Game Playing

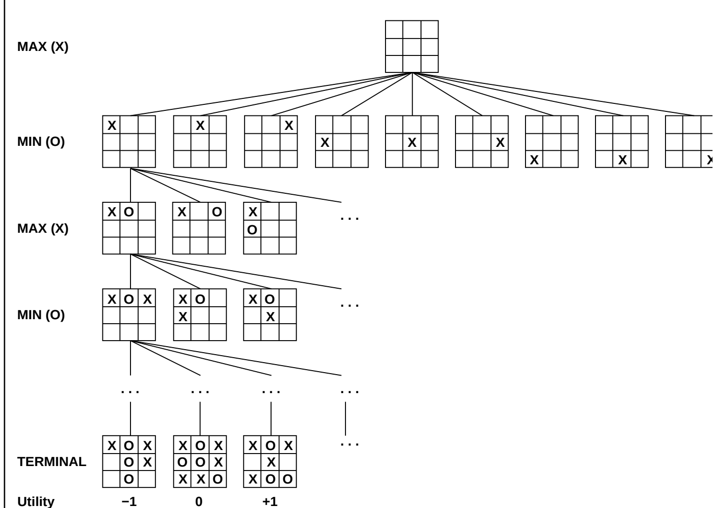
Game Playing and AI

- Games are well-defined problems that are generally interpreted as requiring intelligence to play well.
- Introduces uncertainty since opponents moves can not be determined in advance.
- Search spaces can be very large.
  For chess:
  - Branching factor: 35
  - Depth: 50 moves each player
  - Search tree: 35^{100} nodes (~10^{40} legal positions)
- Despite this, human players do quite well without doing much explicit search. They seem to rely on remembering many patterns.
- Good test domain for search methods and development of pruning methods that ignore portions of the search tree that do not affect the outcome.

Game Playing Problem

- Instance of the general search problem.
- States where the game has ended are called **terminal states**.
- A **utility (payoff) function** determines the value of terminal states, e.g. win=+1, draw=0, lose=-1.
- In two-player games, assume one is called **MAX** (tries to maximize utility) and one is called **MIN** (tries to minimize utility).
- In the search tree, first layer is move by MAX, next layer by MIN, and alternate to terminal states.
- Each layer in the search is called a **ply**.

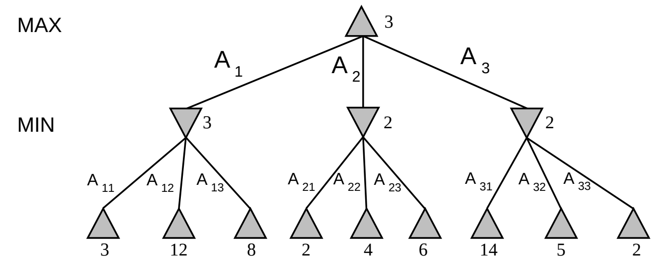
Sample Game Tree (Tic-Tac-Toe)



Minimax Algorithm

- General method for determining optimal move.
- Generate complete game tree down to terminal states.
- Compute utility of each node bottom up from leaves toward root.
- At each MAX node, pick the move with maximum utility.
- At each MIN node, pick the move with minimum utility (assumes opponent always acts correctly to minimize utility).
- When reach the root, optimal move is determined.

Minimax Computation



- Can be performed using a depth-first search

```
function MINIMAX-DECISION(game) returns an operator

  for each op in OPERATORS[game] do
     VALUE[op] ← MINIMAX-VALUE(APPLY(op, game), game)
  end
  return the op with the highest VALUE[op]
--------------------------------------------------------------
function MINIMAX-VALUE(state, game) returns a utility value

  if TERMINAL-TEST[game](state) then
     return UTILITY[game](state)
  else if  MAX is to move in state then
     return the highest MINIMAX-VALUE of SUCCESSORS(state)
  else
     return the lowest MINIMAX-VALUE of SUCCESSORS(state)
```

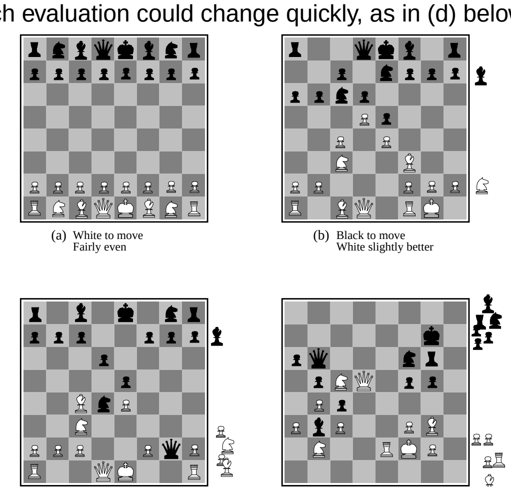

Imperfect Decisions

- Generating the complete game tree for all but the simplest games is intractable.
- Instead, **cut off** search at some nodes and estimate expected utility using a heuristic evaluation function.
- Ideally, a heuristic measures the probability that MAX will win given a position characterized by a given set of features.
- Sample chess evaluation function based on “material advantage:” pawn=1, knight/bishop=3, rook=5, queen=9
- An example of a weighted linear function:

$$w_1 f_1 + w_2 f_2 + \ldots + w_n f_n$$

Determining Cutoff

- Search to a uniform depth (ply) *d*.
- Use iterative deepening to continue search to deeper levels until time runs out (**anytime algorithm**).
- Could end in states that are very dynamic (not quiesent) in which evaluation could change quickly, as in (d) below.



