CS480 – Introduction to Artificial Intelligence

Topic: Game Playing
Chapter: 5

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

https://twitter.com/bilgicm

Adversarial Search

- A **multiagent** and **competitive** environment
- Agents are trying to maximize their utilities at the expense of other agents' utilities
- Zero-sum games, where the sum of the utilities in the end is constant
  - E.g., win = +1, loss = -1, tie = 0
- Games with abstract descriptions attracted the attention of AI researchers quite a bit
  - The states are relatively easy to represent, limited number of actions, precise rules, expected outcomes, etc. E.g., chess, checkers, backgammon
  - Physical games, except soccer, has not attracted much attention

Chapter 5 v.s. Chapter 3

- Chapter 3:
  - Goal-based agent, where the path to the goal is optimized
- Chapter 5:
  - Utility-based agent, where the expected utility in the end is optimized

Games are pretty hard to solve

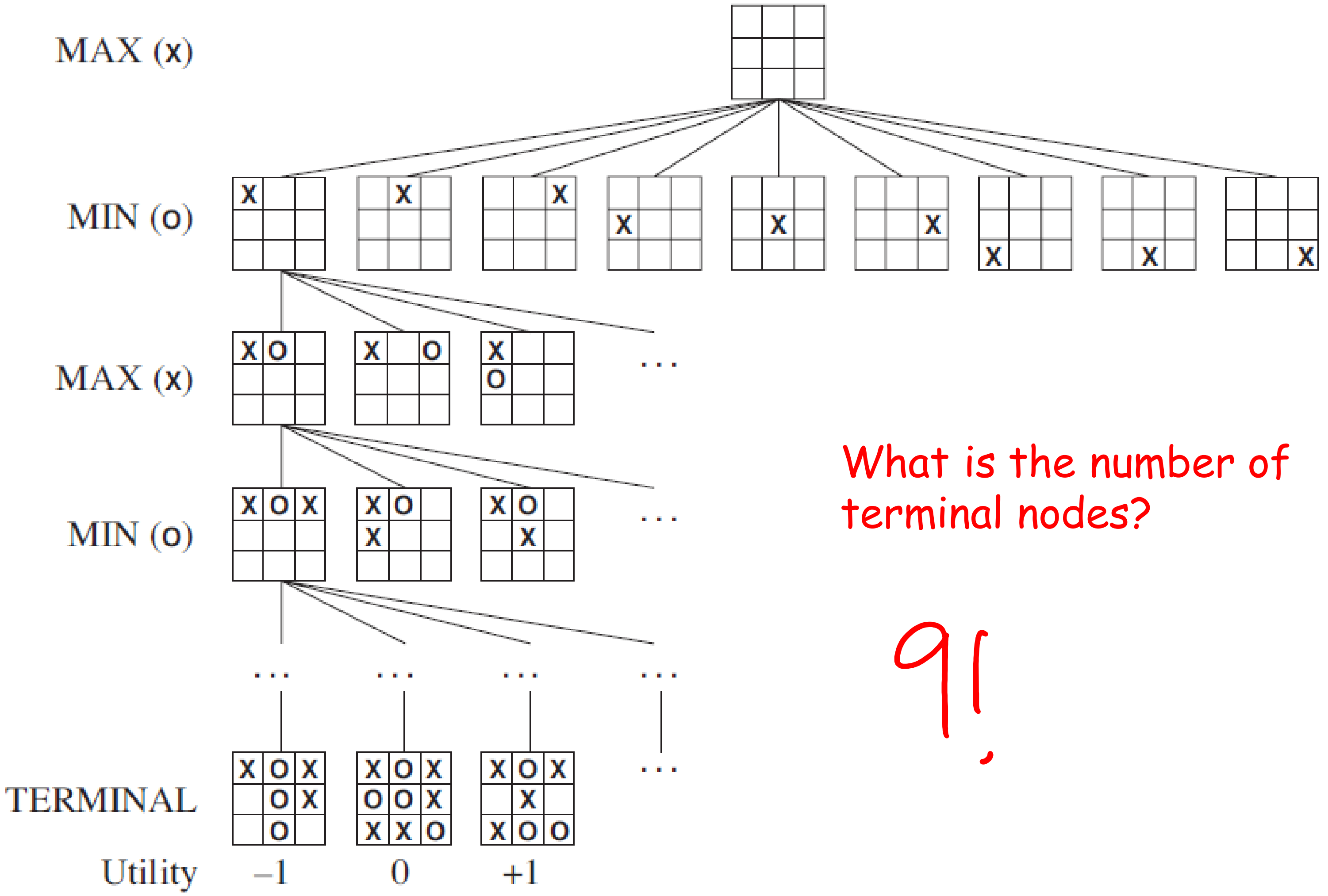
- For example, chess
  - Average branching factor: 35
  - Can go about 50 moves per player
    - Search tree depth is 100
    - Number of nodes: $35^{100} \approx 10^{154}$
    - Number of estimated atoms in the universe: 10^{80}
  - You still have to make a decision before you can calculate the optimal move

Game Definition

- S_0: The initial state
- *Player*(*s*): Which player's turn in state *s*
- *Action(s)*: The set of legal moves in state *s*
- *Result(s, a)*: The transition model
- *Terminal-test*(*s*): Is the game over
- *Utility(s, p)*: The utility of player *p* in state *s*
- Two players: MAX and MIN. MAX moves first

MAX : X   MIN : O

Game Tree – Tic-Tac-Toe



What is the number of terminal nodes?

