CmpE 540
Principles of Artificial Intelligence

Pınar Yolum
pinar.yolum@boun.edu.tr

Department of
Computer Engineering
Boğaziçi University

Game Playing

Chapter 6 (Sections 1—4)

(Based mostly on the course slides from http://aima.cs.berkeley.edu/ and http://www.cmpe.boun.edu.tr/~akin/)

Outline

- **Game playing**
  - The minimax algorithm
  - Resource limitations
  - alpha-beta pruning

Games vs. search problems

- "Unpredictable" opponent → solution is a strategy specifying a move for every possible opponent reply
- Time limits → unlikely to find goal, must approximate
- Plan of attack:
  - Algorithm for perfect play (Zermelo, 1912; Von Neumann, 1944)
  - Finite horizon, approximate evaluation (Zuse, 1945; Wiener, 1948; Shannon, 1950)
  - First chess program (Turing, 1951)
  - Machine learning to improve evaluation accuracy (Samuel, 1952-57)
  - Pruning to allow deeper search (McCarthy, 1956)

Types of games

	deterministic	chance
perfect information	chess, checkers, go, othello, rock–paper–scissors	backgammon monopoly
imperfect information	battleships, kriegspiel, stratego	bridge, poker, scrabble nuclear war

What kind of games?

- Abstraction: To describe a game we must capture every relevant aspect of the game. Such as :
  - Chess
  - Tic-tac-toe
  - …
- Accessible environments: Such games are characterized by perfect information
- Search: game-playing then consists of a search through possible game positions
- Unpredictable opponent: introduces uncertainty thus game-playing must deal with contingency problems

Searching for the next move

- **Complexity:** many games have a huge search space
  - **Chess:** *b = 35, m=100* $\Rightarrow$ *nodes = 100* ^{35}
    if each node takes about 1 ns to explore
    then each move will take about ***10*** ^{50} **millenniums**
    to calculate.
- **Resource (e.g., time, memory) limit:** optimal solution not feasible/possible, thus must approximate
- **Pruning:** makes the search more efficient by discarding portions of the search tree that *obviously* cannot improve quality of result.
- **Evaluation functions:** heuristics to evaluate utility of a state without exhaustive search.

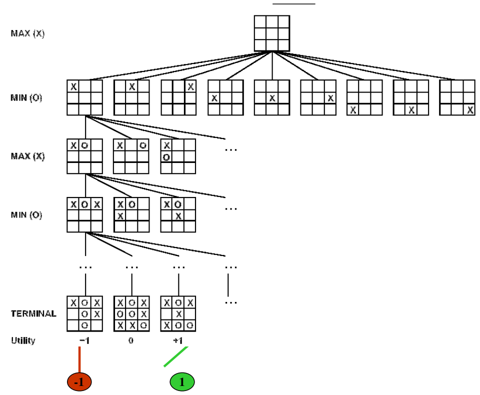
Two-player games

- A game formulated as a search problem:
  - Initial state: ?
  - Operators: ?
  - Terminal state: ?
  - Utility function: ?

Two-player games

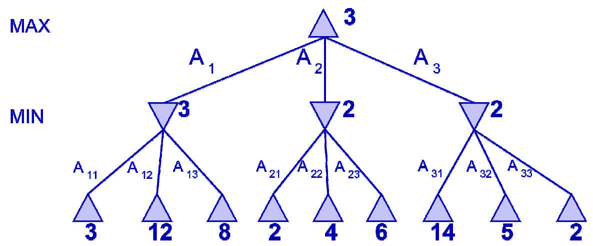
- A game formulated as a search problem:
  - Initial state: board position and turn
  - Operators: definition of legal moves
  - Terminal state: conditions for when game is over
  - Utility function: a numeric value that describes the outcome of the game.
    e.g., -1, 0, 1 for loss, draw, win.

