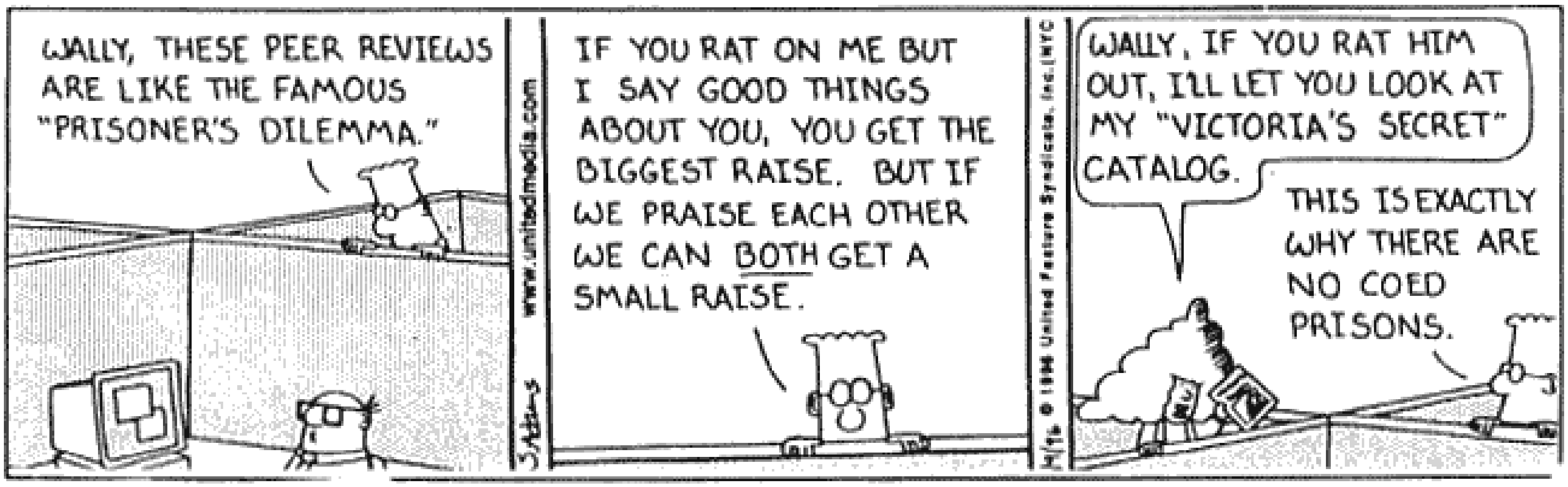
Adversarial Search
Minimax

CS4881 Artificial Intelligence

Jay Urbain, Ph.D.

Game Theory



WALLY, THESE PEER REVIEWS ARE LIKE THE FAMOUS "PRISONER'S DILEMMA."
www.unitedmedia.com
IF YOU RAT ON ME BUT I SAY GOOD THINGS ABOUT YOU, YOU GET THE BIGGEST RAISE. BUT IF WE PRAISE EACH OTHER WE CAN BOTH GET A SMALL RAISE.
WALLY, IF YOU RAT HIM OUT, I'LL LET YOU LOOK AT MY "VICTORIA'S SECRET" CATALOG.
THIS IS EXACTLY WHY THERE ARE NO COED PRISONS.

Dilbert Teaches Game Theory

Outline

- Optimal decisions
- Minimax
- α-β pruning
- Imperfect, real-time decisions

Adversarial Search

- Multiagent environments
  - Any given agent needs to consider the actions of other agents and how they affect their welfare.
- Unpredictability of other agents can introduce many possible contingencies into the agent's problem-solving process.
- Distinguish *cooperative* and *competitive* multi-agent environments.
- Competitive environments in which the agents' goals are in conflict - give rise to *adversarial search problems* – often called *games*.

Games in AI

- Game theory (Economics) – views multiagent environments a game regardless of whether the agents are *cooperative* or *competitive*.

- In AI, "games" are usually:
  - Deterministic
  - Turn-taking
  - Two-player
  - Zero-sum games of perfection

- In our agent environment framework this means:
  - Deterministic (Strategic)
  - Fully observable
  - Multiagent (2) where actions must alternate
  - Utility values at end of game are equal and opposite {win,loose}, {1,0}

Games vs. search problems

- **Unpredictable** opponent → specifying a move for every possible opponent reply.

