Page No.
Date : / /

Additional refinements in mini-max algorithm.

These are methods to improve the performance of mini-max algorithm in addition to Alpha-Beta pruning algorithm.

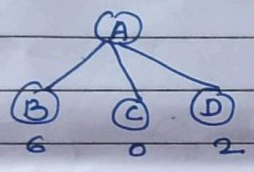
First let us understand problem with fixed depth searches →

Many a times no of possible states in games are too many. Computers can feasibly check (search) only till certain depth in the tree. While doing this may be correct move is not chosen but it's effect is not visible as search is only till limited depth.

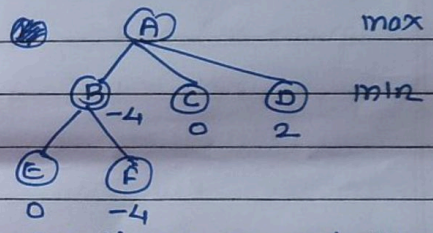
Computer does not search till the end which is beyond it's horizon. (horizon – limited depth)

This is called horizon effect / horizon problem.

eg → (a)



If you expand till level 1, B looks to be right choice/move.

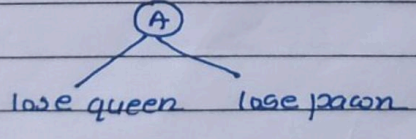


But if you expand till one more level then B does not appear to be correct choice/move

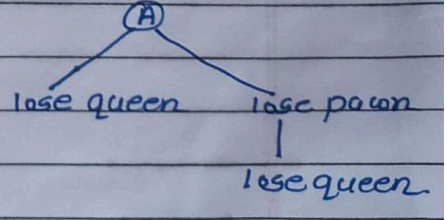
Meaning, move may look good if you expand to some depth but if expand till further depth, it does not appear to be a good move or vice versa.

eg → many situations in life, chess.

eg → (b) chess game →



If you expand till level 1, losing pawn is a good move.



If you expand till one more level, it does not appear to be good move.

Page No.
Date : / /

Two soln for this problem.

1

① Wait till stable position is reached. (Quiescene)

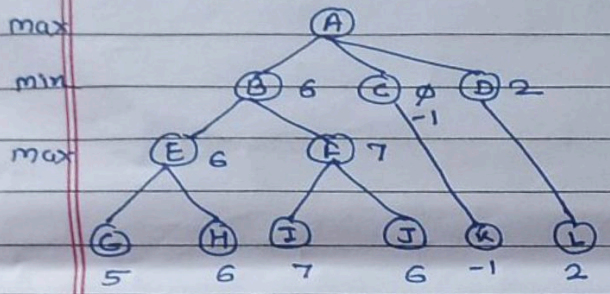
- To decide the move correctly, continue search until there is no drastic change from one level to other level.
- In other words continue search till some stable position is reached.

② Do secondary search. Implement

- It's also called look beyond the horizon.
- When search stops at any ply for eg at 5th ply, it's possible that at 6th ply result will be completely opposite as compare to 5th ply.

∴ To solve this problem explore the node / state that was chosen after searching few plies (in our case 5th ply). Explore that node or state till few more plies & result is acceptable. Else pick up the next best node & explore that node.

- eg → extending eg ⓐ



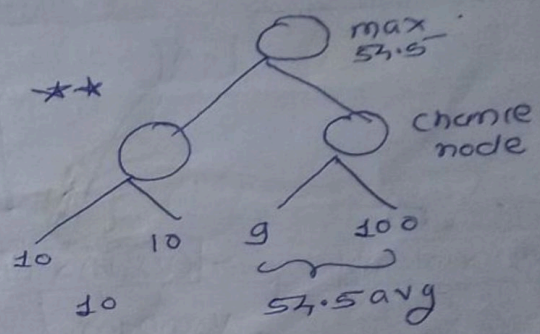
One more refinement →

③ Using Book moves / predetermined moves →

- There will be list / Book of moves & for any state given, next state or move is selected looking at the list / Book of moves.
- It's not feasible approach for complicated games where it's not possible to select the next state looking at list or some time list will be too difficult to create, it may be huge so difficult to create it or no one knows to construct it fully.
- Normally opening & end game moves are highly stylized. (Easily determinable) & Problem comes at mid level game.
  eg → real life game playing
- ∴ For opening & endgame sequence of moves use book / list of moves & apply mini-max algo for midlevel moves.

Game of chance →

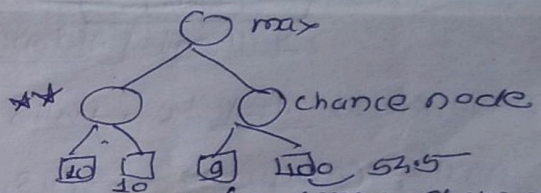
- It is a game whose outcome depends on luck.
  Le an element of chance games than ~~luck~~ skills. Outcome depends on some randomised device.
- eg → dice
  playing cards
  roulette wheel ~~based games~~
  casino



- eg → game of chance →
  - playing cards
  - gambling.
  - board games.

- formal defn → "Games whose outcome is strongly influenced by some randomizing device & upon which contestants may choose to wager the money or anyting of monetary value.".

- How to handle it →
  Use of algorithms like →



Expetimax → It's similar to minimax but allows for some elements of chance.
Here chance node is included before each player's turn & chance of nodes possible outcomes are all possible radom values.
- Decision is taken based on these outcomes.
- Basically chance node takes avg of all possible utilities/outcomes. & then maximizer chooses the node to maximize expected utility **

Monte Carlo → It is used when there is too much of randomness.
- It tries to make many random situations/simulations & tries to make an informed decision based on those many situations.
- It uses concept of probability