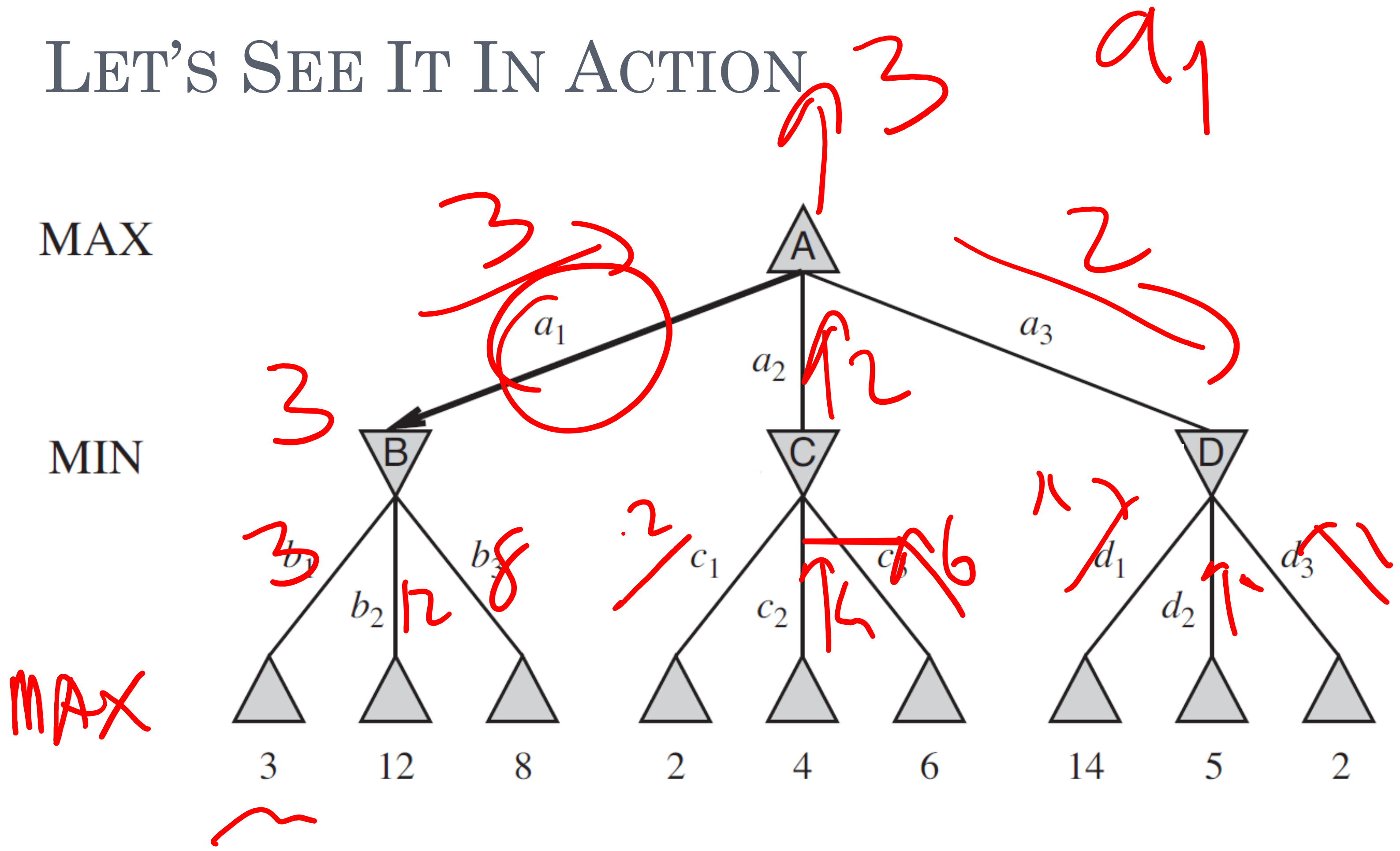
9!