- **Time limits** → unlikely to find goal, must approximate.

Optimal Decisions in Games

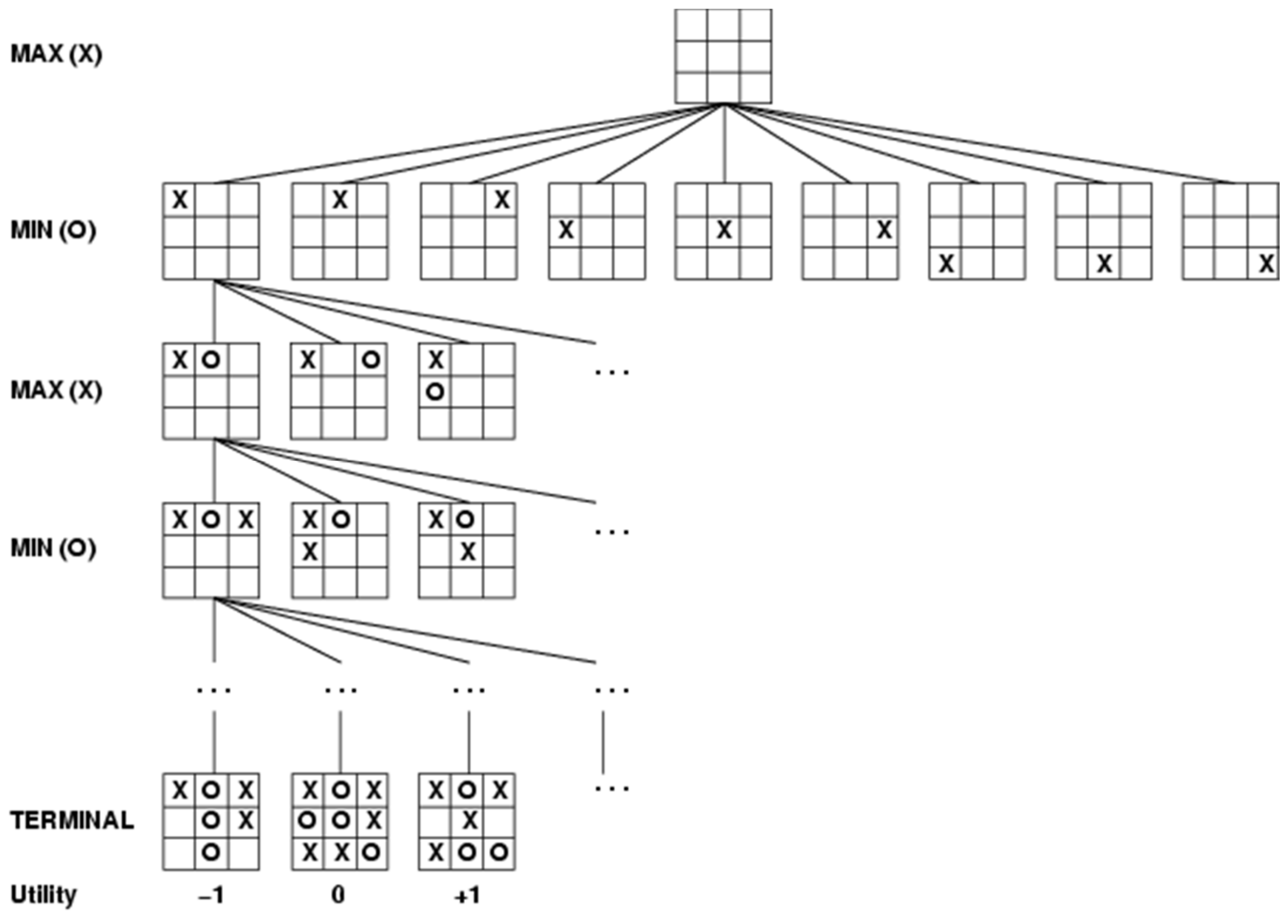
Game search problem:

- Initial state –
  - board position, player to move.
- Successor function –
  - returns list of {move, state} pairs, each indicating a legal move and the resulting state.
- Terminal test –
  - determines when the game is over, i.e., reach *terminal state*.
- Utility function (objective function) –
  - numeric value for terminal states {+1, 0, -1}

Optimal Decisions in Games

Initial state + the *legal moves* for each side define the *game tree*

Game tree (2-player, deterministic, turns)



