Planning and Scheduling: Adversarial Search

Prof. Dr.-Ing. Gerhard K. Kraetzschmar

Hochschule
Bonn-Rhein-Sieg

b-it Bonn-Aachen International Center for Information Technology

Acknowledgements

- These slides refer to Chapter 6 of the textbook:
  S. Russell and P. Norvig:
  Artificial Intelligence: A Modern Approach
  Prentice Hall, 2003, 2nd Edition (or more recent edition)
- These slides are an adaptation of slides by Min-Yen Kan
- The contributions of these authors are gratefully acknowledged.

Outline and Introduction

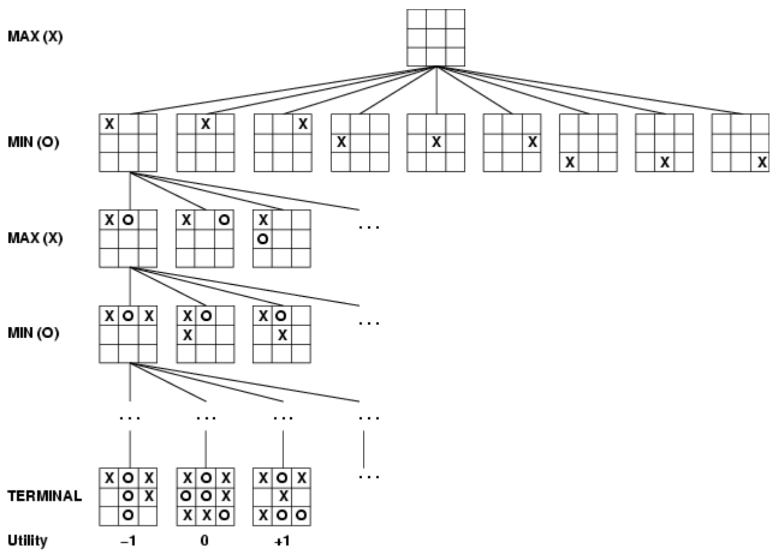
Outline

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

Games vs. search problems

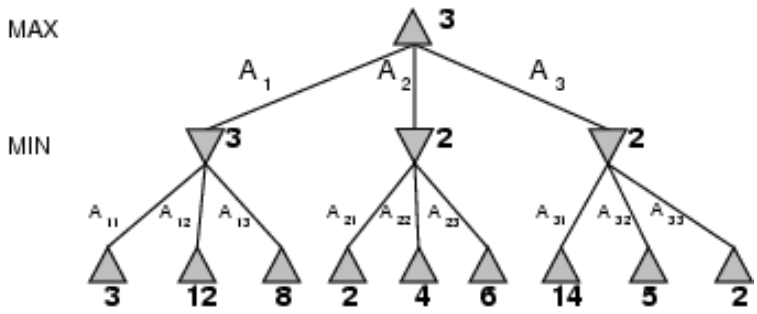
- "Unpredictable" opponent
  → specifying a move for every possible opponent reply
- Time limits
  → unlikely to find goal, must approximate

Game Tree (2-player, deterministic, turns)



Minimax

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



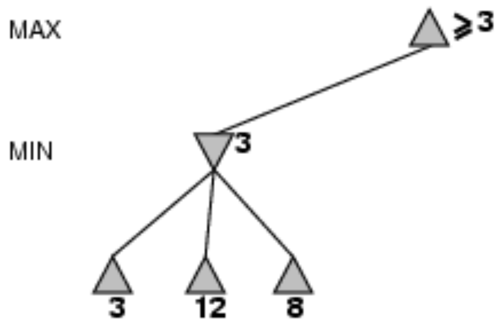
Minimax Algorithm

```
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
```

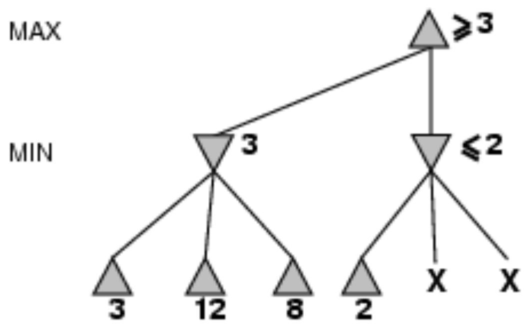
Properties of Minimax

- Complete? Yes (if tree is finite)
- Optimal? Yes (against an optimal opponent)
- Time complexity? $O(b^m)$
- Space complexity? $O(bm)$ (depth-first exploration)
- For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
  → exact solution completely infeasible

