

Adversarial Search

Tomáš Svoboda

Department of Cybernetics, Vision for Robotics and Autonomous Systems,
Center for Machine Perception (CMP)

March 14, 2018

Games, man vs. algorithm

- ▶ Deep Blue
- ▶ Alpha Go
- ▶ Deep Stack
- ▶ Why Games, actually?

Games are interesting for AI *because* they are hard (to solve).

More: Adversarial Learning

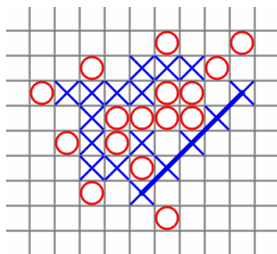


Video: Adversing visual segmentation

Vision for Robotics and Autonomous Systems, <http://cyber.felk.cvut.cz/vras>

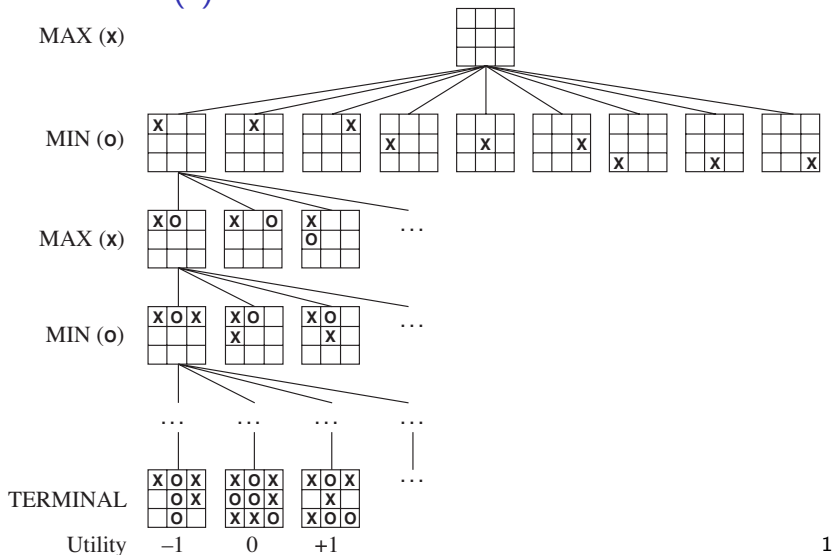
Elements of the game

- ▶ S_0 : The initial state
- ▶ $\text{PLAYER}(s)$. Which player has to move in s .
- ▶ $\text{ACTIONS}(s)$. What are the legal moves?
- ▶ $\text{RESULT}(s, a)$. Transition, result of a move.
- ▶ $\text{TERMINAL-TEST}(s)$. Game over?
- ▶ $\text{UTILITY}(s, p)$. What is prize? Examples for some games ...



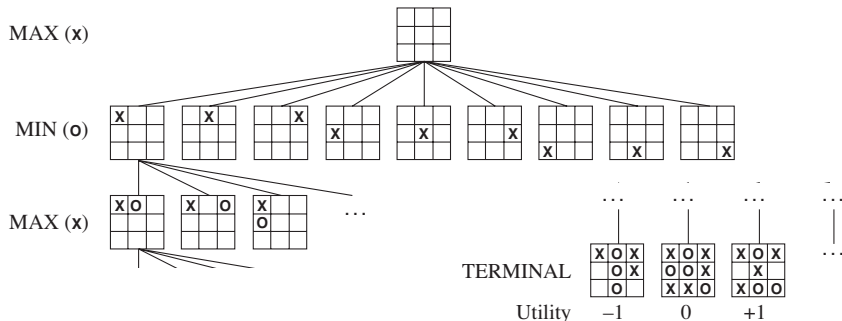
https://commons.wikimedia.org/wiki/File:Tic-tac-toe_5.png

Game Tree(s)



Init state, ACTIONS function, and RESULT function defines game tree

How do we play - **max** for me, **min** for the opponent.



$$\begin{aligned}
 \text{MINIMAX}(s) = & \\
 & \text{UTILITY}(s) \quad \text{if} \quad \text{TERMINAL-TEST}(s) \\
 & \max_{a \in \text{ACTIONS}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) \quad \text{if} \quad \text{PLAYER}(s) = \text{MAX} \\
 & \min_{a \in \text{ACTIONS}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) \quad \text{if} \quad \text{PLAYER}(s) = \text{MIN}
 \end{aligned}$$