Optimal Strategies

- Optimal solution is a sequence of moves leading to a goal state.
- Unfortunately, *Max* and *Min*, have equal and opposite objectives.
- Max must find a *contingent strategy*, which specifies Max's move in the initial state.
- Max moves in *successive states* resulting from every possible response by Min.
- *What approach can we use?*

Minimax - Game Theory

- In the theory of *simultaneous games*, a ***minimax*** strategy is a *mixed strategy* which is part of the solution to a zero-sum game.
- In zero-sum games, the minimax solution is the same as the ***Nash equilibrium***:
  - A *solution concept* of a game involving two or more players.
  - Each player is assumed to know the equilibrium strategies of the other players.
  - No player has anything to gain by changing only his own strategy unilaterally.

Minimax Theorem

- For every two-person, zero-sum game with finitely many strategies, there exists a value V and a mixed strategy for each player, such that:
  - Given player 2's strategy, the best payoff possible for player 1 is V, and
  - Given player 1's strategy, the best payoff possible for player 2 is $-$V.

- Equivalently, Player 1's strategy guarantees him a payoff of V regardless of Player 2's strategy, and similarly Player 2 can guarantee himself a payoff of $-V$.

Minimax Theorem

- The name minimax arises because each player minimizes the maximum payoff possible for the other—since the game is zero-sum, he also maximizes his own minimum payoff.

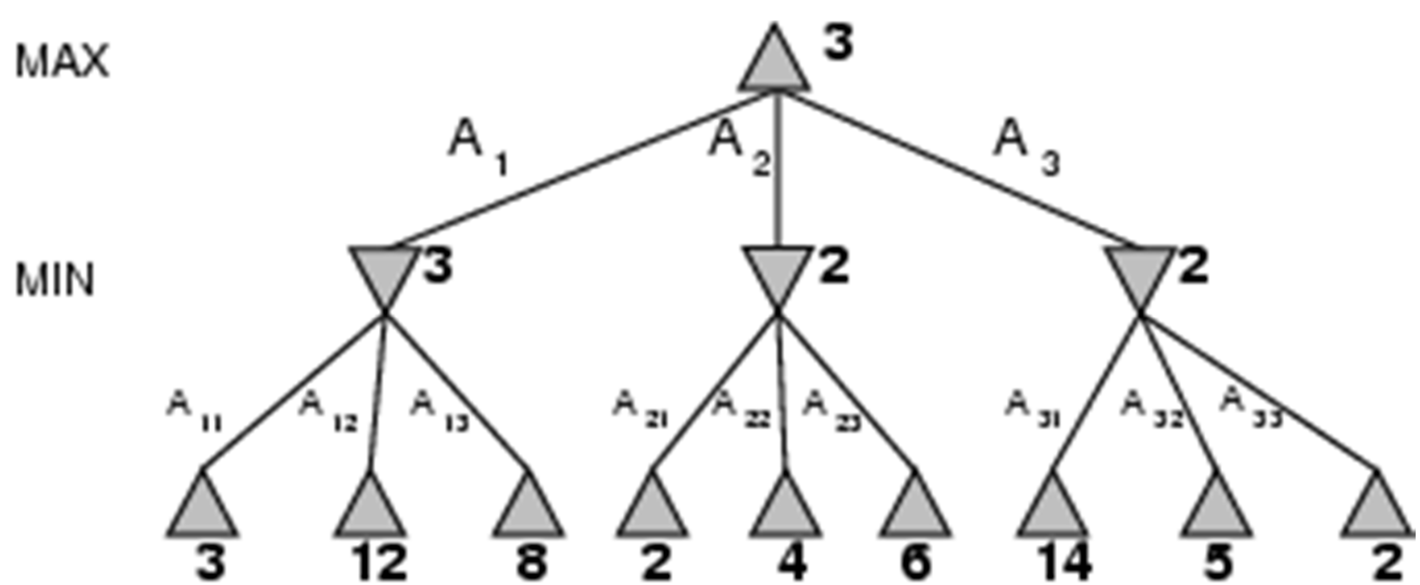
- Theorem established by *John von Neumann*, who is quoted as saying:
  - "As far as I can see, there could be no theory of games … without that theorem … I thought there was nothing worth publishing until the *Minimax Theorem* was proved".

Optimal Strategies (cont.)

