

Artificial Intelligence

Lecture 2c: Adversarial Search

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence



Games as Search Problems

- Hostile Agent (opponent) included in the world!
- Unpredictable Opponent => Solution is a Contingency Plan
- Time Limit => Unlikely to find goal! Approximation is necessary
- One of the oldest areas of AI! In 1950 Shannon and Turing created the first Chess programs!
- Chess:
 - Everyone believes that intelligence is needed to play
 - Simple rules but the game is complex
 - World fully accessible to the agent
 - Average branching factor of 35, 50-game play => 35^{100} leaves in the search tree (although there are only 10^{40} legal positions)
- Cutting concepts in the search tree and evaluation function!

Game Types

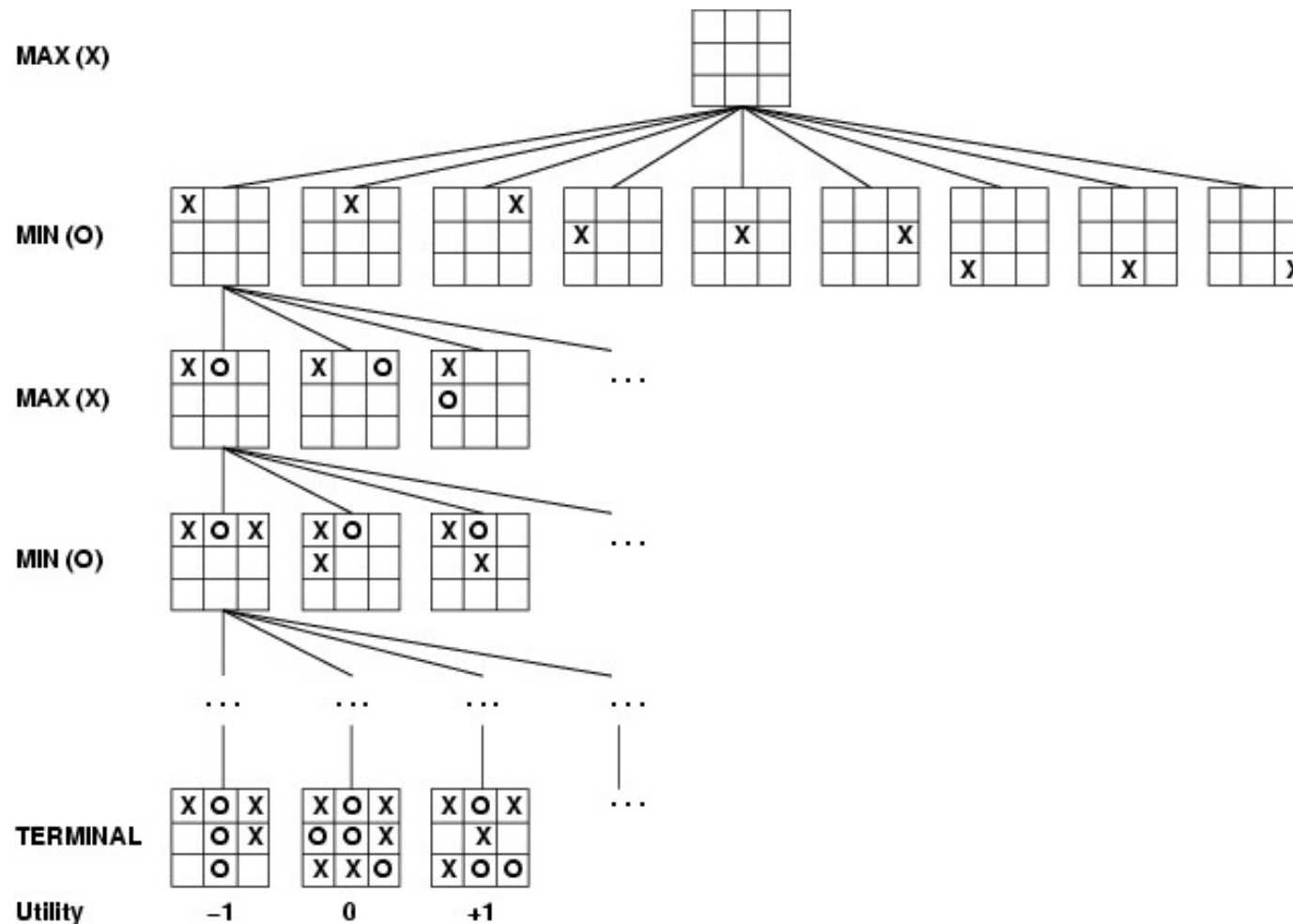
- **Information:**
 - Perfect: Chess, Checkers, Go, Othello, Backgammon, Monopoly
 - Imperfect: Poker, Scrabble, Bridge, King
- **Luck / Deterministic:**
 - Deterministic: Chess, Checkers, Go, Othello
 - Games with luck: Backgammon, Monopoly, Poker, Scrabble, Bridge, King
- **“Attack” plan:**
 - Algorithm for the perfect game
 - Finite horizon, approximate evaluation
 - Tree cuts to reduce costs

Perfect Decisions in Opponent Games - MiniMax Algorithm

- **Game: Search problem with:**
 - Initial State (board position and who is next to play)
 - Set of Operators (which define the legal movements)
 - Terminal Test (which determines whether the game is over or is in a terminal state)
 - Utility function (which gives a numerical value to the result of the game, for example 1-win, 0-draw, -1-defeat)
- **Minimax algorithm strategy:**
 - Generate the complete tree to the end states
 - Apply the utility function to these states
 - Calculate the utility values back to the root of the tree, one layer at a time
 - Choose the movement with the highest value!