Example: Tic-Tac-Toe



The minimax algorithm

- Perfect play for deterministic, perfect-information games
- Idea: choose move to position with highest minimax value = best achievable utility against best play
- e.g., 2-ply game:



Minimax Algorithm

```
function MINIMAX-DECISION(state) returns an action
   inputs: state, current state in game
   return the a in ACTIONS(state) maximizing MIN-VALUE(RESULT(a, state))

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
```

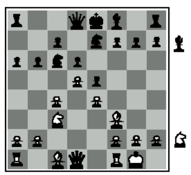
Properties of Minimax

- Completeness: if tree is finite (chess has specfiic rules for this)
- Optimality: Yes, against an optimal opponent.
- Time complexity: $O(b^m)$
- Space complexity: O(bm) (depth-first exploration)
- For chess, b ≈35, m ≈ 100 for “reasonable" games
  ∴ exact solution completely infeasible
- But do we need to explore every path?

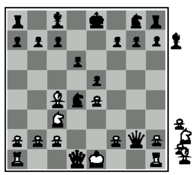
Move evaluation without complete search

- Complete search is too complex and impractical
- Evaluation function: evaluates value of state using heuristics and cuts off search
- New MINIMAX:
  - CUTOFF-TEST: cutoff test to replace the termination condition (e.g., deadline, depth-limit, etc.)
  - EVAL: evaluation function to replace utility function (e.g., number of chess pieces taken)

Evaluation functions



Black to move
White slightly better



White to move
Black winning

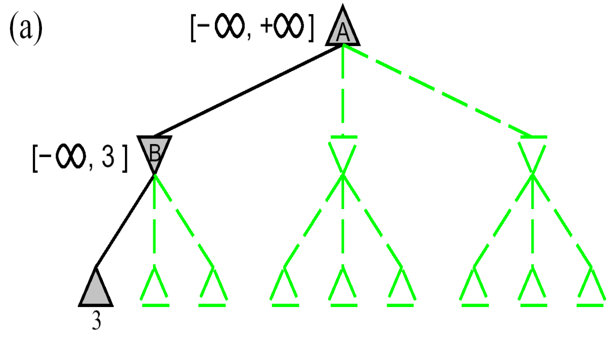
- **Weighted linear evaluation function:** to combine *n* heuristics
  $f = w_1 f_1 + w_2 f_2 + \ldots + w_n f_n$
- e.g, *w's* could be the values of pieces (1 for pawn, 3 for bishop etc.) *f's* could be the number of type of pieces on the board

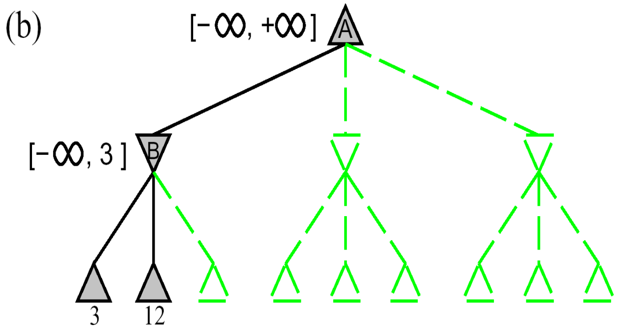
α-β pruning: search cutoff

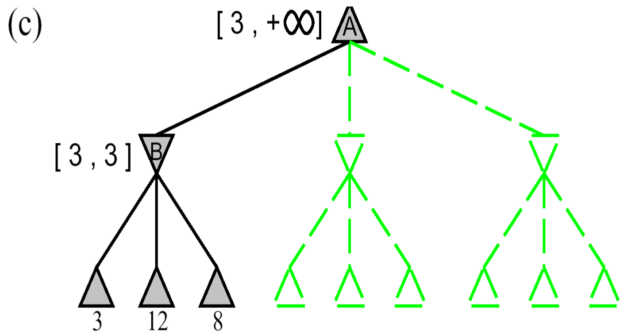
- **Pruning:** eliminating a branch of the search tree from consideration without exhaustive examination of each node
- **α-β pruning:** the basic idea is to prune portions of the search tree that cannot improve the utility value of the max or min node, by just considering the values of nodes seen so far.
- Does it work? Yes, it roughly cuts the branching factor from b to √b resulting in as double as far look-ahead than pure minimax