$9 \times 8 \times 7 \times 6 \ldots 1$

Optimal Strategy

- Maximize utility
- An optimal strategy leads to outcomes at least as good as any other strategy when one is playing on *infallible* opponent
  - What if the opponent is not playing optimally?
- The **minimax** algorithm (Figure 5.3)

Let's See It In Action



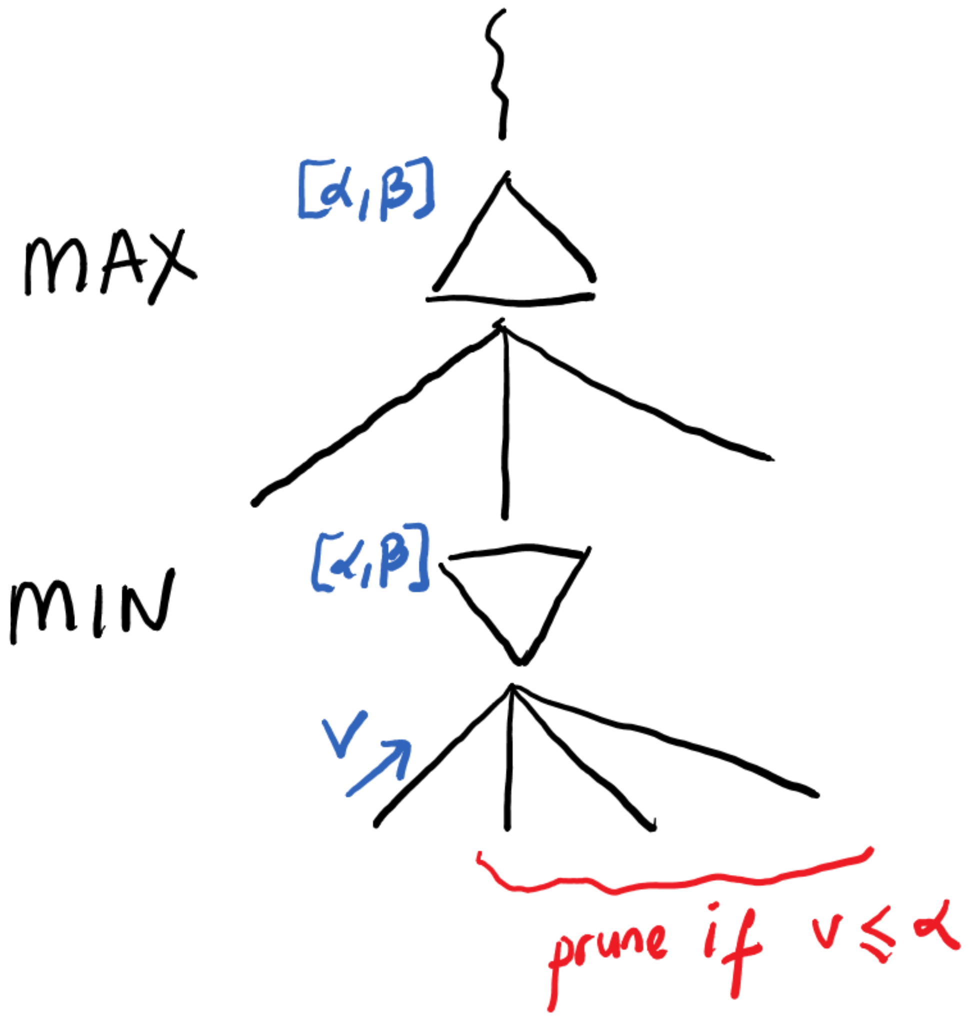
Time and Space Complexity

- The minimax algorithm is a depth-first search algorithm
  - Space: $O(bm)$
  - Time: $O(b^m)$
- Can we do better?