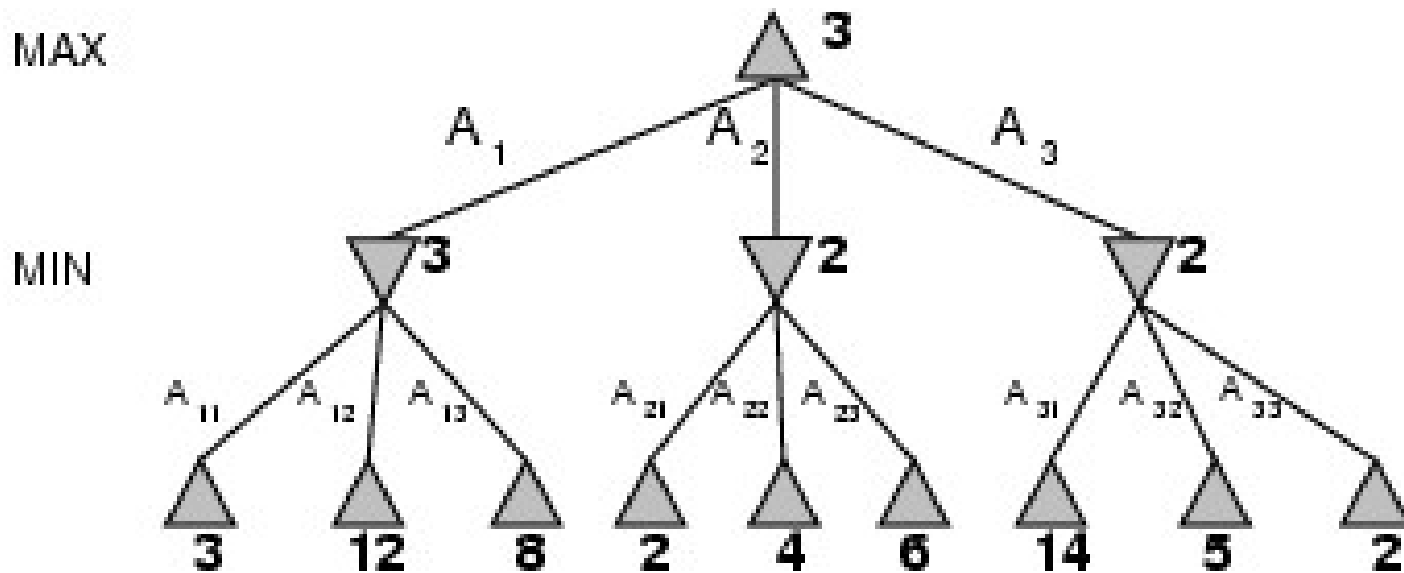
Minimax – Tic-Tac-Toe Game Example

Easy to solve the Tic-Tac-Toe Game using Minimax



Minimax – General Example

- **Strategy:**
 - Choose the movement that has the highest minimax value = the best that can be achieved against the best responses from the opponent!



Minimax Algorithm

function MINIMAX-DECISION(*state*) *returns an action*

$v \leftarrow \text{MAX-VALUE}(\textit{state})$

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*)