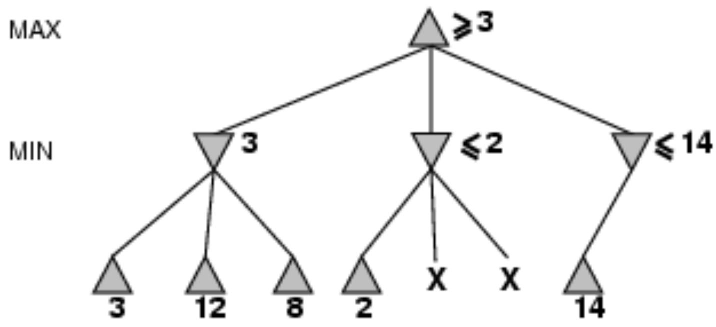
α-β Pruning Example



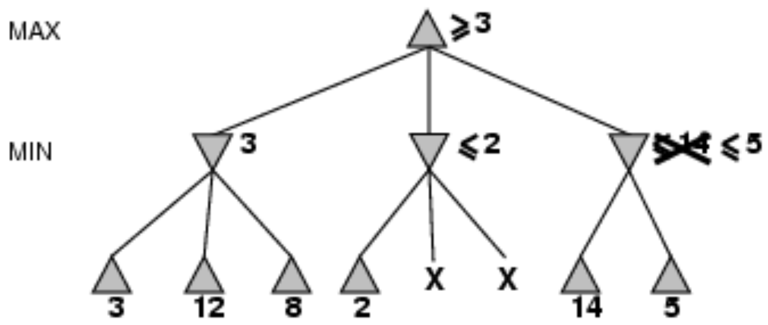
α-β Pruning Example



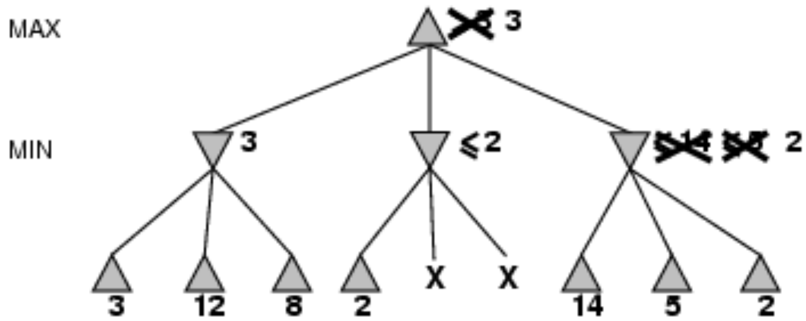
α-β Pruning Example



α-β Pruning Example



α-β Pruning Example

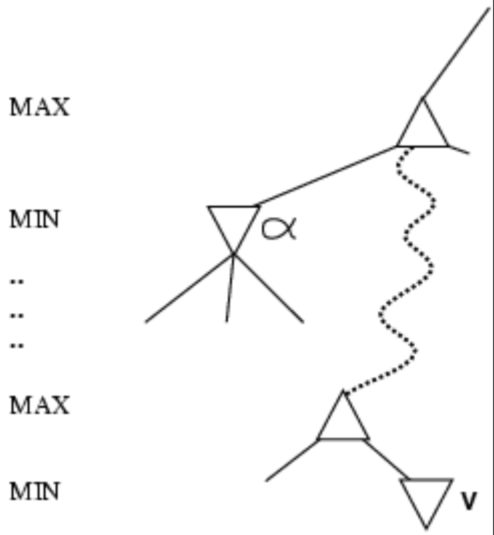


Properties of α-β

- Pruning does not affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity = $O(b^{m/2})$
  - → doubles depth of search
- A simple example of the value of reasoning about which computations are relevant (a form of **metareasoning**)

Why is it called 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 ← MAX-VALUE(state, −∞, +∞)
   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 ← −∞
   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
```

Hochschule Bonn-Rhein-Sieg

The α-β Algorithm

```
function MIN-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 ← MIN(v, MAX-VALUE(s, α, β))
      if v ≤ α then return v
      β ← MIN(β, v)
   return v
```

Hochschule Bonn-Rhein-Sieg

Resource Limits

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

Standard approach:

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

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.

Cutting Off Search

MinimaxCutoff is identical to MinimaxValue except

- Terminal? is replaced by Cutoff?
- 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 precomputed 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.

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.

Hochschule Bonn-Rhein-Sieg
© 2009 Gerhard K. Kraetzschmar

Summary

- Games are fun to work on!
- They illustrate several important points about AI
- Perfection is unattainable → must approximate
- Good idea to think about what to think about