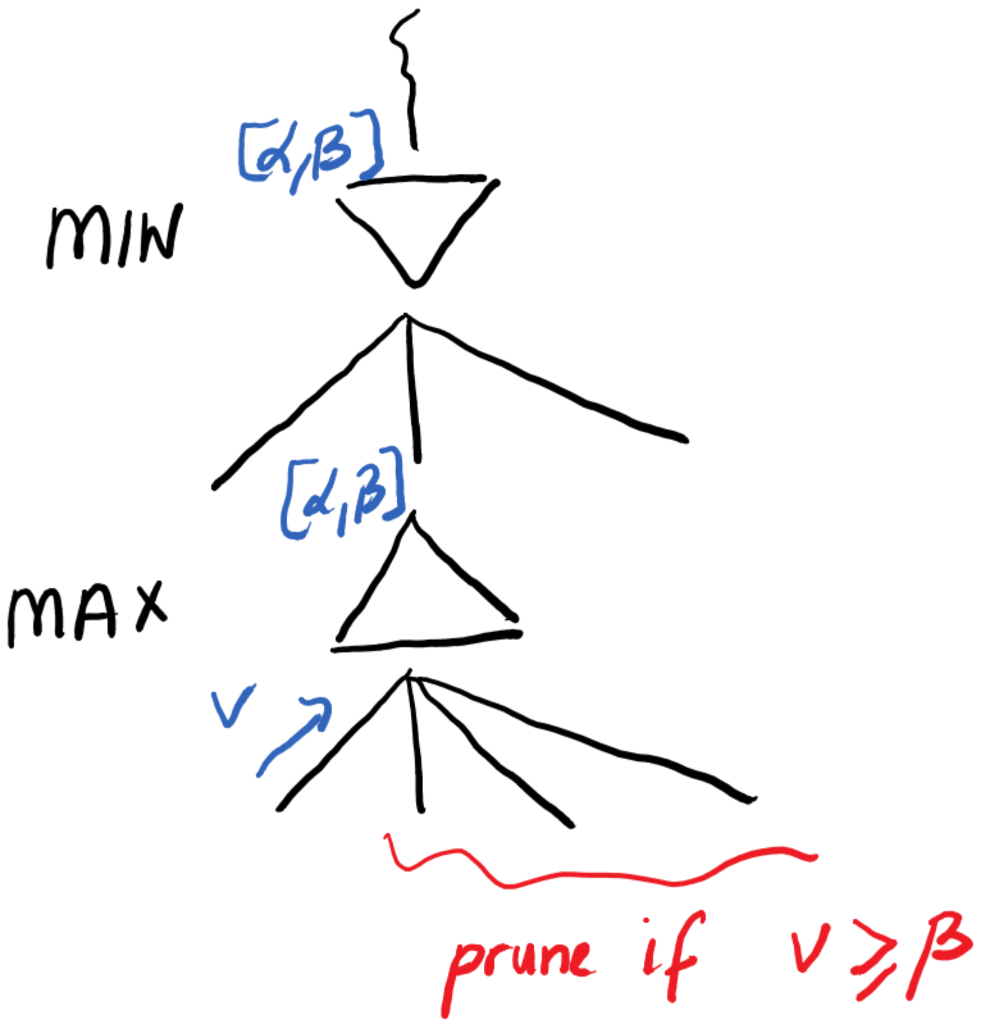
ALPHA-BETA PRUNING

- Keep lower and upper bounds (α, β)
- MAX updates the lower bound α
- MIN updates the upper bound β
- Both pass the current bounds to their children
- MIN(α, β) prunes if $v \leq \alpha$; MAX already has a better choice α
- MAX(α, β) prunes if $v \geq \beta$; MIN already has a better choice β

Example: MIN prunes



Example: MAX prunes



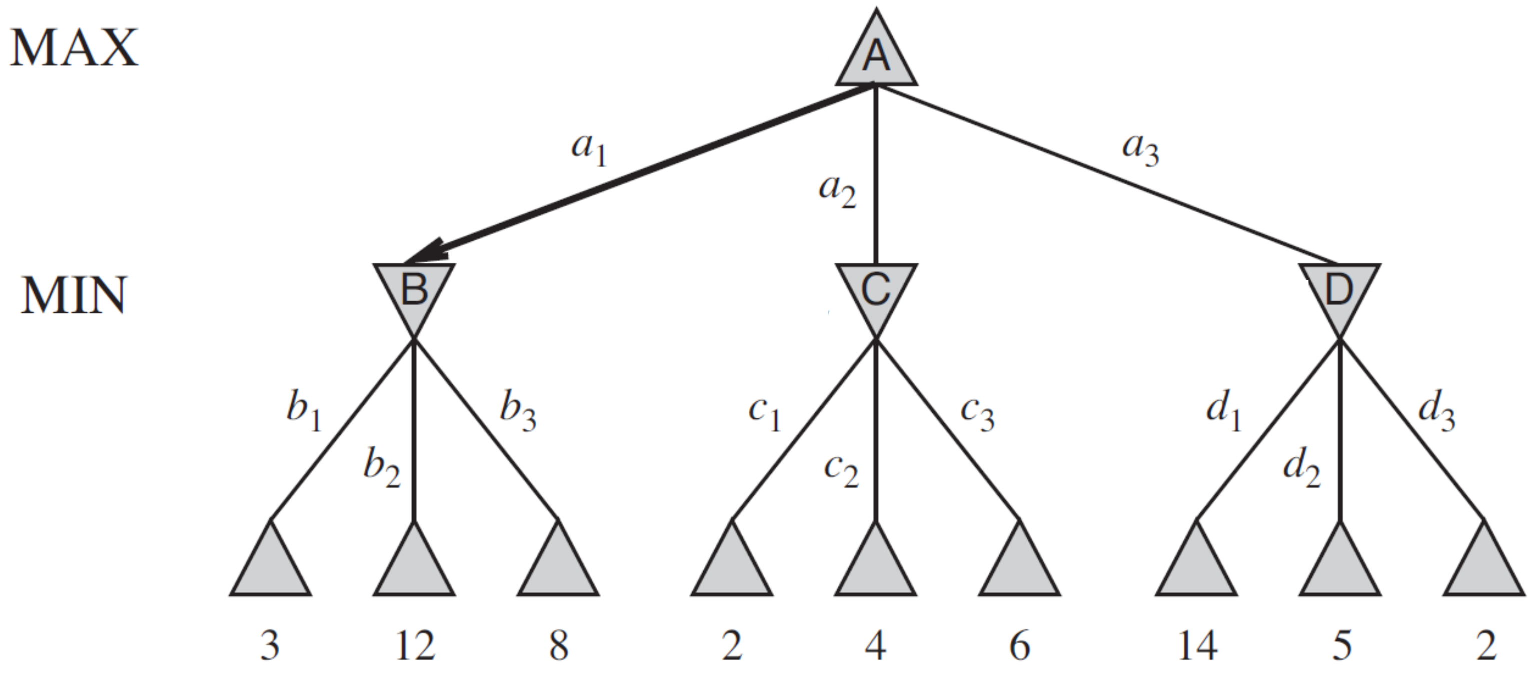
ALPHA-BETA PRUNING ALGORITHM

```
function ALPHA-BETA-SEARCH(state) returns an action
    $v \leftarrow$ MAX-VALUE(state, $-\infty$, $+\infty$)
    return the action in ACTIONS(state) with value $v$
```