$v \leftarrow -\infty$

for *a, s* in SUCCESSORS(*state*) **do**

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))$

return *v*

function MIN-VALUE(*state*) *returns a utility value*

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

$v \leftarrow \infty$

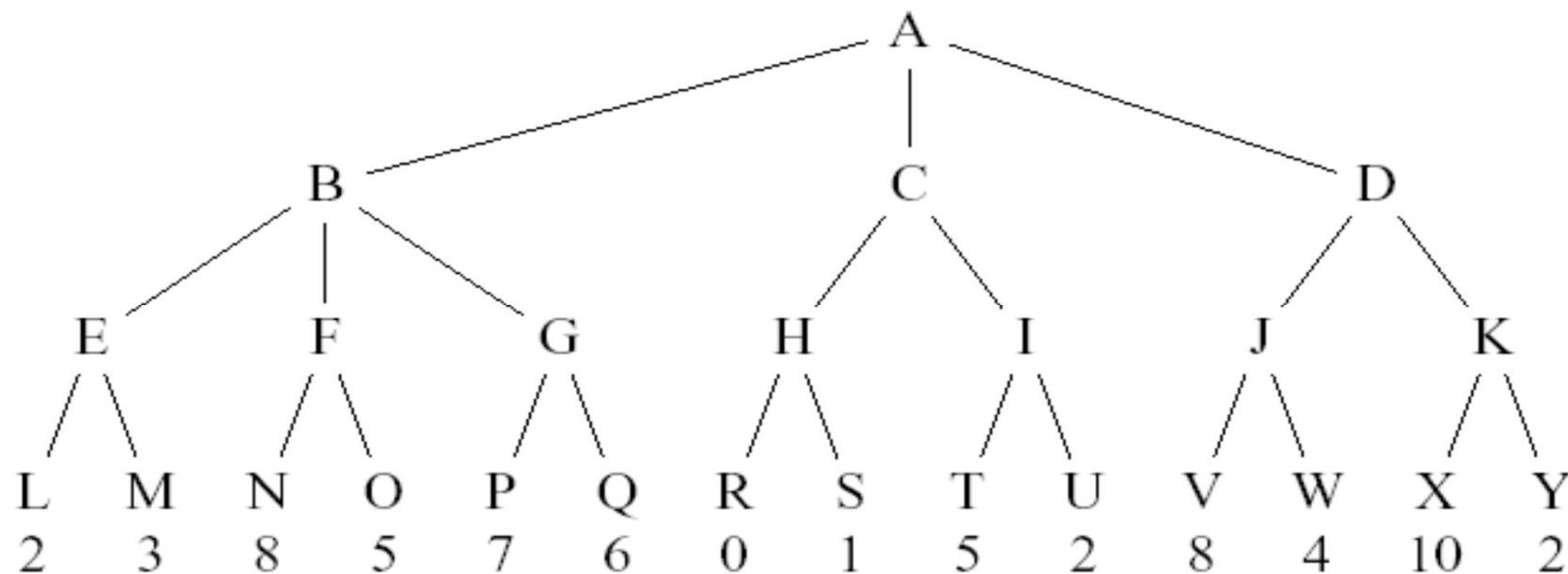
for *a, s* in SUCCESSORS(*state*) **do**

$v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))$

return *v*

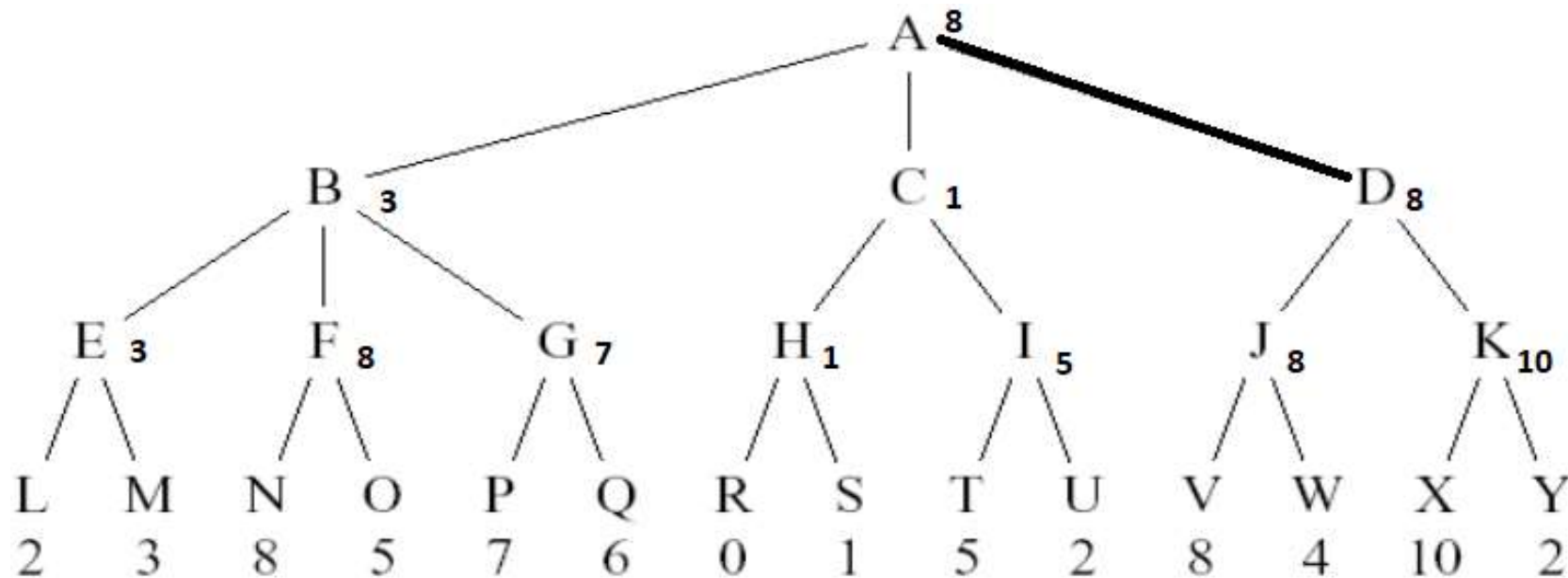
Exercise – Minimax Algorithm

- Assuming that MAX is the first to play, apply the Minimax Algorithm to the following tree, indicating the movement selected by the algorithm and the respective estimated value



