Artificial Intelligence: Game playing

Peter Antal
antal@mit.bme.hu

Outline

- What are games?
- Optimal decisions in games
  - Which strategy leads to success?
- $\alpha-\beta$ pruning
- Games of imperfect information
- Games that include an element of chance

What are and why study games?

- Games are a form of *multi-agent environment*
  - What do other agents do and how do they affect our success?
  - Cooperative vs. competitive multi-agent environments.
  - Competitive multi-agent environments give rise to adversarial problems a.k.a. *games*
- Why study games?
  - Interesting subject of study because they are hard
  - Fun; historically entertaining, gaming industry(!): https://www.youtube.com/watch?v=NJarxpYyoFI
  - Chess as model organism: 50’s: A.Kronrod, 70’s D.Michie: “*Chess*, the *Drosophila* Melanogaster of Artificial Intelligence”

The #1 model organism in genetics: Fruit fly

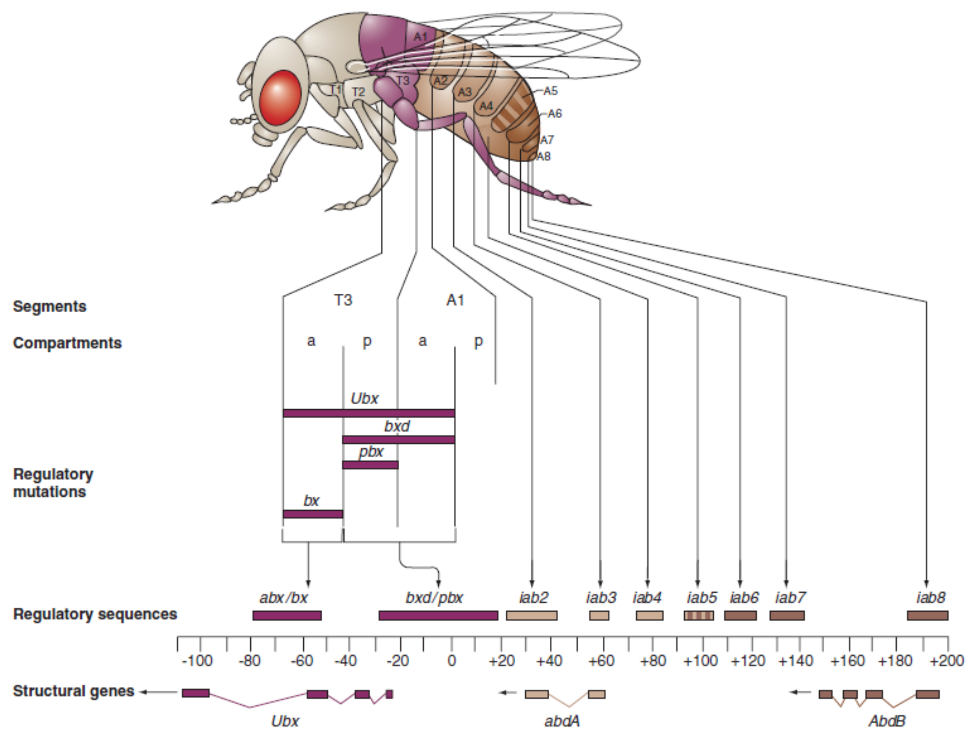


Figure D.28 **A close-up view of the 300 kb of the bithorax complex.** There are only three homeotic genes in this complex: *Ubx*, *abd-A*, and *Abd-B*. Many homeotic mutations such as *bx* and *pbx* affect regulatory regions that influence the transcription of one of these three genes in particular segments. For example, *bx* mutations (*at the left end of the complex*) prevent the transcription of *Ubx* in the anterior compartment of the third thoracic segment, while *iab-8* mutations (*far right*) affect the transcription of *Abd-B* in segment A8. Note that the order of these regulatory regions corresponds to the anterior-to-posterior order of segments in the animal.

http://www.drdobbs.com/parallel/computer-chess-the-drosophila-of-ai/184405171

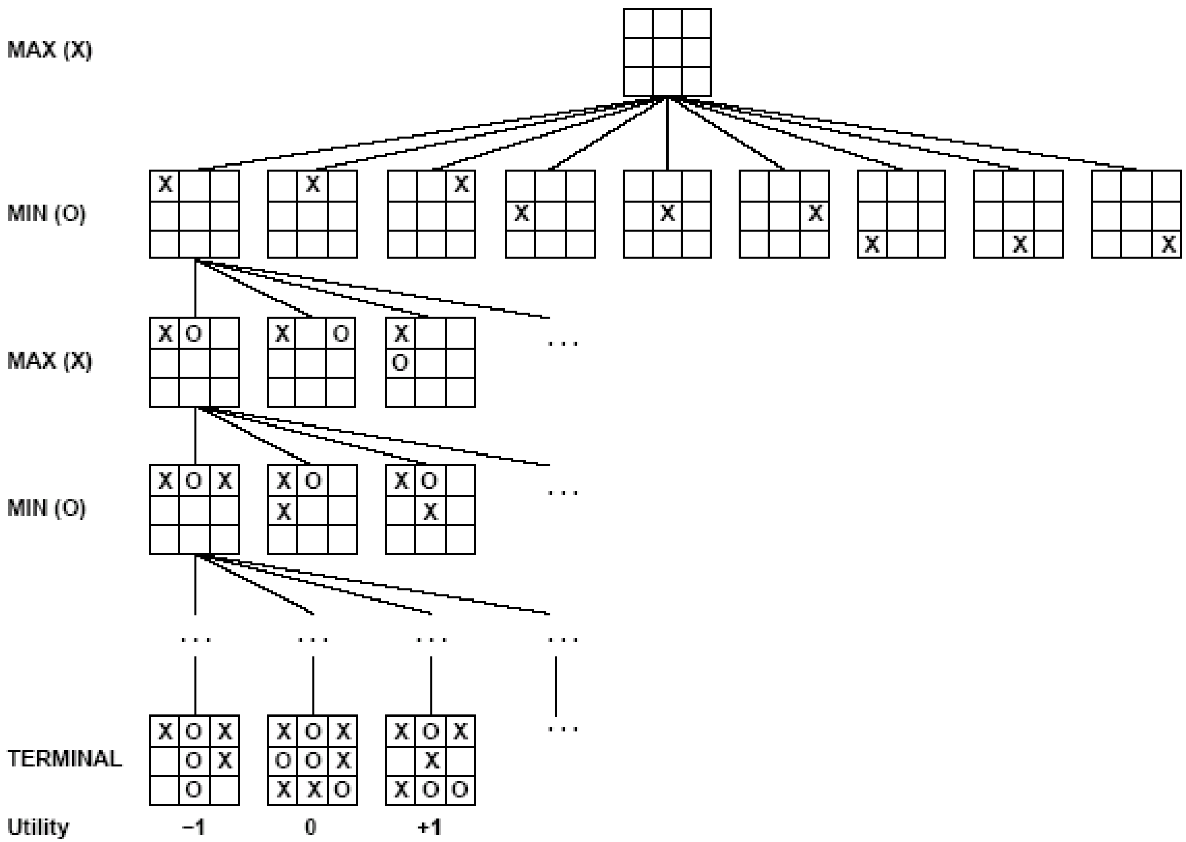
Relation of Games to Search

- Search
  - Solution is (heuristic) method for finding goal
  - Heuristics and CSP techniques can find *optimal* solution
  - Evaluation function (heuristics!): estimate of cost from start to goal through given node
  - Examples: path planning, scheduling activities
- Games
  - Solution is strategy (strategy specifies move for every possible opponent reply).
  - Time limits force an *approximate* solution
  - Evaluation function (heuristics): evaluate “goodness” of game position
  - Examples: chess, checkers, Othello, backgammon

Game setup

- Two players: MAX and MIN (Neumann:"game theory")
- MAX moves first and they take turns until the game is over. Winner gets award, looser gets penalty.
- Games as search:
  - Initial state: e.g. board configuration of chess
  - Successor function: list of (move,state) pairs specifying legal moves.
  - Terminal test: Is the game finished?
  - Utility function: Gives numerical value of terminal states. E.g. win (+1), loose (-1) and draw (0) in tic-tac-toe (next)
- MAX uses search tree to determine next move.