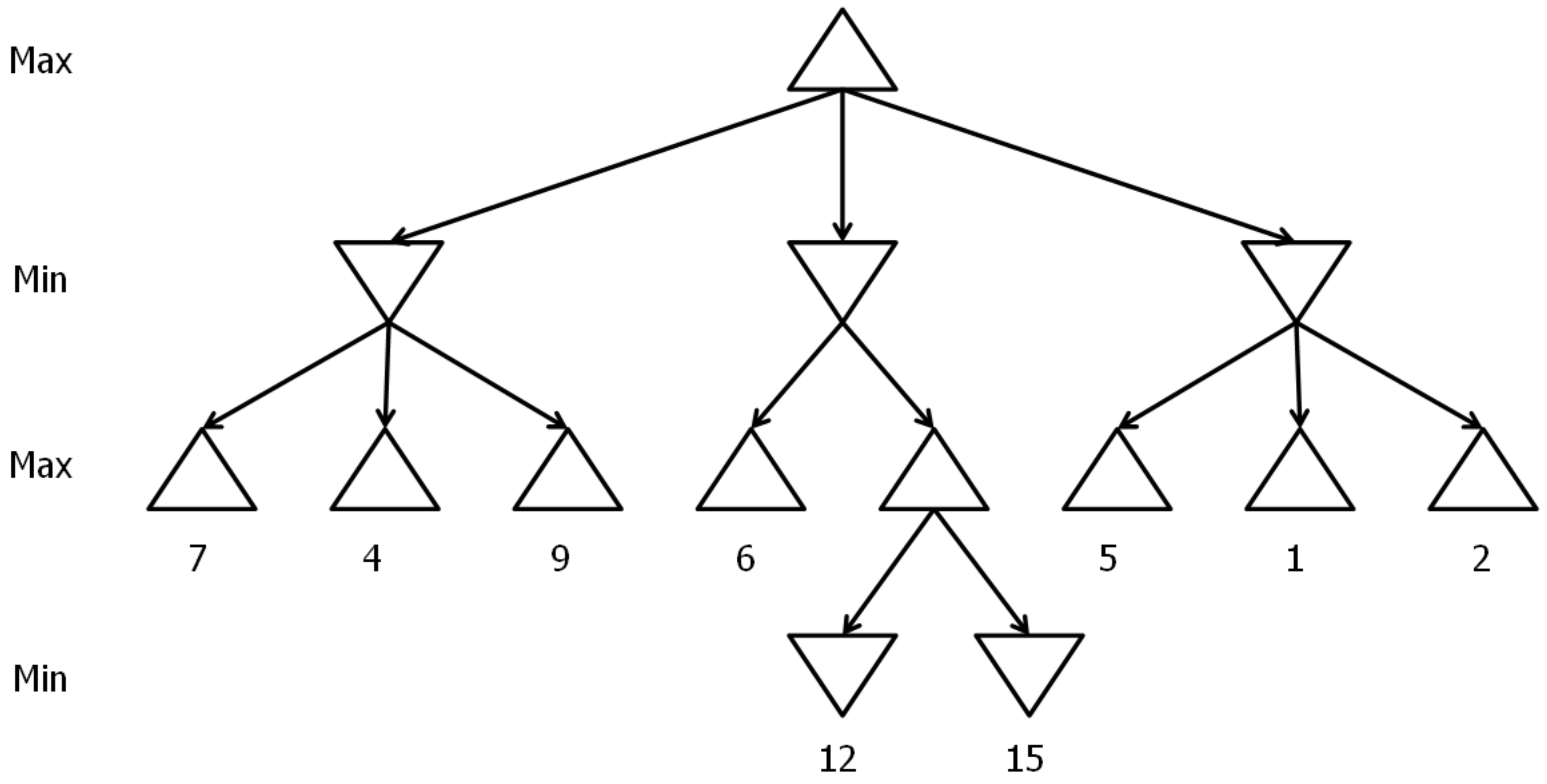
```
function MAX-VALUE(state, $\alpha$, $\beta$) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow -\infty$
    for each a in ACTIONS(state) do
        $v \leftarrow$ MAX($v$, MIN-VALUE(RESULT(s,a), $\alpha$, $\beta$))
        if $v \geq \beta$ then return $v$
        $\alpha \leftarrow$ MAX($\alpha$, $v$)
    return $v$
```

```
function MIN-VALUE(state, $\alpha$, $\beta$) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow +\infty$
    for each a in ACTIONS(state) do
        $v \leftarrow$ MIN($v$, MAX-VALUE(RESULT(s,a) , $\alpha$, $\beta$))
        if $v \leq \alpha$ then return $v$
        $\beta \leftarrow$ MIN($\beta$, $v$)
    return $v$
```