Exercise – Minimax Algorithm

- Assuming that MAX is the first to play, apply the Minimax Algorithm to the following tree, indicating the movement selected by the algorithm and the respective estimated value.
- Solution: A → D (expected value = 8, e.g. oponente will play D→J and we will play J→V)



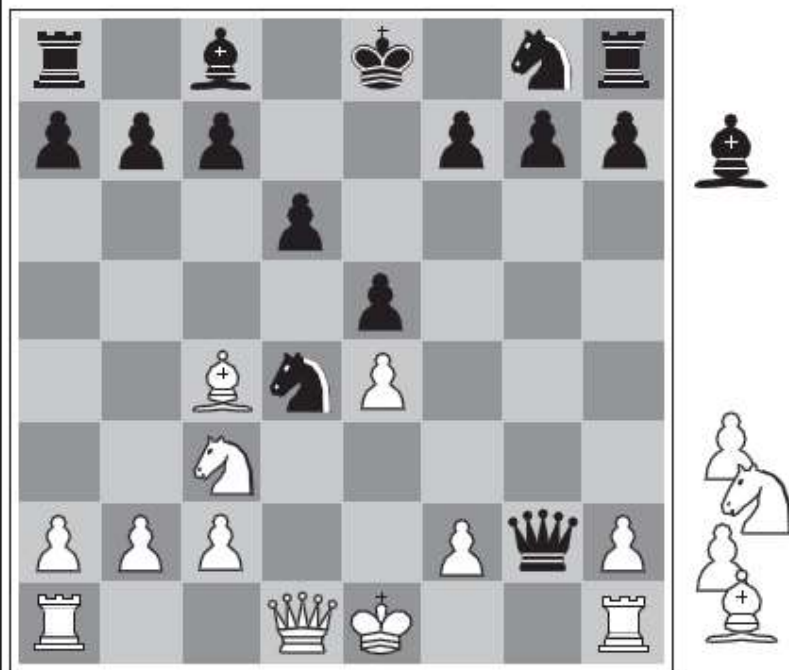
Minimax Properties

- **Properties:**
 - Complete? Yes if the tree is finite!
 - Optimal? Yes against an optimal opponent! If no?
 - Time Complexity? $O(b^m)$
 - Complexity in Space? $O(bm)$ (exploration first in depth)
- **Problem:**
 - Not viable for any minimally complex game
- **Example:**
 - For chess ($b = 35$, $m = 100$), $b^m = 35^{100} = 2.5 * 10^{154}$
 - Assuming that 450 million hypotheses are analysed per second $\Rightarrow 2 * 10^{138}$ years to arrive at the solution!

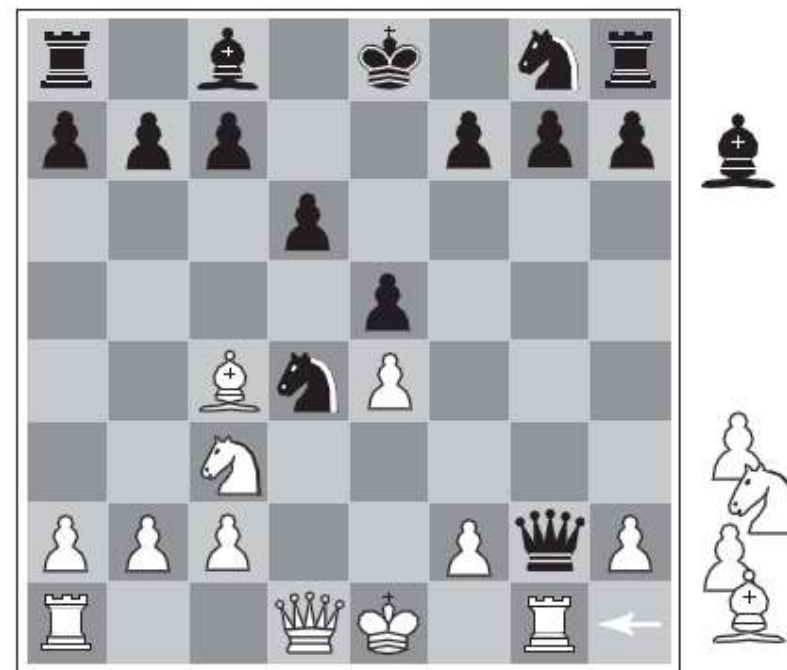
Limited Resources

- Assuming we have 100 seconds and explore 10^4 nodes / second, we can explore 10^6 knots per movement
- Usual approach:
 - Cutting Test: Depth Limit
 - Evaluation Function: Estimated utility (interest) for the position/state
- Example (Chess):
 - Cutting Test: Depth of Analysis n
 - Simple evaluation function = sum of the values of the white pieces in play minus the sum of the values of the black pieces in play!
 - Evaluation function should only be applied to stable positions (in terms of their value). For example, positions with possible captures should be further explored ...
- Another Problem: Horizon problem!