- Optimal strategy leads to outcomes at *least* as good as any other strategy when one is playing an *infallible opponent.*
- Will do even better with fallible opponent.
- First, consider *optimal strategy*, even though its computationally problematic for all but the simplest games!
- Given a game tree, the optimal strategy can be determined by examining the *minimax* value of each node.

Minimax

- Perfect play for deterministic games
- Idea: choose move with highest *minimax* value
  = best achievable payoff against best play
- E.g., 2-ply game:



Minimax algorithm

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

Properties of minimax

- Complete? Yes! (if tree is finite)
- Optimal? Yes! (against an *optimal* opponent)
- Time complexity? $O(b^m)$ – ouch!
- Space complexity? $O(bm)$ (depth-first exploration)

- For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
  → exact solution completely infeasible: 35^{100}

- *Note: minimax can easily be extended to > 2 players.*

Alpha-beta Pruning

Huge number of games states - $O(b^m)$

- Can't eliminate exponent, but we can cut it in ~half.

- Trick:
  - Possible to compute the correct *minimax* decision without looking at *every* node in the game tree.
  - Using idea of pruning (from informed search) to eliminate (potentially) large parts of tree.

- *Alpha-beta* pruning
  - Apply to *minimax* game tree
  - Prune away branches that cannot possibly influence the final decision.

Alpha-beta Pruning (cont.)

Consider simplification of the formula for minimax-value:

Minimax-value(root)

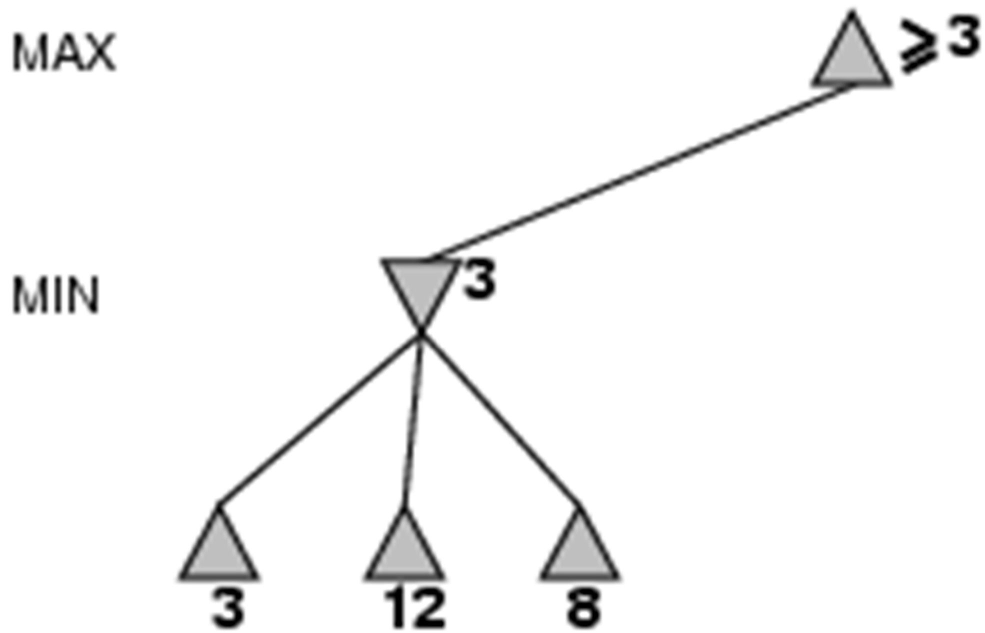
= max (min(3,12,8), min(2,x,y),min(14,5,2)) // leaf nodes

= max (3, min (2,x,y),2)