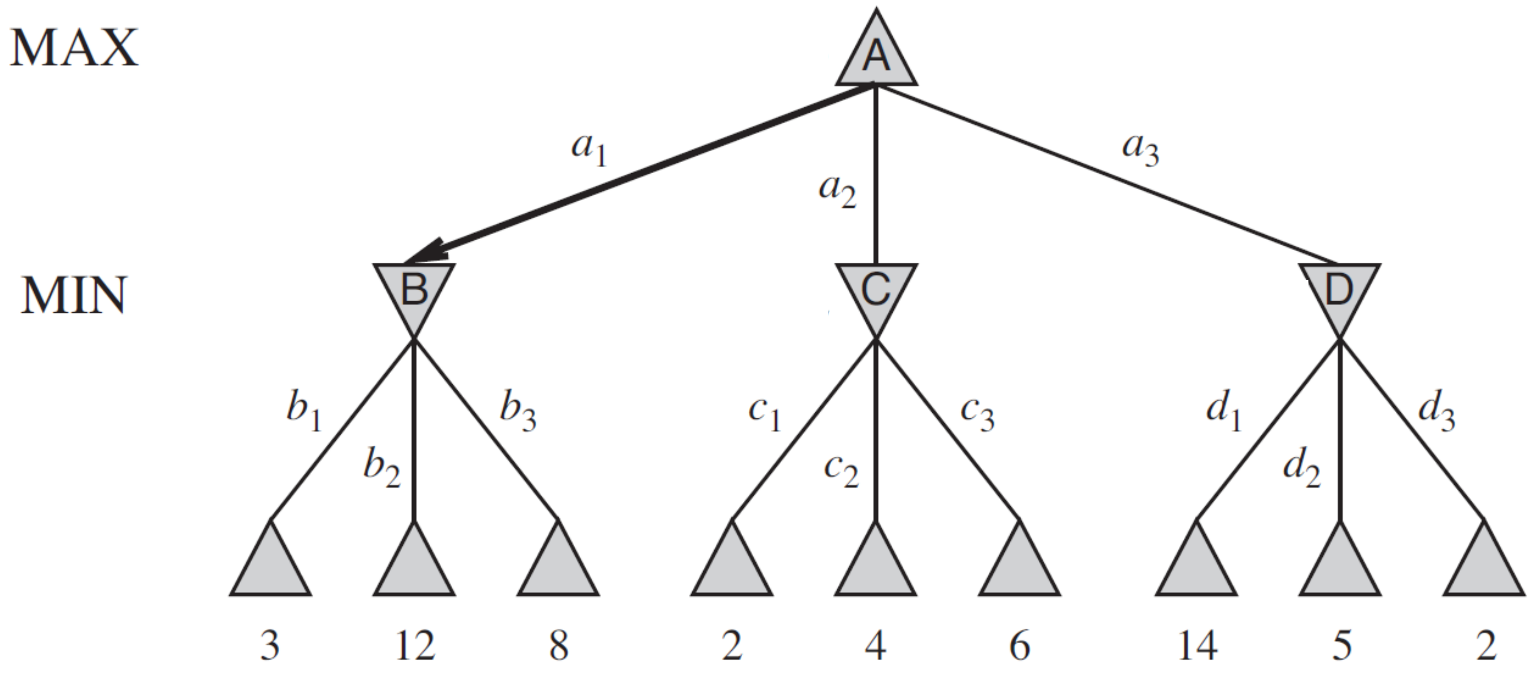
Alpha-Beta Pruning (Figure 5.7)



