team mate's state, you will be Knowing only your local state

↓ exponentially harder than MMDP.

→ goal: OPTIMISE THE TEAM REWARD, but now we need to think through all the possibilities of your team mate [considering all conditions]

Policy Notation:

Policy denoted by π
optimal policy denoted by π^*

$\pi^*(s)$ = optimal policy, state s.

Sound(message1.wav)

A B C D

Markov Chain → $\left(U_{t+1}(1) = \underset{\downarrow \text{ no max}}{R(1,A)} + \gamma \sum_{J} P(J|1,A)^{*} U_{T}(J)\right)$

→ we can use value iteration algorithm, but with fixed action ↴

you are not decinding the action, the policy has already done that for you.

In value iteration, argmax is not needed ↴

No need to pick the best action.

Compairing two policies.

0 0 4 for i=1:4

window(1:2)

→ If agents continue to live forever, rewards accamulated would be infinite

↓

Discounting ↴

Introducing "γ"

how to compare policies? &

$$U_{t+1}(1) = \max_{A} \left[R(1,A) + \gamma \sum_{J} P(J|1,A)^{*} U_t(J) \right]$$

POMDP – Partially observable MDP

↓

observation uncertainity (vision sensors)

||

interprcting the world.

↴

transition uncertainity.

[Can have unintended consequences]

Locomotion, sensors, your actions

→ we need to have additional details.

Searching with Partial Observations → Action can lead to several possible outcomes.

→ if agent's percepts provide no information at all → "sensorless problem".

searching in belief states ≡ Every possible set of physical states N ⇒ 2^N belief states.

Initial state ≡ Set of all state in P unless additional information is provided.

Action sets ⇒ P ≡ physical problem

- All the states in belief state had same action A in P, A ≡ action set.

- Some state had different actions, if non defined actions do not have any effect then new action set, UNION SETS OF ALL States.

- Non defined action ≡ end of the world
  ↓
  Intersection of all states.