(a) White to move Fairly even

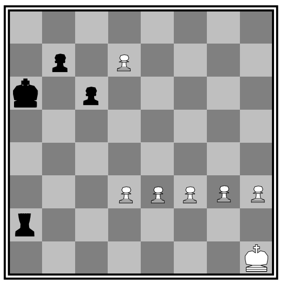
(b) Black to move White slightly better

(c) White to move Black winning

(d) Black to move White about to lose

Determining Cutoff (cont)

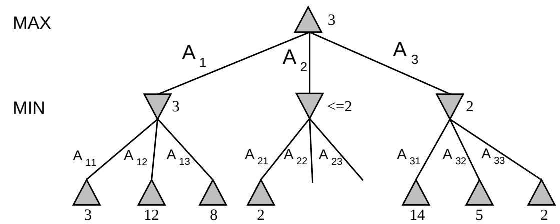
- Continue **quiescence search** at dynamic states to improve utility estimate.
- **Horizon problem**: Inevitable problems can be pushed over the search boundary.
- Example: Delay inevitable queening move by pawn by exploring checking moves.



Black to move

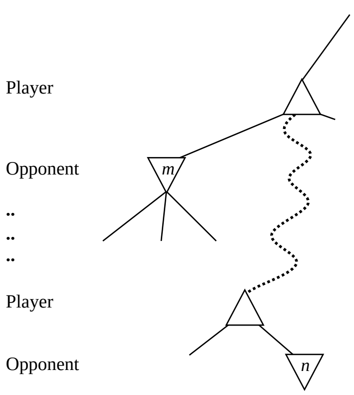
Alpha-Beta Pruning

- Frequently, large parts of the search space are irrelevant to the final decision and can be **pruned**.
- No need to explore options that are already definitely worse than the current best option.



Alpha-Beta General Principle

- Consider a node *n* where it is Player's choice of moving to that node. If Player has a better choice *m* at either the parent node of *n* or at any choice point further up, then *n* will never be reached in actual play.



- Maintain two parameters in depth-first search, α, the value of the best (highest) value found so far for MAX along any path; and β, the best (lowest) value found along any path for MIN. Prune a subtree once it is known to be worse than the current α or β.

Alpha-Beta Algorithm

```
function MAX-VALUE(state, game, α, β) returns the minimax value of state
  inputs: state, current state in game
          game, game description
          α, the best score for MAX along the path to state
          β, the best score for MIN along the path to state

  if CUTOFF-TEST(state) then return EVAL(state)
  for each s in SUCCESSORS(state) do
     α ← MAX(α, MIN-VALUE(s, game, α, β))
     if α ≥ β then return β
  end
  return α

function MIN-VALUE(state, game, α, β) returns the minimax value of state

  if CUTOFF-TEST(state) then return EVAL(state)
  for each s in SUCCESSORS(state) do
     β ← MIN(β, MAX-VALUE(s, game, α, β))
     if β ≤ α then return α
  end
  return β
```

Effectiveness of Alpha-Beta

- Amount of pruning depends on the order in which siblings are explored.
- In optimal case where the best options are explored first, time complexity reduces from O(b^d) to O($b^{d/2}$), a dramatic improvement. But entails knowledge of best move in advance!
- With successors randomly ordered, assymptotic bound is O((b/log$b$)^d) which is not much help but only accurate for b>1000. More realistic expectation is something like O($b^{3d/4}$).
- Fairly simple ordering heuristic can produce closer to optimal results than random results (e.g. check captures & threats first).
- Theoretical analysis makes unrealistic assumptions such as utility values distributed randomly across leaves and therefore experimental results are necessary.

State of the Art Game Programs

- Chess: DeepBlue beat world champion
  - Customized parallel hardware
  - Highly-tuned evaluation function developed with expert
  - Comprehensive opening and end-game databases
- Checkers:
  - Samuel's learning program eventually beat developer.
  - Chinook is world champion, previous champ held title for 40 years had to withdraw for health reasons.
- Othello: Programs best players (e.g. Iago).
- Backgammon: Neural-net learning program TDGammon one of world's top 3 players.
- Go: Branching factor of ~360 kills most search methods. Best programs still mediocre.