Alpha-Beta Pruning – Another Example

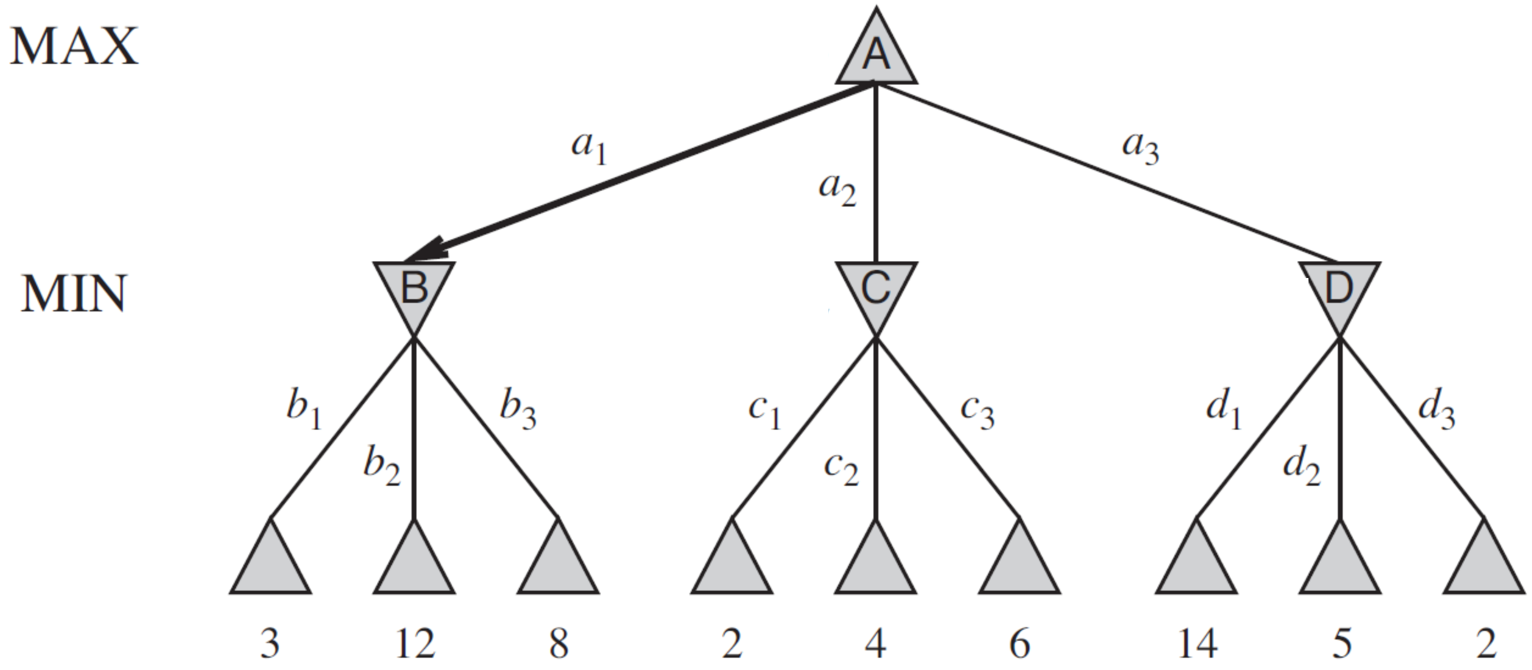


Alpha-Beta Pruning (Figure 5.7)



Can we do better?

Move Ordering



What if we re-order these?

Still

- To be able to prune, the alpha-beta algorithm has to visit some of the leaf nodes
  - Of course, this is not practical for most games
- A solution: early cut-off
  - How can we do this? Remember, the utility function is defined for the terminal states.