Chess – Evaluation Functions



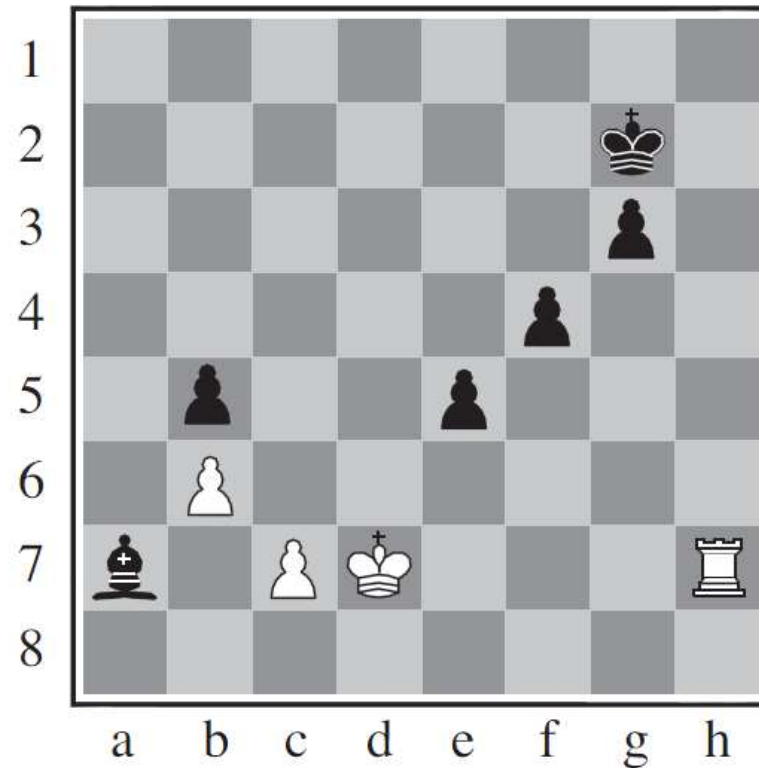
(a) White to move



(b) White to move

Chess – Horizon Problem

- With a limited horizon it seems interesting for Black to check with the pawns and delay the capture of the bishop (beyond the horizon)

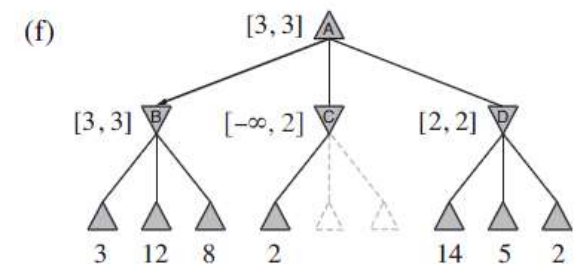
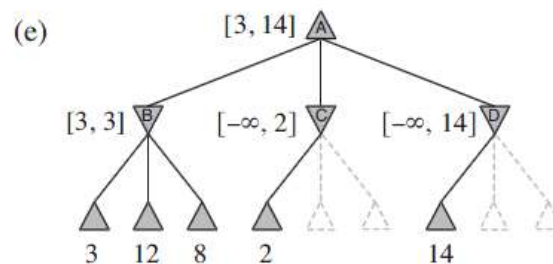
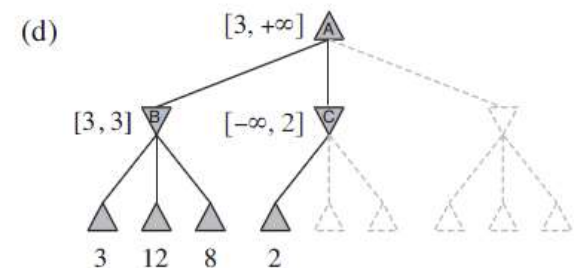
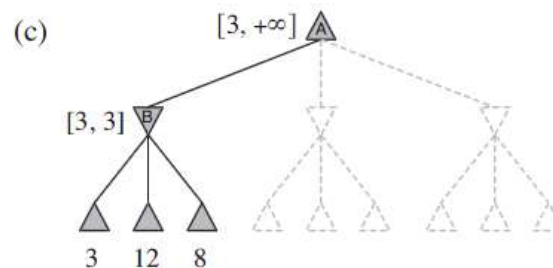
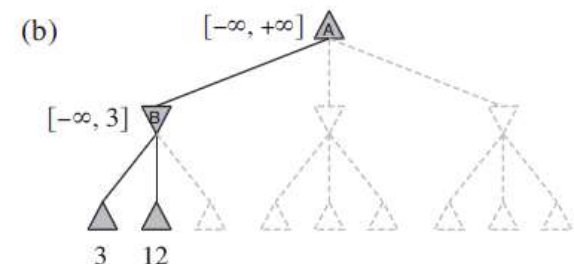
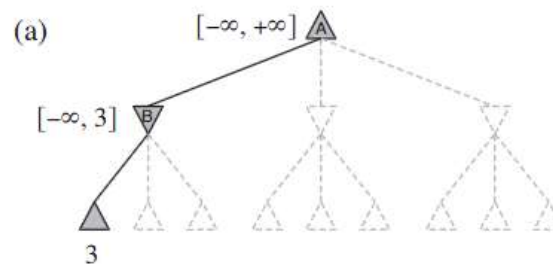