Partial Game Tree for Tic-Tac-Toe



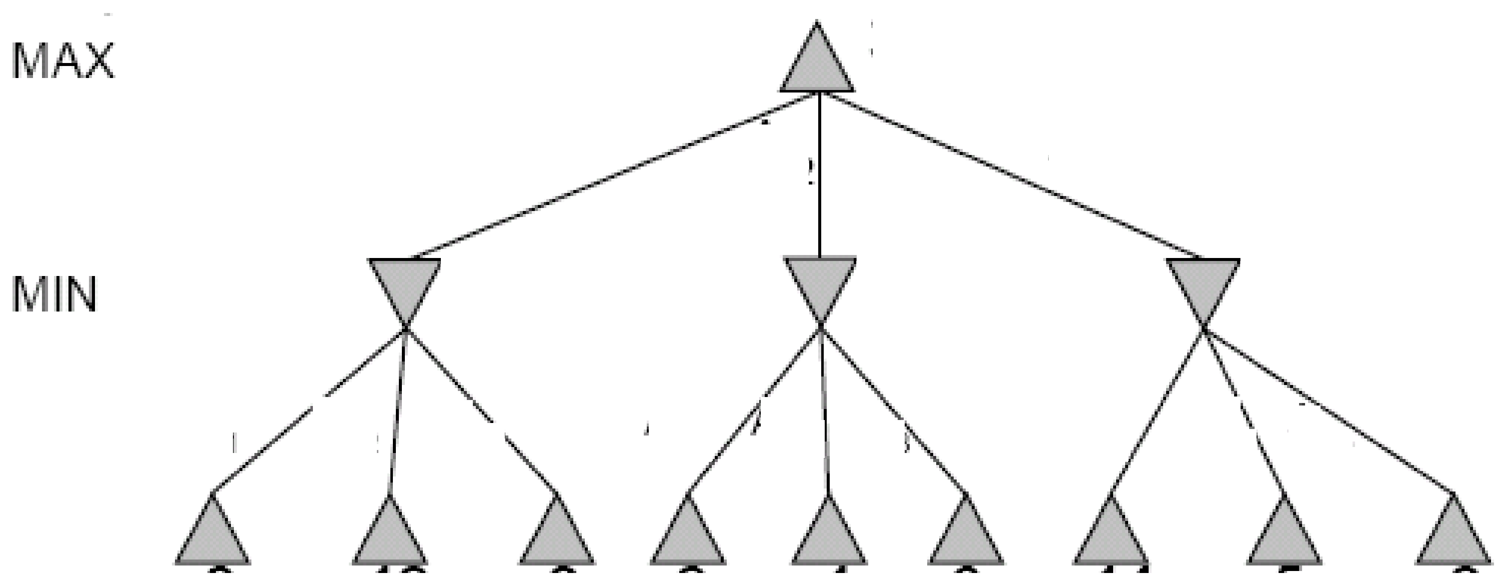
Optimal strategies

- Find the contingent *strategy* for MAX assuming an infallible MIN opponent.
- Assumption: Both players play optimally !!
- Given a game tree, the optimal strategy can be determined by using the minimax value of each node:

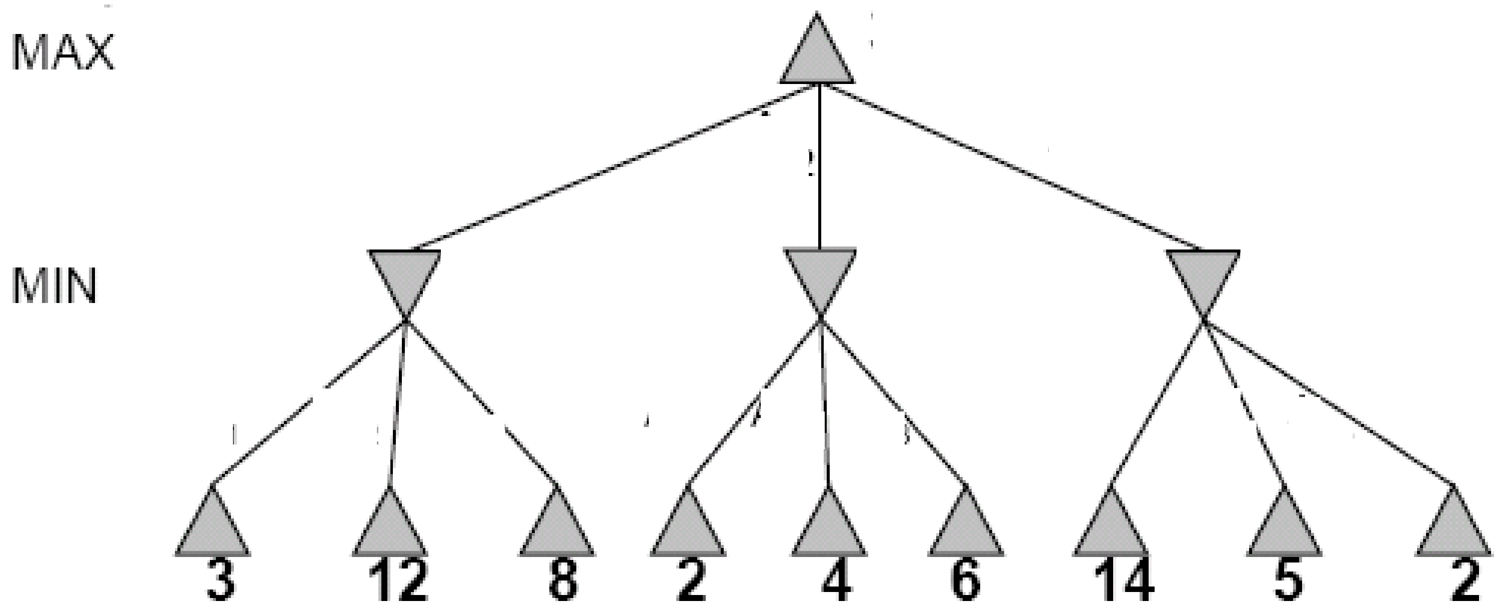
MINIMAX-VALUE(n)=

UTILITY(n)	If n is a terminal
$\max_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a max node
$\min_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a min node