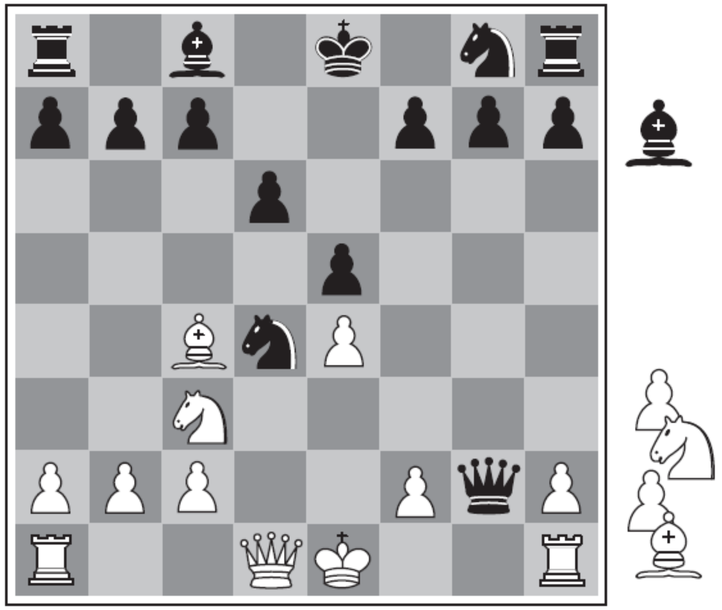
Heuristic Evaluation Functions

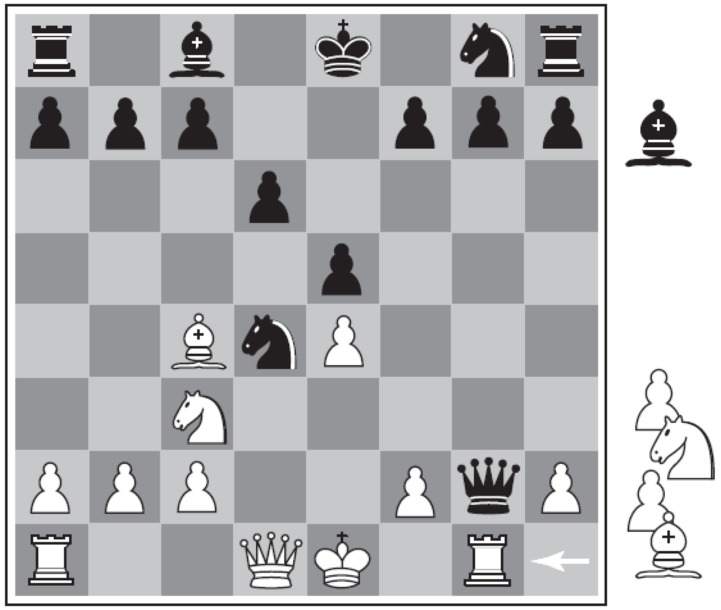
- An estimate of the utility is returned rather than the actual utility
- Define features and a weighted feature combination function
  - e.g., EVAL(s) = $w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$
- Chess
  - Pawn: 1, Knight or bishop: 3, rook: 5, queen: 9

Heuristic Evaluation Functions

- Just like in Chapter 3, heuristic functions are not part of the problem description
  - It takes additional expertise and learning to devise useful heuristic functions
- They are not very easy to define

Linear Combination Fails


(a) White to move


(b) White to move

Cutting-Off is not Trivial

- A simple depth limit will not work
  - Cut-off at positions that are quiescent
    - Positions in which a drastic change is unlikely to happen
- Horizon effect
  - A serious damage is unavoidable but the search delays it by making little sacrifices
    - The little sacrifices are unnecessary as the serious damage is unavoidable but a depth-limited search cannot see it

Horizon Effect

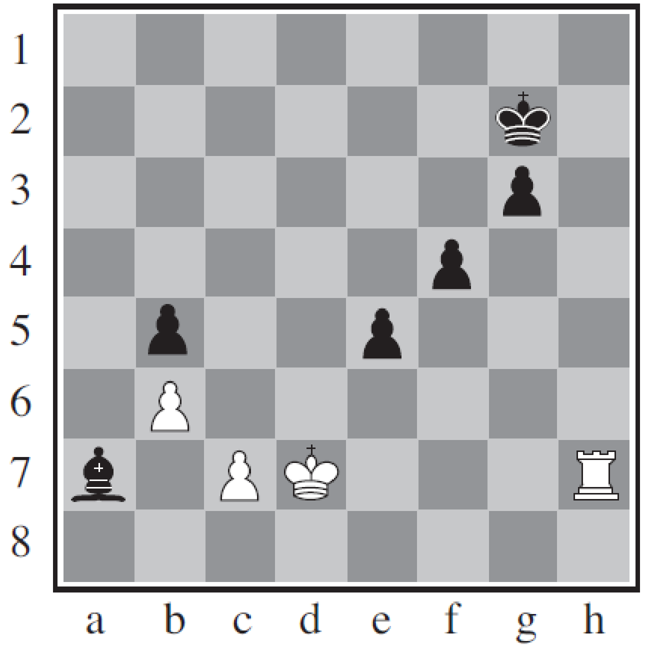


Figure 5.9 **FILES: figures/horizon.eps (Tue Nov 3 16:23:03 2009).** The horizon effect. With Black to move, the black bishop is surely doomed. But Black can forestall that event by checking the white king with its pawns, forcing the king to capture the pawns. This pushes the inevitable loss of the bishop over the horizon, and thus the pawn sacrifices are seen by the search algorithm as good moves rather than bad ones.

Search v.s. Lookup

- Opening of a game
  - A depth-limited search is unlikely to help
  - Human expertise and learning from previous games for opening moves
- Ending a game
  - When very few pieces are left, the number of possible states is not many; rather than a depth-limited search, look-up a winning strategy

So Far

- Deterministic
- Fully-observable

Stochastic Games

- An element of chance, such as dice
- Modify the search tree and the minimax algorithm:
  - Include a chance node, that leads to all possible outcomes
  - Rather than computing utilities, compute expected utilities

Partially Observable Games

- The current state is only partially observable
- Model what is not observed
- Examples
  - Card games
  - Battleships
  - Kriegspiel

State of the Art

- Chess:
  - IBM's Deep Blue
    - Parallel computation on 30 computers
    - Alpha-beta search
    - Custom VLSI processors
    - Searched up to 30 billion positions per move, reaching up to depth of 14
    - Allowed going deeper on certain states
    - Evaluation function had 8000 features
    - Opening book of 4000 positions was used
  - Others
    - World Computer Chess Championship

State of the Art

- Checkers:
  - Chinook: Plays perfectly with alpha-beta search and 39 trillion endgame positions
- Backgammon:
  - TD-GAMMON learned a good heuristic function by playing more than a million training games against itself
- Go:
  - vs. Chess
    - Board size:19x19=361 vs 8x8 = 64
    - Average branching factor: ~150-200 versus ~35
  - https://en.wikipedia.org/wiki/Go_(game)#Software_players

Deep Blue Beat G. Kasparov in 1997

- http://www.youtube.com/watch?v=NJarxpYyoFI