Cuts - *MinimaxCutoff*

- ***MinimaxCutoff* is identical to *MinimaxValue* except:**
 - Terminal-Test is replaced by *Cutoff*
 - Utility is replaced by Evaluation (which calculates an evaluation of the reached position)
- **Does it work in practice?**
 - If $b^m = 10^6$ with $b = 35 \Rightarrow m = 4$
- **Playing chess with depth 4 is absolutely miserable!**
 - Depth 4 \Rightarrow Newbie Player
 - Depth 8 \Rightarrow PC, Good Human Player
 - Depth 12 \Rightarrow Deep Blue, Kasparov

Alpha-Beta Cuts (1)

- α is the best value (for Max) found so far on the current path
- If V is worse than α , Max should avoid it \Rightarrow cut off the branch
- β is defined in the same way for Min

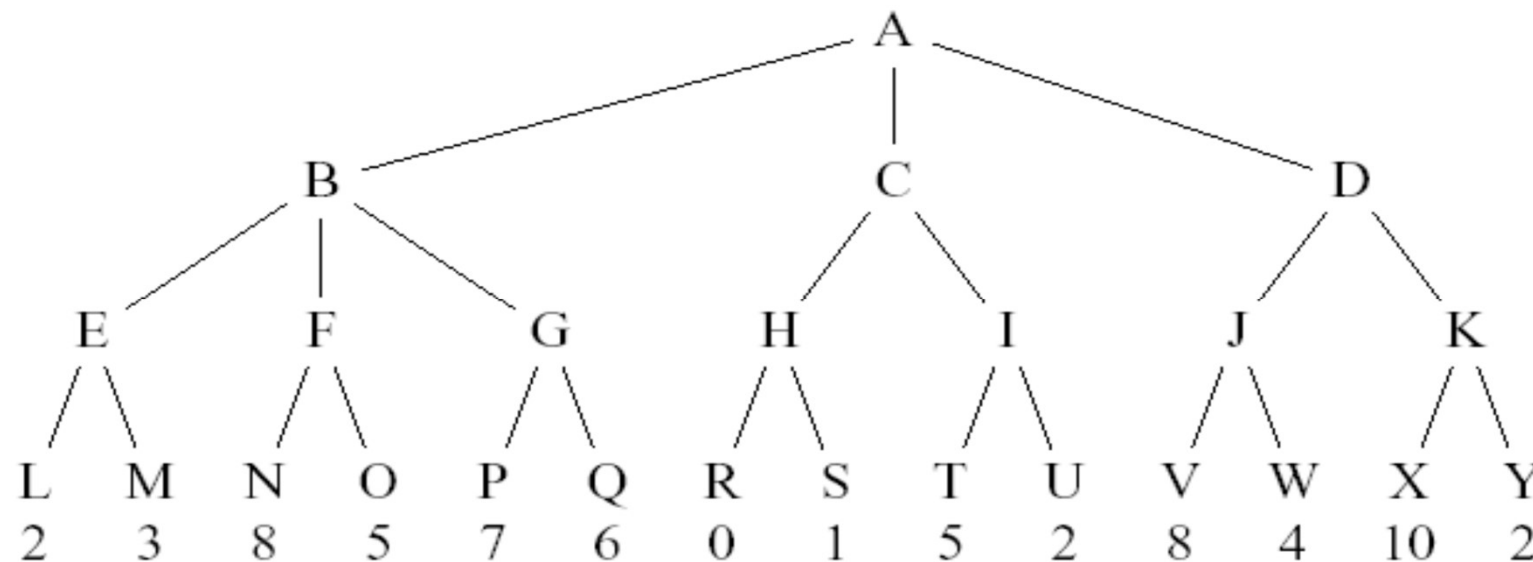


Alpha-Beta Cuts (2)

- **Alpha-Beta cuts do not affect the final result**
- **Good ordering improves cutting efficiency**
 - Essential to reason about node ordering
- **Perfectly arranged: Time Complexity = $O(b^{m/2})$**
 - Doubles search depth
 - Depth 8 => Good chess player
- **Good example of the value of reasoning about which computations are relevant:**
 - This is essential in Intelligent Systems