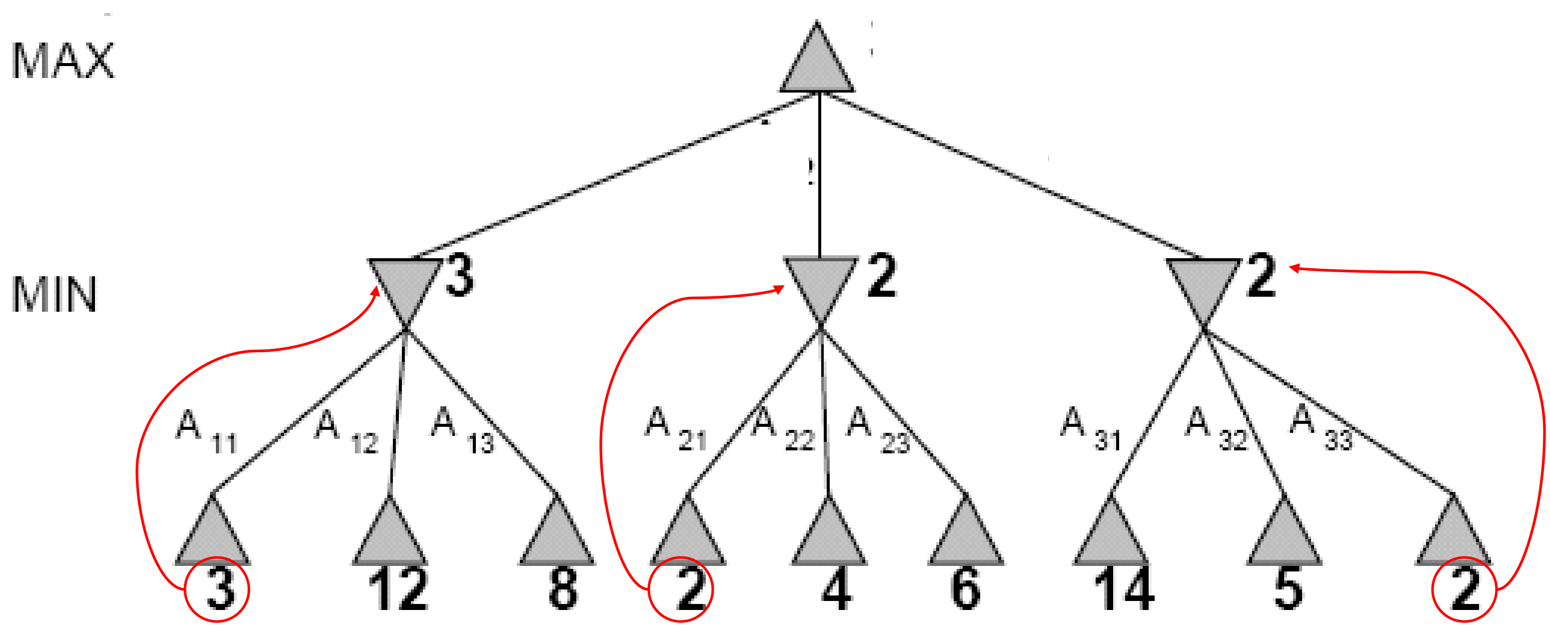
Two-Ply Game Tree



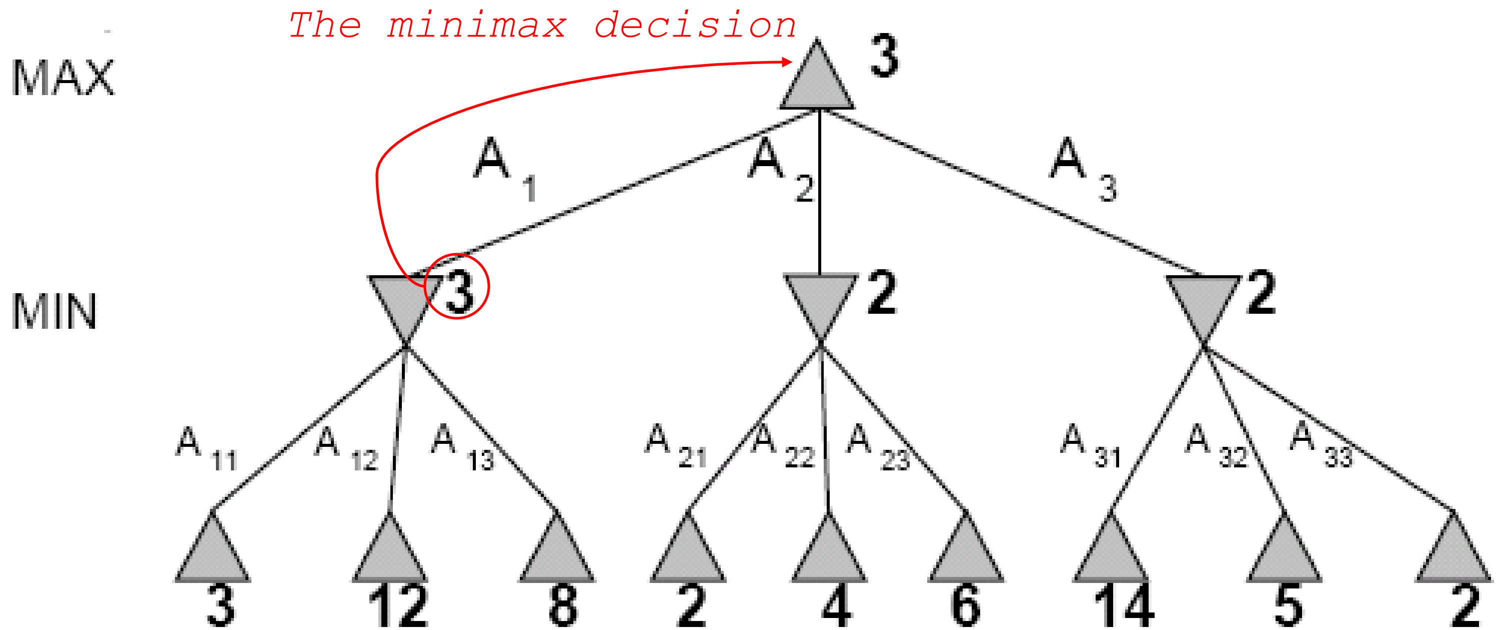
Two-Ply Game Tree



Two-Ply Game Tree



Two-Ply Game Tree



Minimax maximizes the worst-case outcome for max.

What if MIN does not play optimally?

- Definition of optimal play for MAX assumes MIN plays optimally: maximizes worst-case outcome for MAX.
- But if MIN does not play optimally, MAX will do even better. [Can be proved.]

Minimax Algorithm

function MINIMAX-DECISION(*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*) 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*

---

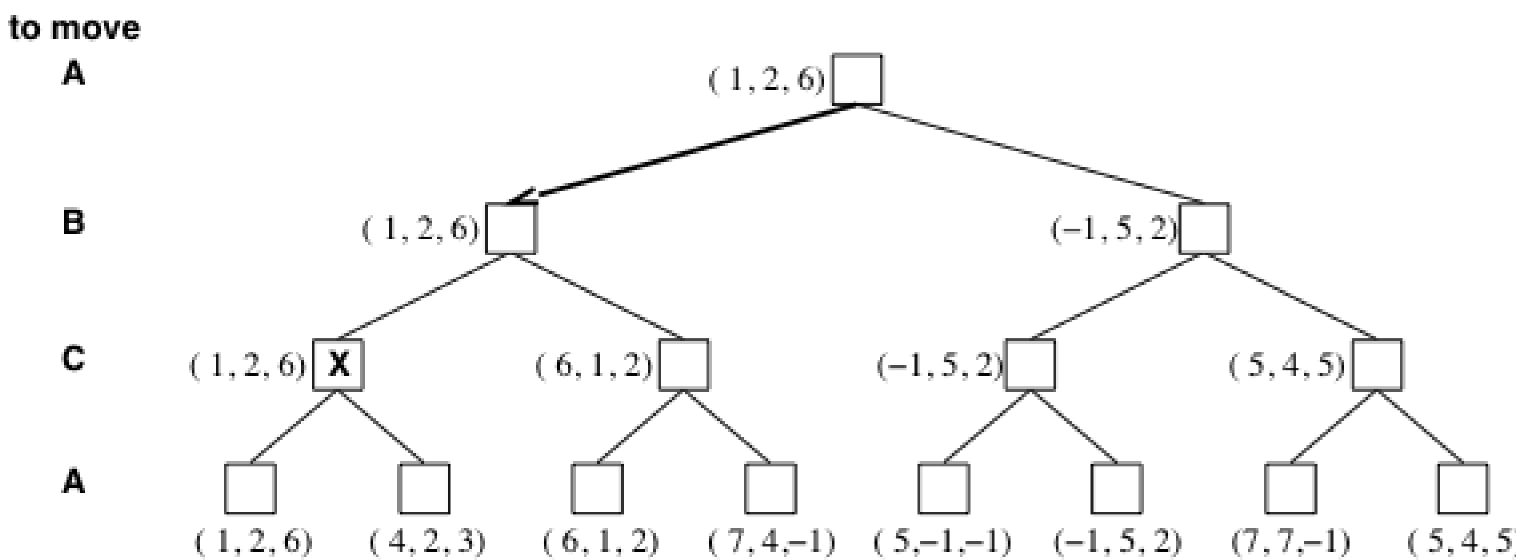
function MIN-VALUE(*state*) returns *a utility value*
  if TERMINAL-TEST(*state*) then return UTILITY(*state*)
  v ← ∞
  for *a,s* in SUCCESSORS(*state*) do
    v ← MIN(*v*,MAX-VALUE(*s*))
  return *v*

Properties of Minimax

Criterion	Minimax
Complete?	Yes ☺
Time	$O(b^m)$ ☹
Space	O(bm) ☺
Optimal?	Yes* ☺

Multiplayer games

- Games allow more than two players
- Single minimax values become vectors

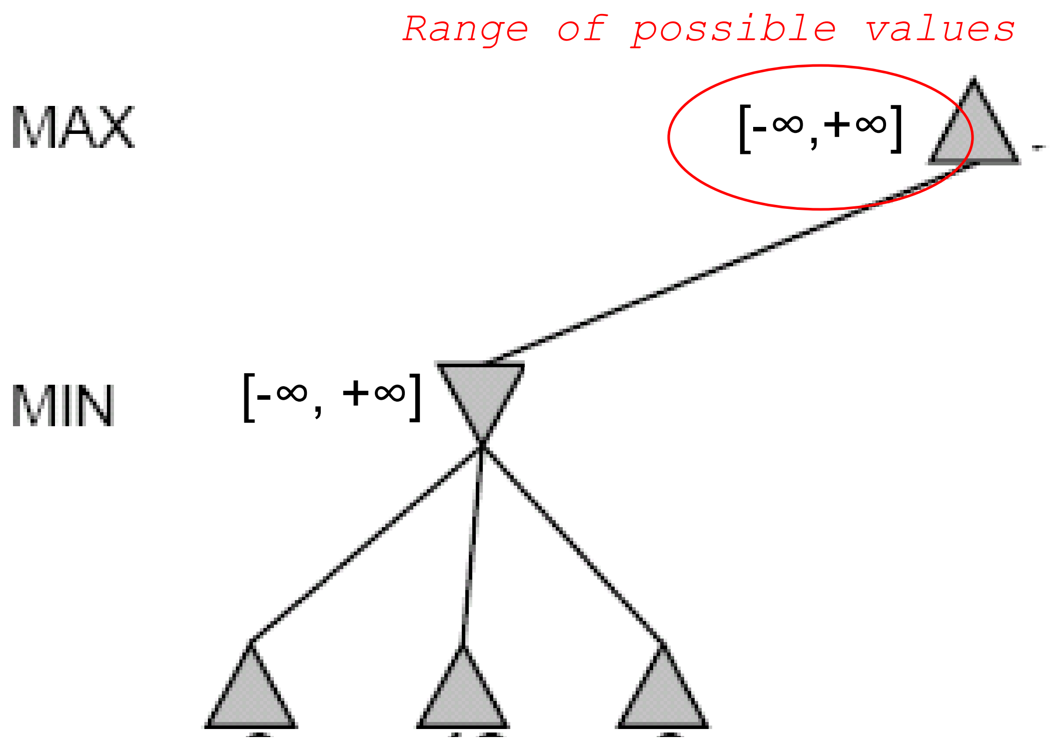


Problem of minimax search

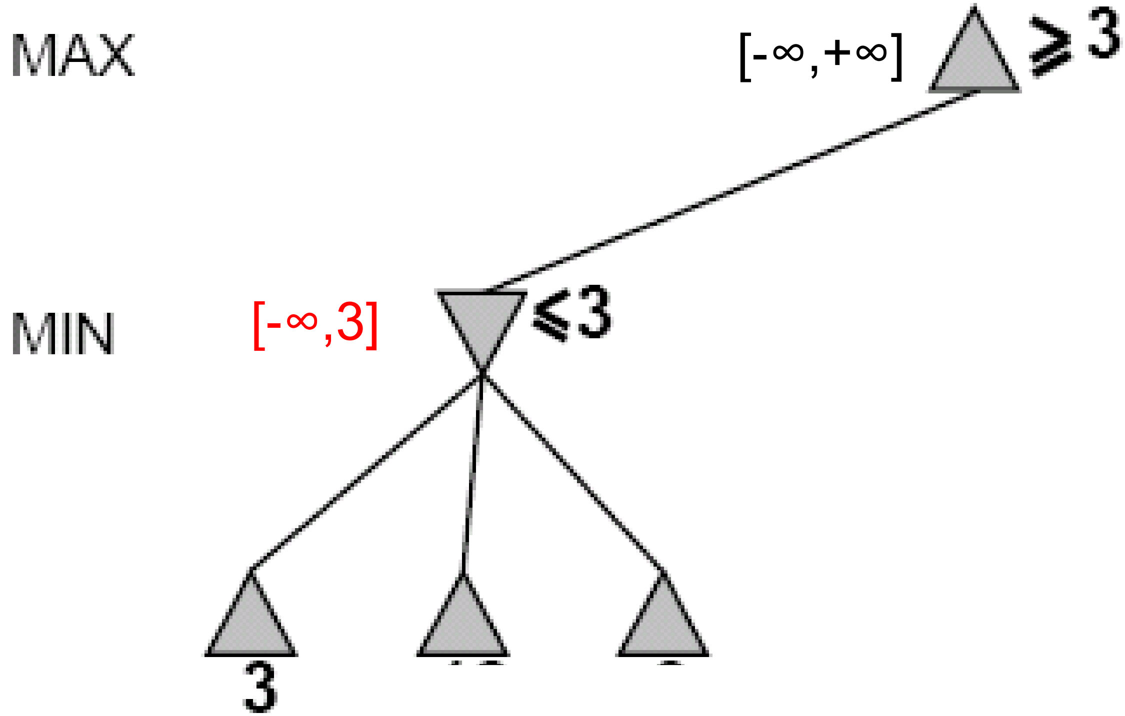
- Number of games states is exponential to the number of moves.
  - Solution: Do not examine every node
  - ==> Alpha-beta pruning
    - Alpha = value of best choice found so far at any choice point along the MAX path
    - Beta = value of best choice found so far at any choice point along the MIN path
- Revisit example …