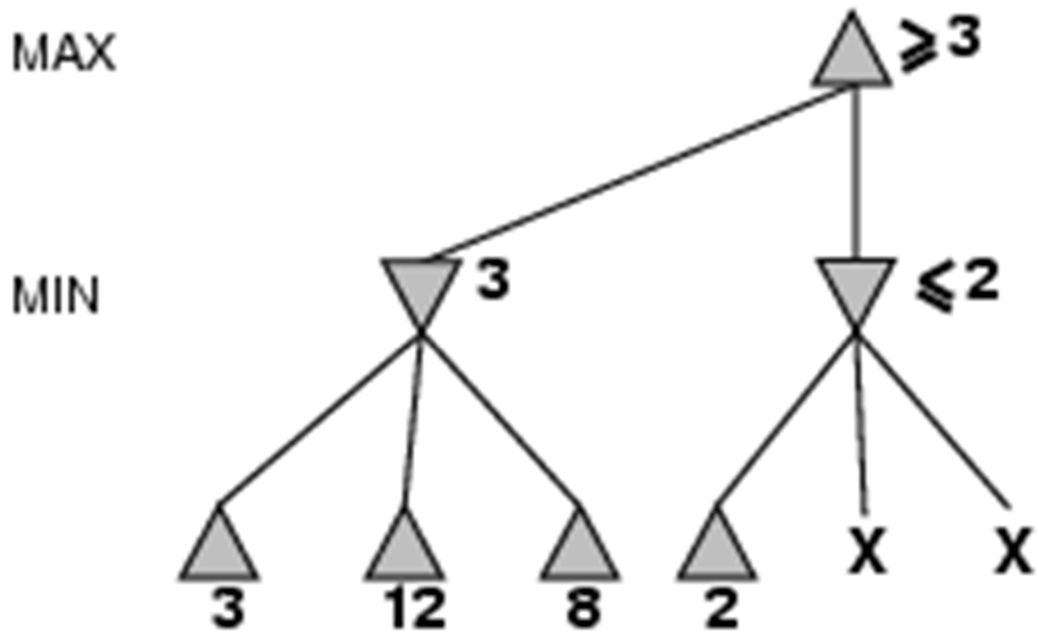
= max (3,z,2) where z<=2

= 3

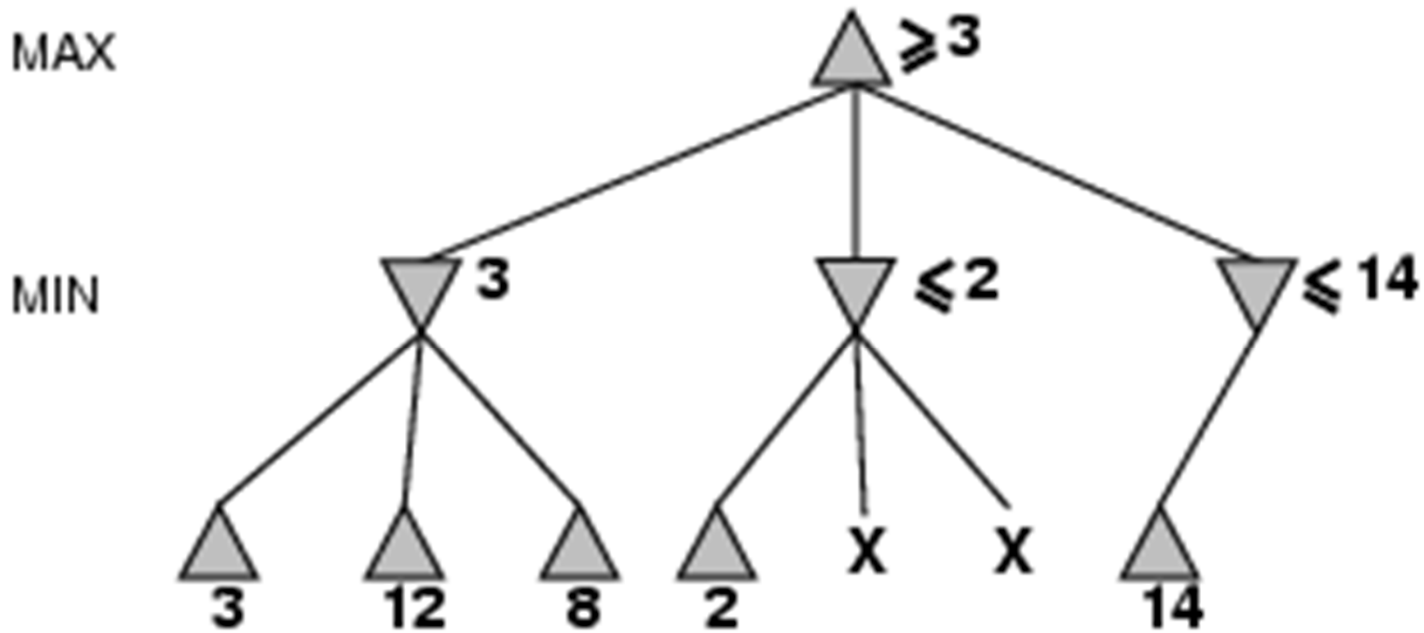
α-β pruning example



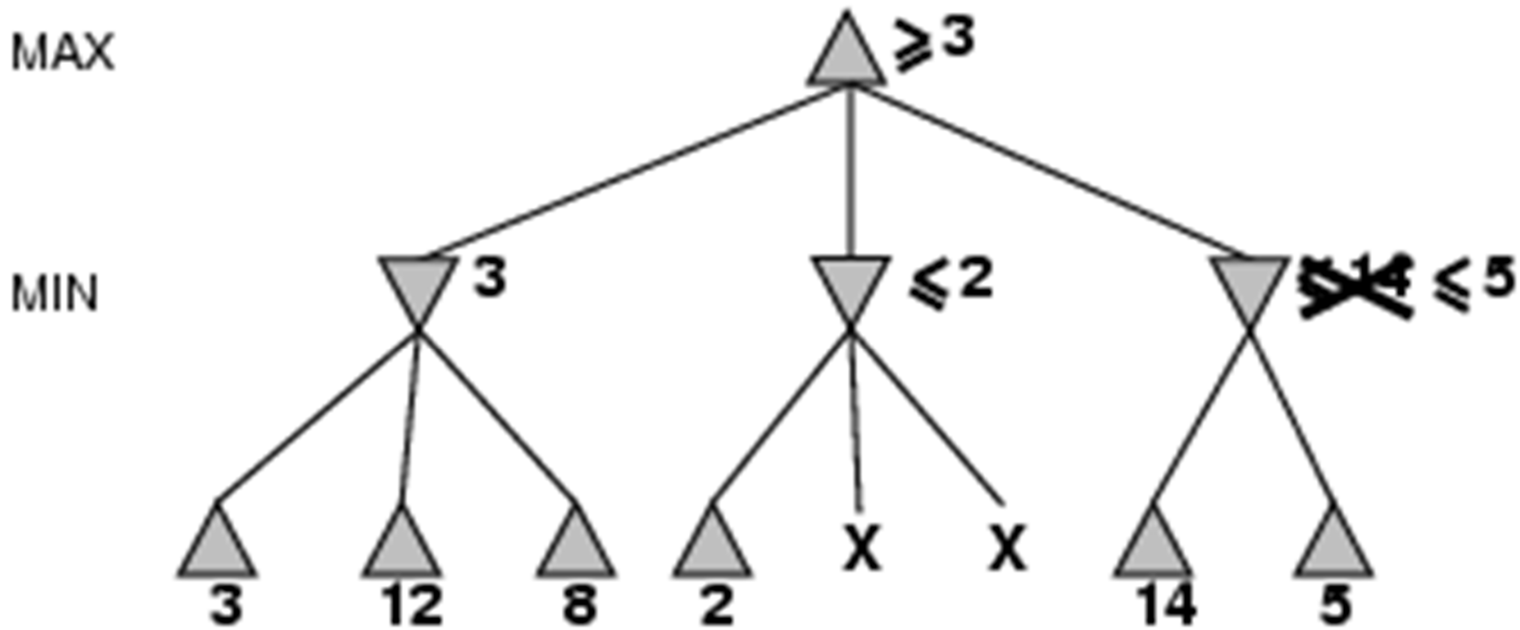
α-β pruning example



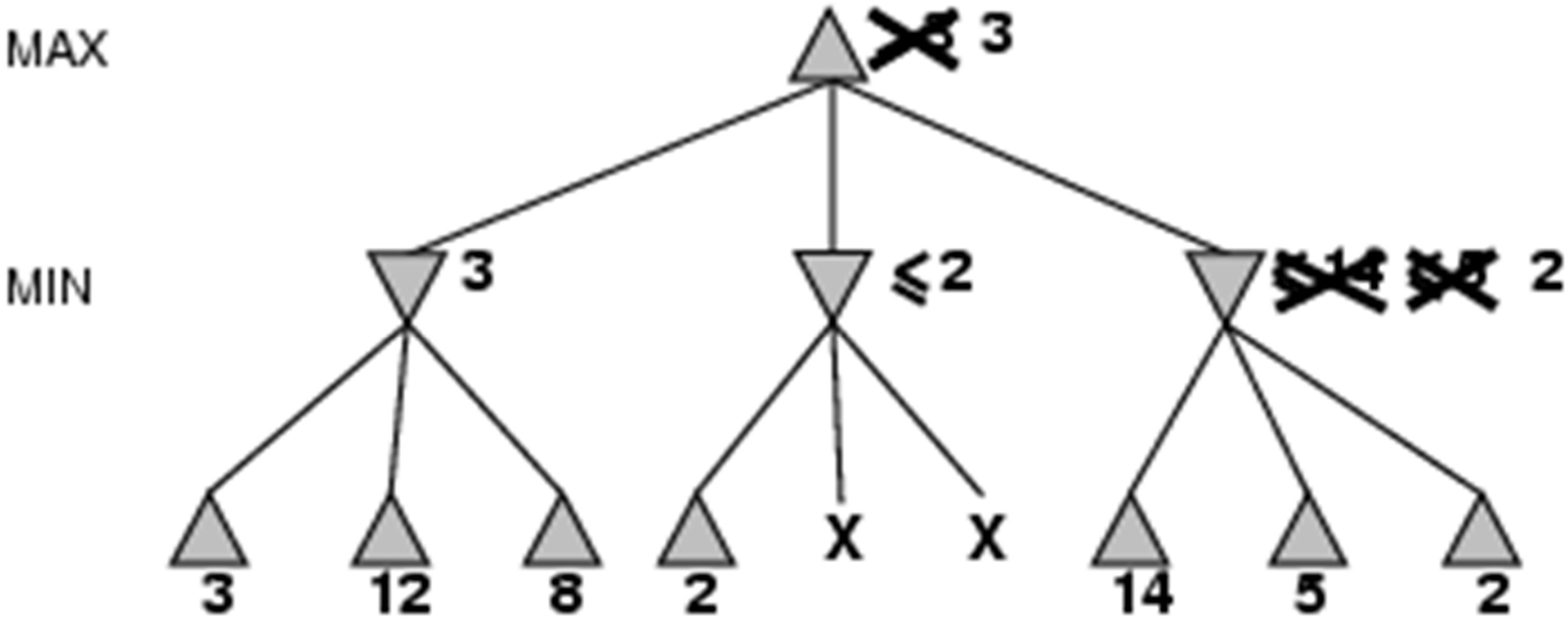
α-β pruning example



α-β pruning example



α-β pruning example

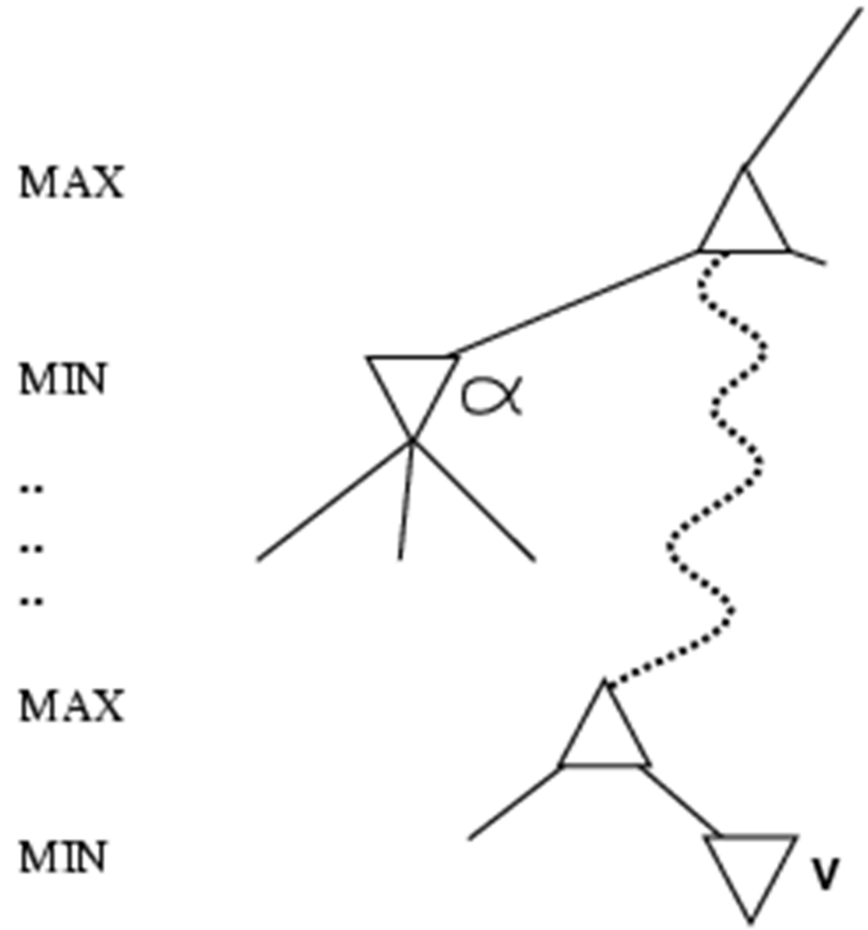


Properties of α-β

- Pruning does not affect the final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity = $O(b^{m/2})$
- Example of the value of reasoning about which computations are relevant (*meta-reasoning*)

α-β?

- α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for *max*
- If *v* is worse than α, *max* will avoid it
  → prune that branch
- Define β similarly for *min*



The α-β algorithm

function ALPHA-BETA-SEARCH(*state*) **returns** *an action*
  inputs: *state*, current state in game

  $v \leftarrow$ MAX-VALUE(*state*, $-\infty, +\infty$)
  return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE(*state*, α, β) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

The α-β algorithm

function MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow +\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

Resource limits

Suppose we have 100 seconds, explore 10^4 nodes/sec
→ 10^6 nodes per move

Standard approach:

- cutoff test:
  e.g., depth limit (perhaps add *quiescence* search)
- evaluation function (heuristic)
  = estimated desirability of position

Resource limits

Quiescence search –

- Select values unlikely to exhibit wild swings in value.
- E.g. chess - in positions where favorable captures can be made are *not* quiescent for an evaluation function that just counts material.
- Nonquiescent positions can be expanded further until quiescent positions are reached.

Horizon affect –

- Arises when the program is facing a move by the opponent that causes serious damage and is ultimately unavoidable.
- Hard to avoid.
- E.g., pawn making it across the board & being replaced by queen.
- Can use *singular exclusions* – move that is clearly better than any other move at the current position. Can go beyond normal depth limit since its branching factor = 1. Very effective in chess.

Evaluation functions

- For chess, typically linear weighted sum of features

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

- e.g., $w_1 = 9$ with

  $f_1(s)$ = (number of white queens) – (number of black queens), etc.
- Can use Machine Learning to optimize your *weights.*
- *More sophisticated – learn moves!*

Cutting off search

MinimaxCutoff is identical to *MinimaxValue* except

1. *Terminal* is replaced by *Cutoff*
2. *Utility* is replaced by *Eval*

Does it work in practice?

$b^m = 10^6$, $b=35 \rightarrow m=4$

4-ply lookahead is a hopeless chess player!

- 4-ply ≈ human novice
- 8-ply ≈ typical PC, human master
- 12-ply ≈ Deep Blue, Kasparov

Deterministic games in practice

- Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a pre-computed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of *444 billion* positions.

- Chess: Deep Blue defeated human world champion Garry Kasparov in a six-game match in 1997. Deep Blue searches *200 million* positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.

- Othello: human champions refuse to compete against computers, who are too good.
- In 2008, thanks to an efficient message-passing parallelization, **MoGo** won one game (out of three) against Catalin Taranu, 5th dan pro, in 9x9 with standard time settings (30 minutes per side).

Summary

- Reviewed games to understand what optimal play means and to understand how to play well in practice.
- Game can be defined by the *initial state* (how the board is set up), legal *actions* in each state, a *terminal test* (game over), and a *utility function* that applies to terminal states.
- For two-player, zero-sum games with perfect information, the *minimax* algorithm can select optimal moves using a depth-first enumeration of the game tree.
- *Alpha-beta* search computes the same optimal move as minimax, but achieves much greater efficiency by eliminating subtrees that are provably irrelevant.
- Usually not feasible to consider the whole game tree, need to *cut the search* off at some point and apply *eval* function that gives a *guesstimate* of the utility of a state.

Summary

- Games of chance can be handled by an extension to minimax algorithm that values a *chance* node by taking the *average utility* of all its children nodes *weighted by the probability* of each child.

- Optimal play in games of imperfect information (bridge) requires reasoning about the current and future belief states of each player. Can average value of an action over each possible action.

- Programs can match or beat the best human players in many games.

- Games are fun to work on!
  - Illustrate several important points about AI
  - perfection is unattainable → must approximate
  - good idea to think about what to think about