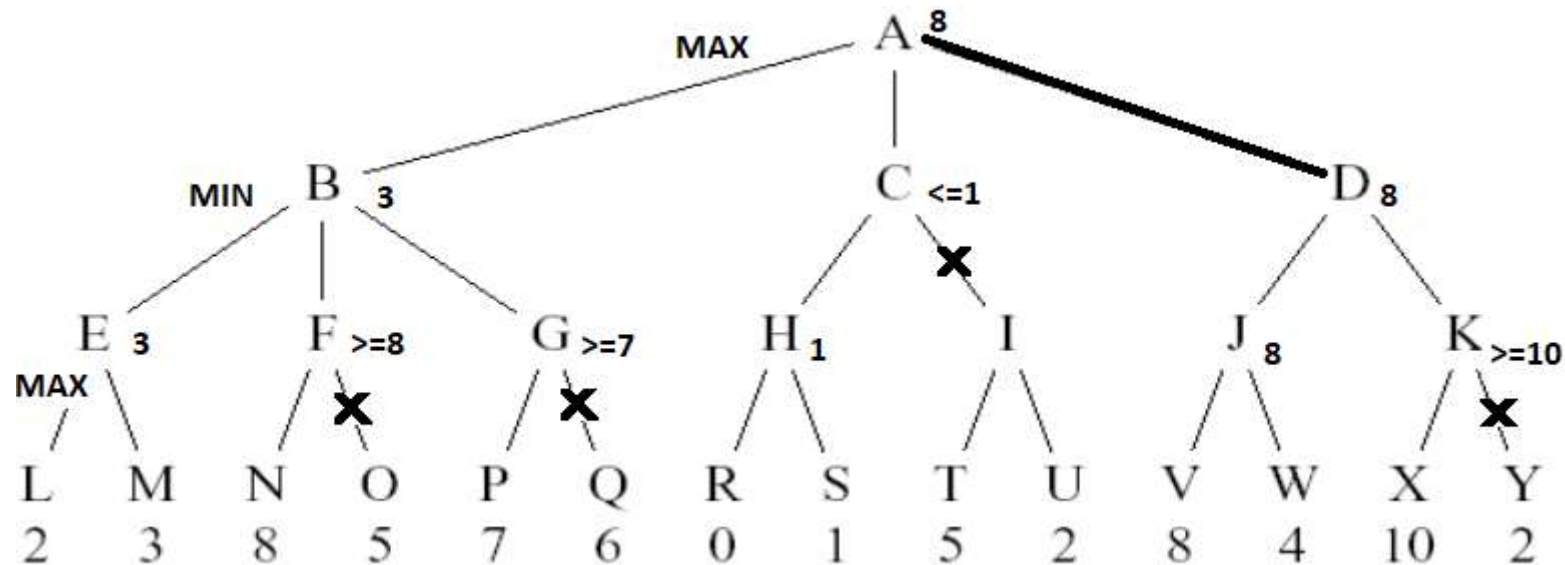
Exercise – MINIMAX with Cuts

- Assuming that MAX is the first to play, apply the Minimax Algorithm with Alpha-Beta cuts to the following tree, indicating the movement selected by the algorithm. Indicate graphically and justify all the cuts made when applying the Minimax algorithm



Exercise – MINIMAX with Cuts

- Assuming that MAX is the first to play, apply the Minimax Algorithm with Alpha-Beta cuts to the following tree, indicating the movement selected by the algorithm. Indicate graphically and justify all the cuts made when applying the Minimax algorithm
- Solution:



Exercise – MINIMAX with Cuts

- Assuming that MAX is the first to play, apply the Minimax Algorithm with Alpha-Beta cuts to a tree with three levels, a branch factor 3 and with the following values of the evaluation function for the final line:

[8 6 3 1 10 3 15 20 6 7 4 6 25 10 4 4 3 5 1 20 4 12 1 10 4 22 10]

- Indicate graphically and justify all the cuts made when applying the Minimax algorithm

Deterministic Games

- **Checkers:**
 - **Chinook ended the 40-year reign of human champion Marion Tinsley in 1994.** He used a database for match ends defining the perfect way to win for all positions involving 8 or less pieces (in total 443748401247 positions). Nowadays checkers is a solved problem.
- **Chess:**
 - **Deep Blue defeated human world champion Gary Kasparov** in a 6-game game **in 1997**. Deep Blue searched 200 million positions per second and used an extremely sophisticated evaluation function and (undisclosed) methods to extend some lines of search beyond depth 40!
- **Othello:**
 - **Human champions refuse to compete with computers** because they have no chance! (b between 5 and 15)