Alpha-Beta Example

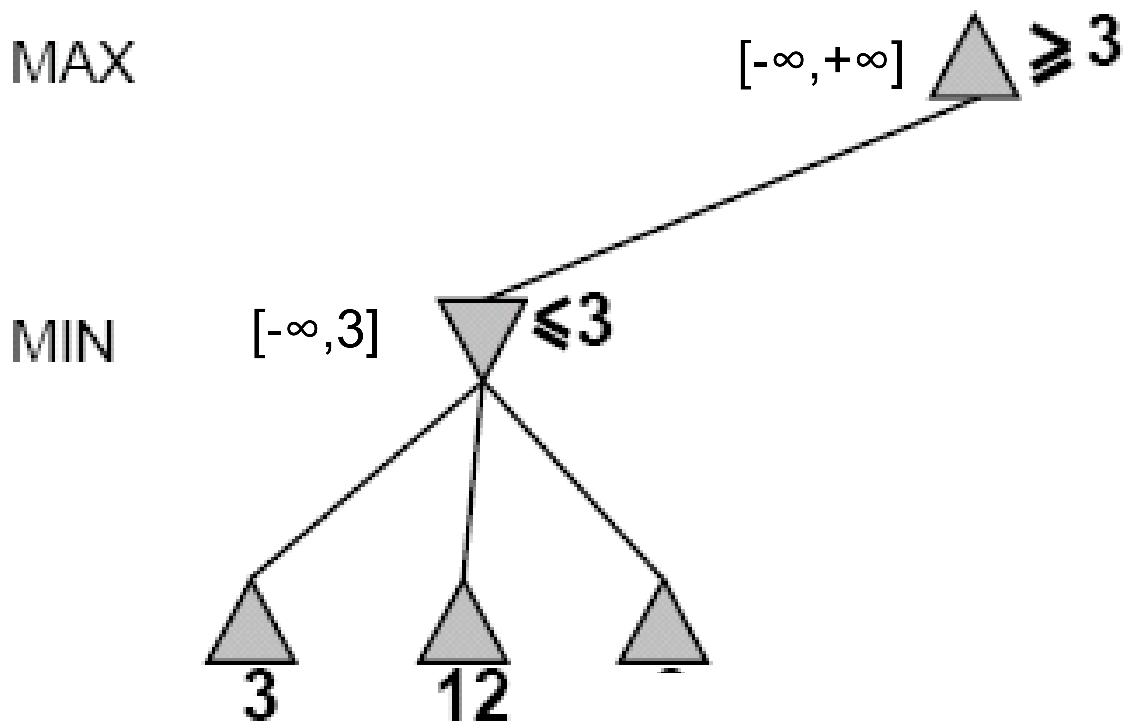
Do DF-search until first leaf



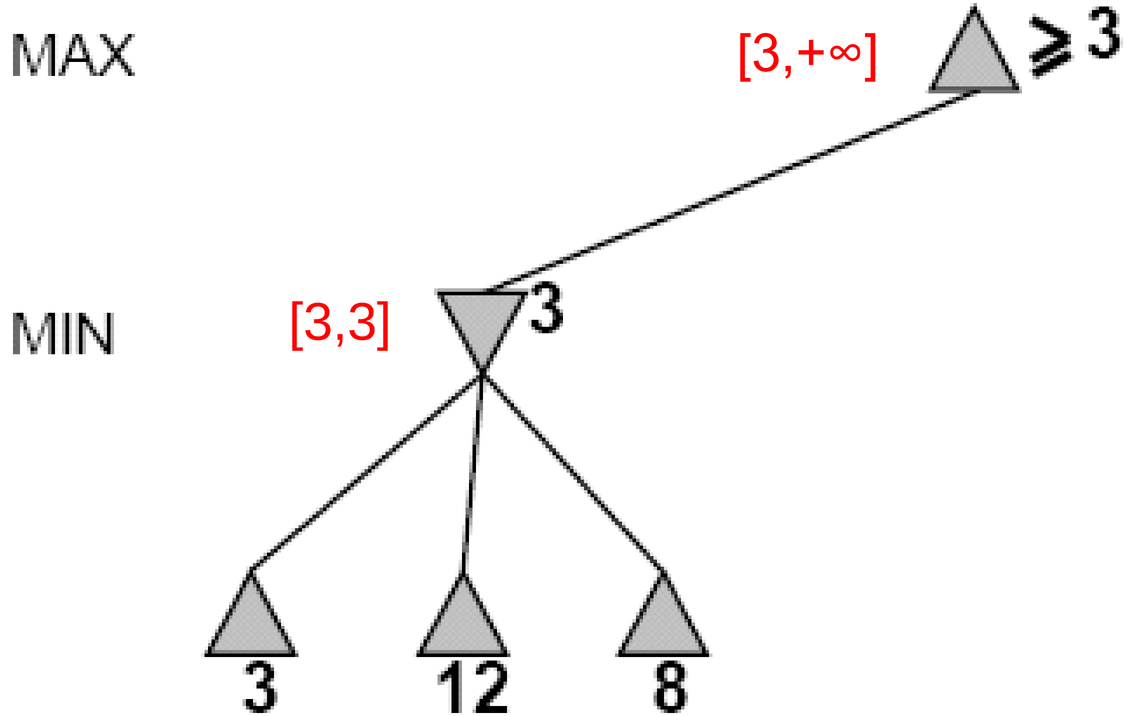
Alpha-Beta Example (continued)



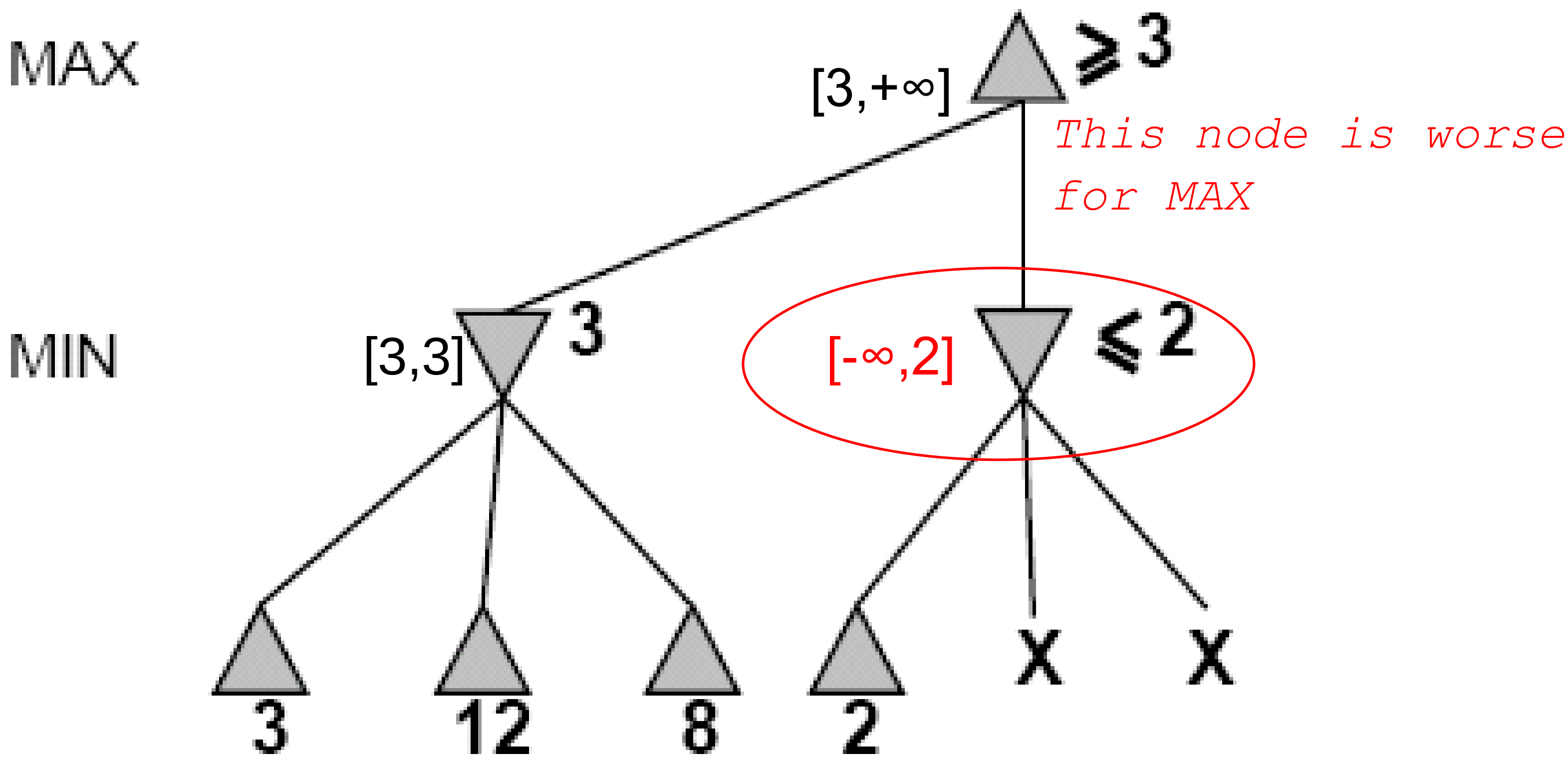
Alpha-Beta Example (continued)