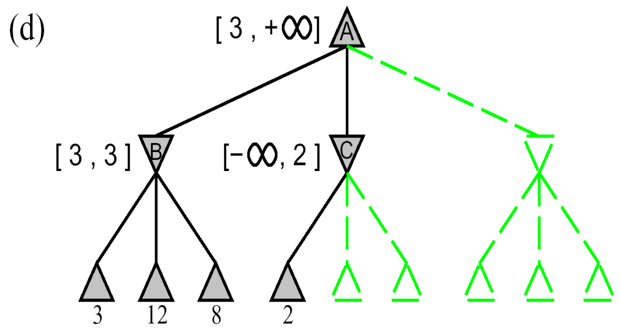
alpha-beta search

```
function ALPHA-BETA-DECISION(state) returns an action
   return the a in ACTIONS(state) maximizing MIN-VALUE(RESULT(a, state))

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 ← −∞
   for a, s in SUCCESSORS(state) do
      v ← MAX(v, MIN-VALUE(s, α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v

function MIN-VALUE(state, α, β) returns a utility value
   same as MAX-VALUE but with roles of α, β reversed
```

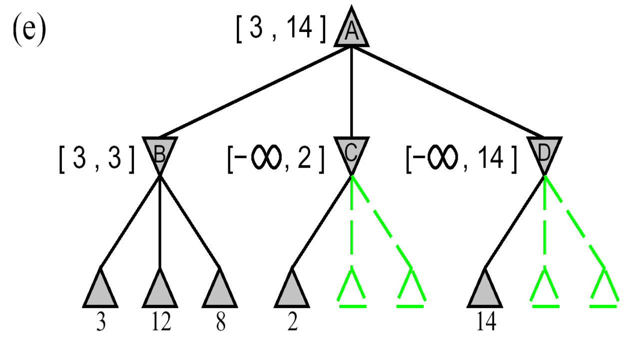





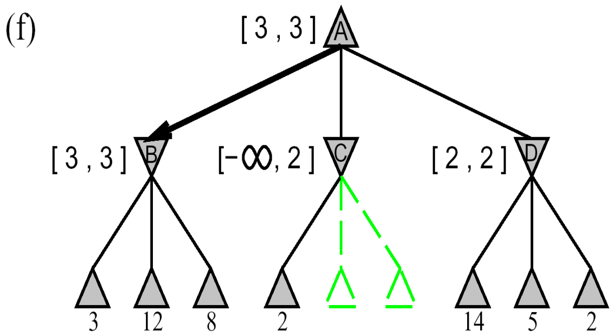




21

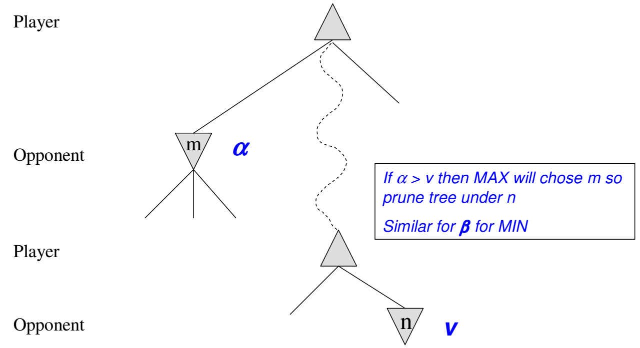


22



23

α-β pruning: general principle



24

State-of-the-art for deterministic games

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

- Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions. Exact solution imminent.

- Othello: Human champions refuse to compete against computers, who are too good.

- Go: Human champions refuse to compete against computers, who are too bad. In go, b > 300, so most programs use pattern knowledge bases to suggest plausible moves.