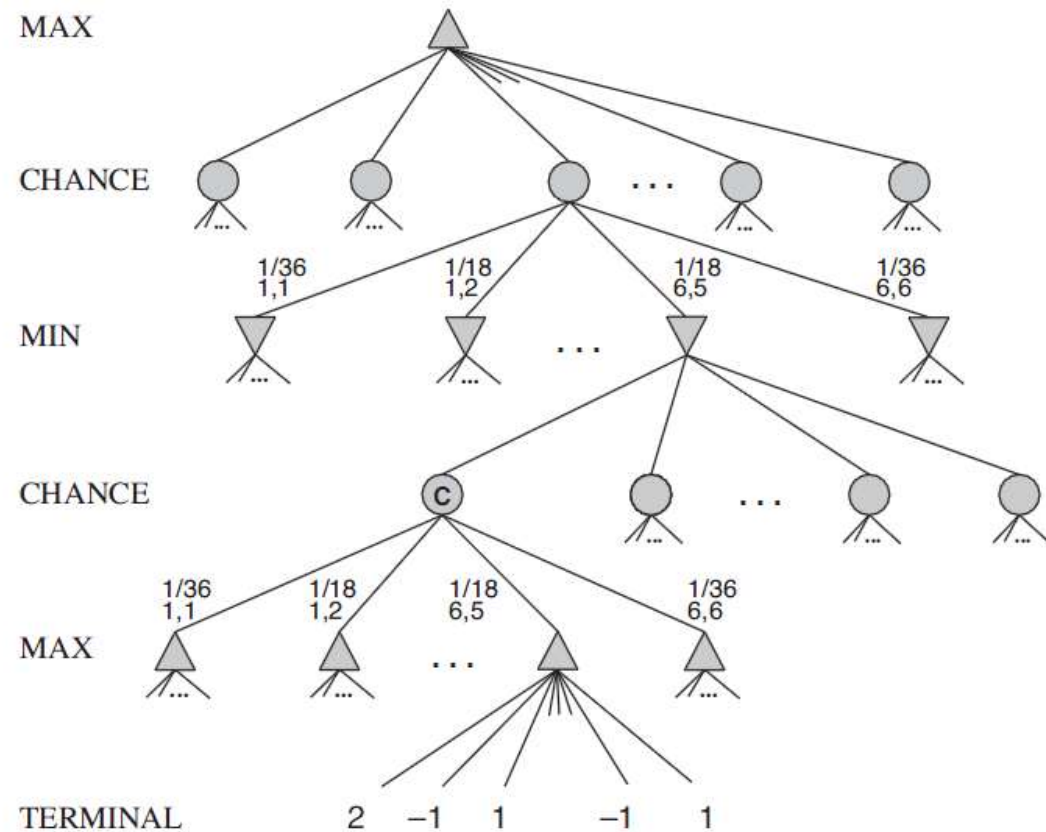
Deterministic Games - Go

- **Go (2015):**
 - Human champions refuse to compete with computers because machines cannot play reasonably ($b > 300$)
- **Go (2017-2020):**
 - AlphaGo (Fan, Lee), AlphaGo Master, AlphaGo Zero and AlphaZero! Machines beat 100-0 human champions and previous machines.



Games of Luck - *ExpectiMiniMax*

- In many games, unlike chess, there are external events that affect the game, such as drawing a card or rolling a dice!
- Examples: Card games, Backgammon, Scrabble, ...
- Search tree must include probability nodes!
- Decision is made based on the expected value!
- *ExpectiMiniMax*



Exercises – Adversarial Search/Games

Formulate some of the following 2-player board games as a adversarial search problem (game) and design an intelligent agent capable of playing it, using Minimax with alpha-beta cuts:

- | | | |
|---|---|--|
| 1. Chess | 19. Link Five | 35. Paper & Pencil Racing |
| 2. Shogi | 20. Mancala (normal/4x8) | 36. Arimaa |
| 3. Checkers | 21. Fanorana | 37. Gipf |
| 4. Connect 4 | 22. Nine Mens Morris | 38. Lines of Action |
| 5. Attaxx | 23. Alquerque | 39. Connections |
| 6. Chinese Checkers | 24. Tablut | 40. Fanorama |
| 7. Othello (Reversi) | 25. Surakarta | 41. Hexxagon |
| 8. Abalone | 26. Terrace | 42. Jungle Game |
| 9. Hex | 27. Go | 43. Seega |
| 10. Omega Chess | 28. Dots and Boxes | 44. Halma (2 players) |
| 11. Tori Shogi | 29. Dots and Hexagons | 45. Quits (Gigamic) |
| 12. Quoridor (Normal/Kids)
(Gigamic) | 30. Amazons | 46. Pylos (Gigamic) |
| 13. Quarto (Gigamic) | 31. Scrabble (visible letters and
known draws) | 47. Tantrix (2 players) |
| 14. Quixo (Gigamic) | 32. Tic tac toe (normal/ memory/
movement, 3D) | 48. Carcassone (2 players, known
draws) |
| 15. Quads (Gigamic) | 33. Dominoes (visible pieces and
known draws) | 49. Blokus |
| 16. Sahara (Gigamic) | 34. Backgammon (known draws) | 50. Sputnik (Gigamic) |
| 17. Pentaminoes (8x8) | | 51. Katamino (Gigamic) |
| 18. Stratego | | 52. Gobblet (Gigamic) |

Summary - Games

- **Working with games is extremely interesting (but also dangerous...)**
 - Easy to test new ideas!
 - Easy to compare agents with other agents
 - Easy to compare agents with humans!
- **Games illustrate several interesting points of AI:**
 - Perfection is unattainable => it is necessary to approximate!
 - It is a good idea to think about what to think!
 - Uncertainty restricts the allocation of values to states!
- **Games work for AI like Formula 1 for car building!**

Artificial Intelligence

Lecture 2c: Adversarial Search

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence