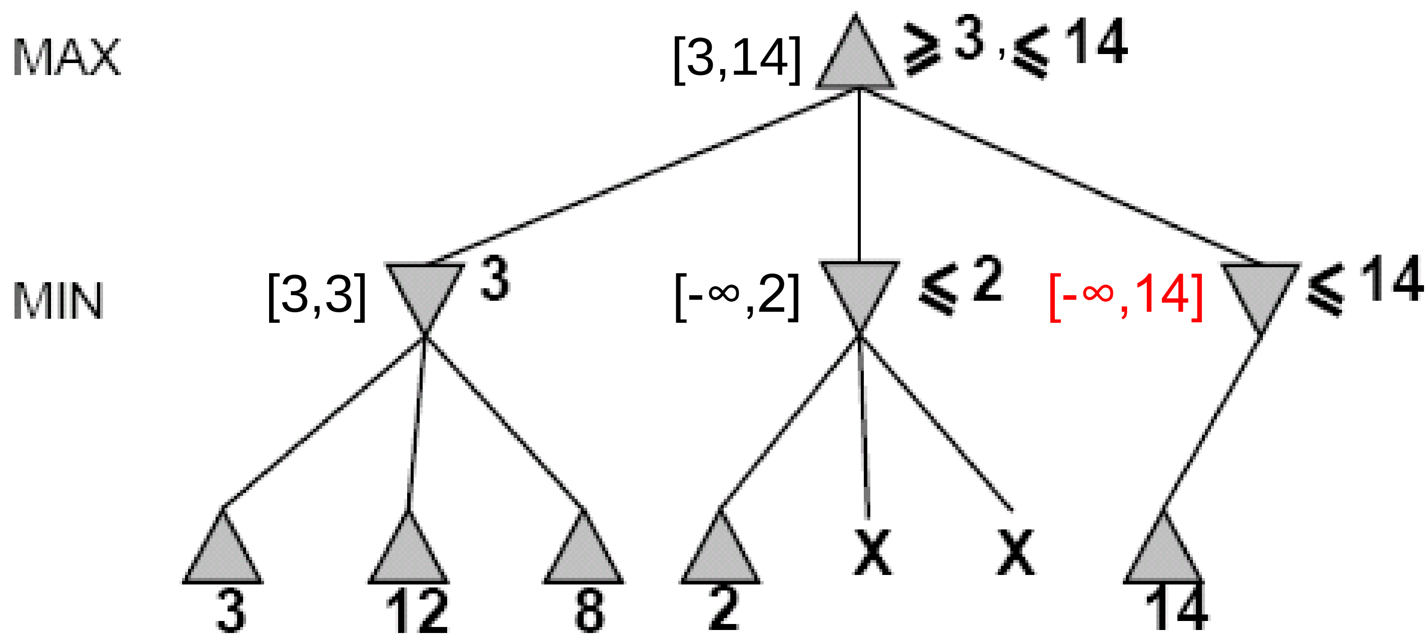
Alpha-Beta Example (continued)



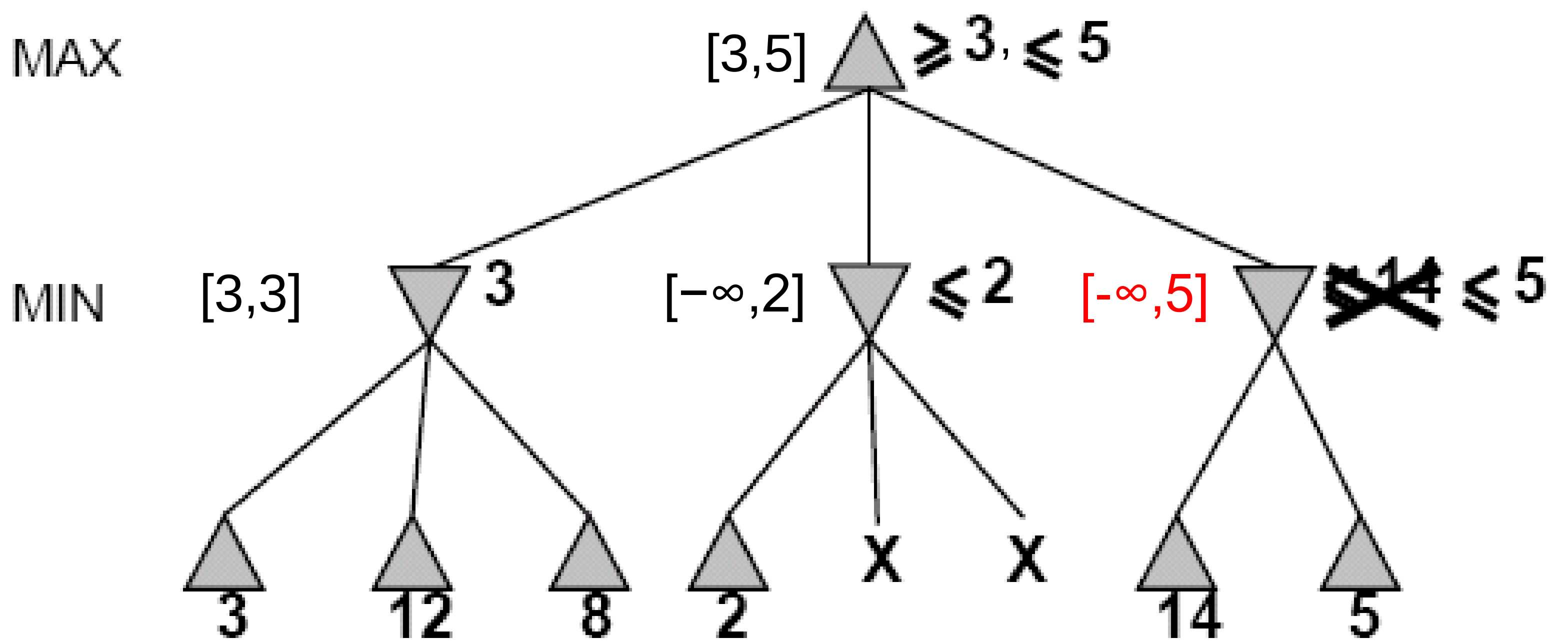
Alpha-Beta Example (continued)



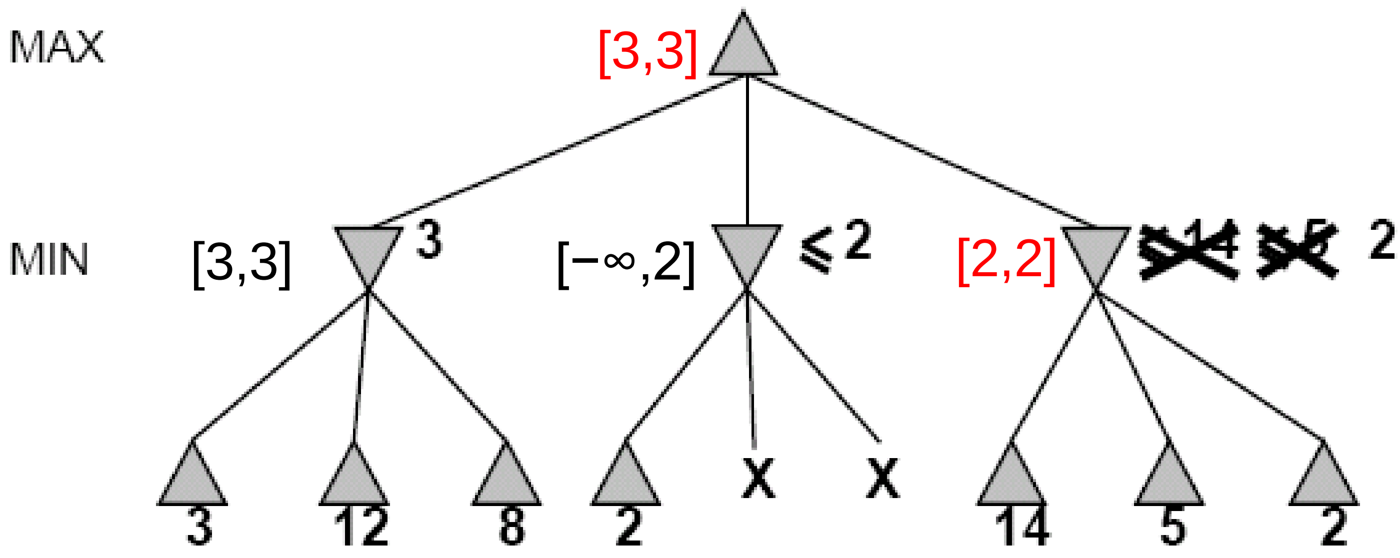
Alpha-Beta Example (continued)



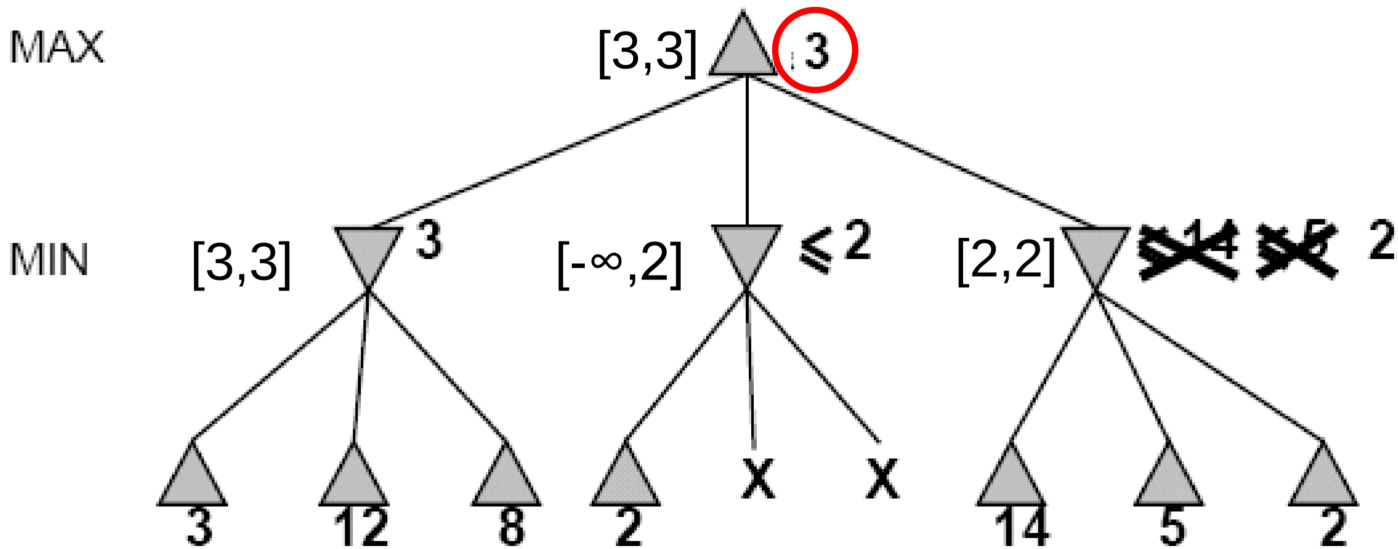
Alpha-Beta Example (continued)



Alpha-Beta Example (continued)



Alpha–Beta Example (continued)



Alpha-Beta Algorithm

function ALPHA-BETA-SEARCH(*state*) returns *an action*
  inputs: *state*, current state in game
  $\boxed{v \leftarrow \text{MAX-VALUE}(state, -\infty, +\infty)}$
  return the *action* in SUCCESSORS(*state*) with value *v*

---

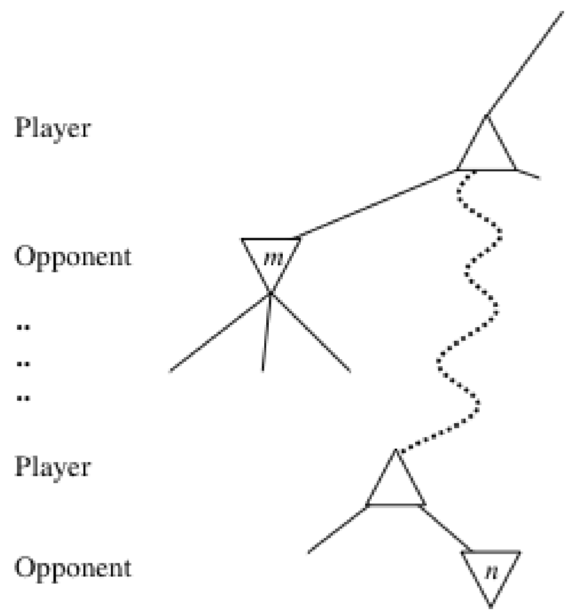
function $\boxed{\text{MAX-VALUE}(state, \alpha, \beta)}$ returns *a utility value*
  if TERMINAL-TEST(*state*) then return UTILITY(*state*)
  $v \leftarrow -\infty$
  for *a,s* in SUCCESSORS(*state*) do
    $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s, \alpha, \beta))$
    if $v \geq \beta$ then return *v*
    $\alpha \leftarrow \text{MAX}(\alpha, v)$
  return *v*

Alpha-Beta Algorithm

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

General alpha-beta pruning

- Consider a node *n* somewhere in the tree
- If player has a better choice at
  - Parent node of n
  - Or any choice point further up
- *n* will **never** be reached in actual play.
- Hence when enough is known about *n*, it can be pruned.



Final Comments about Alpha-Beta Pruning

- Pruning does not affect final results
- Entire subtrees can be pruned.
- Good move *ordering* improves effectiveness of pruning
- With "perfect ordering," time complexity is $O(b^{m/2})$
  - Branching factor of sqrt(b) !!
  - Alpha-beta pruning can look twice as far as minimax in the same amount of time
- Repeated states are again possible.
  - Store them in memory = transposition table

Games of imperfect information

- Minimax and alpha-beta pruning require too much leaf-node evaluations.
- May be impractical within a reasonable amount of time.
- SHANNON (1950):
  - Cut off search earlier (replace TERMINAL-TEST by CUTOFF-TEST)
  - Apply heuristic evaluation function EVAL (replacing utility function of alpha-beta)

Cutting off search

- Change:
  - **if** TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

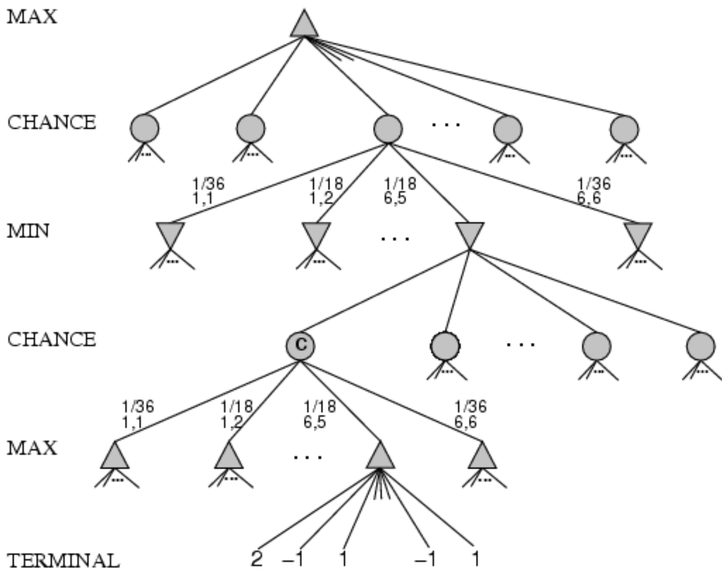
  into
  - **if** CUTOFF-TEST(*state,depth*) **then return** EVAL(*state*)

- Introduces a fixed-depth limit *depth*
  - Is selected so that the amount of time will not exceed what the rules of the game allow.
- When cuttoff occurs, the evaluation is performed.

Heuristic EVAL

- Idea: produce an estimate of the expected utility of the game from a given position.
- Performance depends on quality of EVAL.
- Requirements:
  - EVAL should order terminal-nodes in the same way as UTILITY.
  - Computation may not take too long.
  - For non-terminal states the EVAL should be strongly correlated with the actual chance of winning.
- Only useful for quiescent (no wild swings in value in near future) states

Games that include chance – The expected minimax value



Assumption: Can not calculate definite minimax value, only *expected* value.

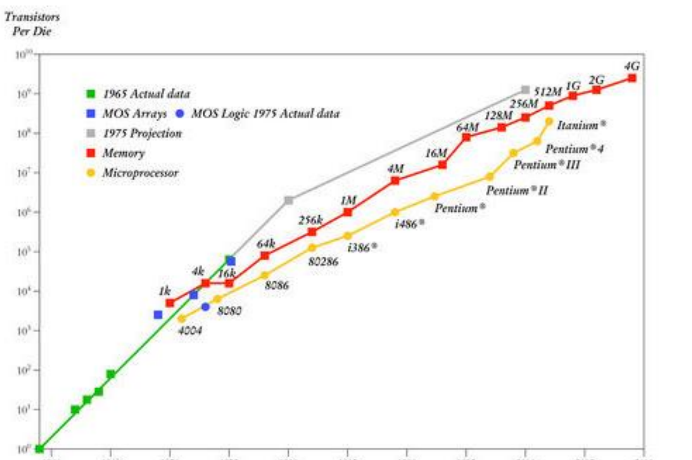
EXPECTED-MINIMAX-VALUE(n)=

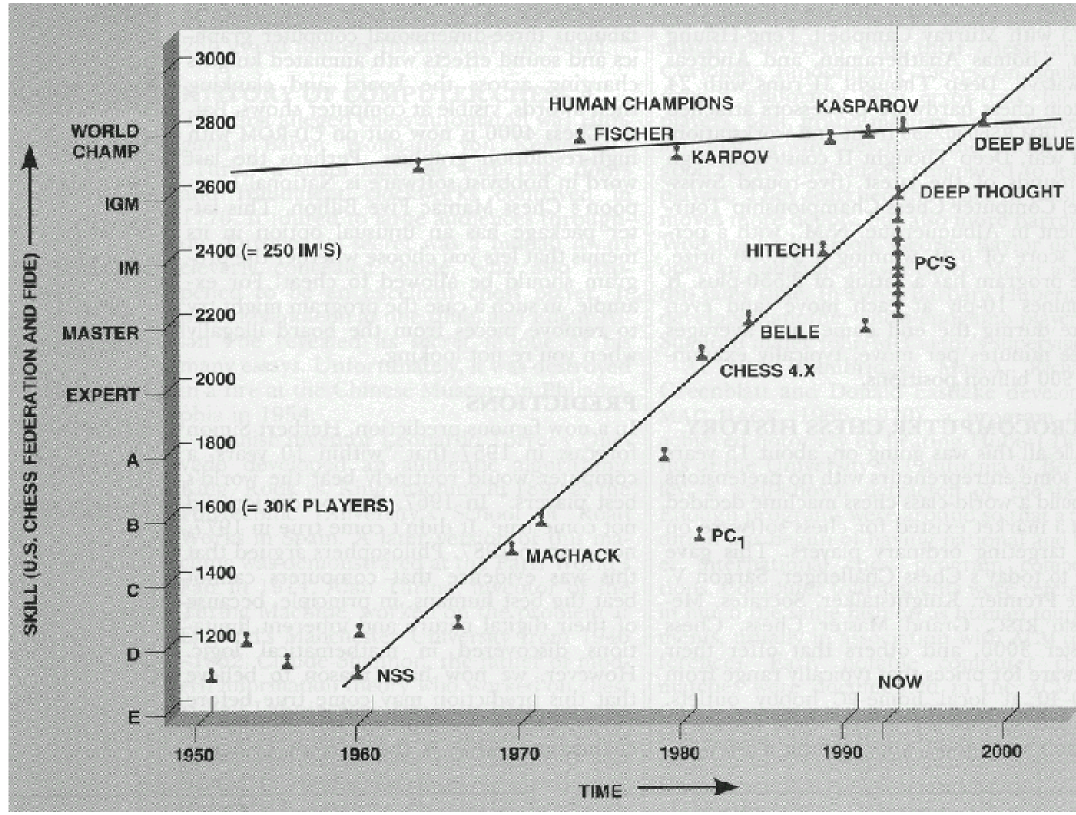
UTILITY(n)	If n is a terminal
$\max_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a max node
$\min_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a min node
$\sum_{s \in successors(n)} P(s)$. EXPECTEDMINIMAX(s)	If n is a chance node

These equations can be backed-up recursively all the way to the root of the game tree.

Lessons from chess

- Brute-force search
- Knowledge is power
- Stages of expertise
  - Quantity: #(concepts)
  - Quality: type of reasoning and learning





Chess and cognition – general(?) levels of expertise

- Reconstruction of full chess positions:
  - Chase&Simon: Perception in chess, 1973
  - Chi: Knowledge structures and memory development, 1978
  - Schneider: Chess expertise and memory for chess positions, 1993
  - ...
  - Simons: How experts recall chess positions, 2012
    - http://theinvisiblegorilla.com/blog/2012/02/15/how-experts-recall-chess-positions/
  - The Élő rating system
    - Beginner/novice, expert, master, grandmaster
    - Number of gestalt, schema, schemata, pattern, chunk,..
  - General levels of a beginner, expert, master, grandmaster?
    - Mérő: Ways of Thinking: The Limits of Rational Thought and Artificial Intelligence, 1990

Summary

- Contraint satisfaction problem,
  - as a model of „holistic” problem solving.
  - Application of search methods to solve CSP on a serial architecture.
- Search in games
  - Application of search methods
    - MINIMAX, alpha-beta cuts,
    - Decision theoretic framework in a sequential problem.

- Suggested (preparatory) task
  - „Adam, Betty, and Chris played and a window got broken.
    - Adam states: ‘Betty made, Chris is innocent.'
    - Betty states: ‘If Adam is guilty, then Chris too'.
    - Chris states: ‘I am innocent; someone else did it'."
  - Questions:
    - Is it possible that none of them lies?
    - If it is not, can we tell who lies?
    - In any case, can we infer who is guilty?