Minimax algorithm

```
function MINIMAX(state) returns an action
  return  $\operatorname{argmax}_{a \in \text{Actions}(s)}$  MIN-VALUE(RESET(state, a))
end function
```

```
function MIN-VALUE(state) returns a utility value  $v$ 
  if TERMINAL-TEST(state) then return UTILITY(state)
  end if
   $v \leftarrow \infty$ 
  for all ACTIONS(state) do
     $v \leftarrow \min(v, \text{MAX-VALUE}(\text{RESULT}(\text{state}, a)))$ 
  end for
end function
```

```
function MAX-VALUE(state) returns a utility value  $v$ 
  if TERMINAL-TEST(state) then return UTILITY(state)
  end if
   $v \leftarrow -\infty$ 
  for all ACTIONS(state) do
     $v \leftarrow \max(v, \text{MIN-VALUE}(\text{RESULT}(\text{state}, a)))$ 
  end for
end function
```

A two ply game, down to terminal and back again ...

function MINIMAX(s) **returns** a MAX
 $\operatorname{argmax}_{a \in \text{Actions}(s)} \text{MINVAL}(\text{RES}(s, a))$

end function

function MINVAL(s) **returns** v MIN

if TERMINAL(s) **then** UTIL(s)

end if

$v \leftarrow \infty$

for all ACTIONS(s) **do**

$v \leftarrow \min(v, \text{MAXVAL}(\text{RES}(s, a)))$

end for

end function

function MAXVAL(s) **returns** v

if TERMINAL(s) **then** UTIL(s)

end if

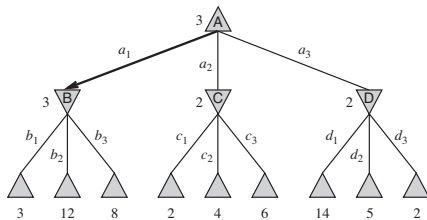
$v \leftarrow -\infty$

for all ACTIONS(s) **do**

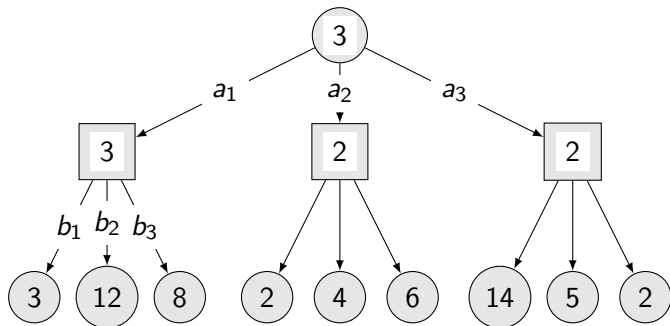
$v \leftarrow \max(v, \text{MINVAL}(\text{RES}(s, a)))$

end for

end function



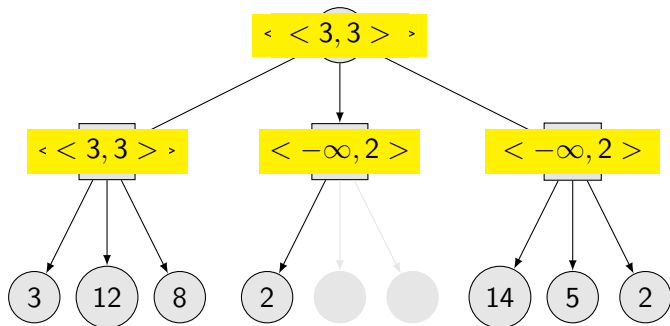
A two ply game, revisited



What is the complexity? How many nodes to visit?

Can we do better? How?

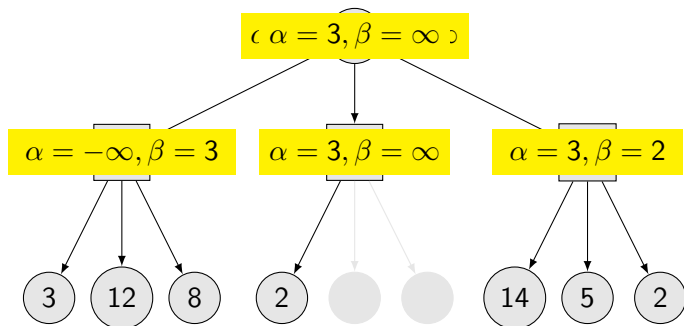
Nodes worth visiting



α - β pruning

α highest value choice found so far for any choice along MAX

β lowest value choice found so far for any choice along MIN



In MIN-VAL: $v \leftarrow 2$

$v \leq \alpha$ then: return v !

function ALPHA-BETA-SEARCH(*state*) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, \infty)$
 return the action in $\text{ACTIONS}(\text{state})$ with value v
end function

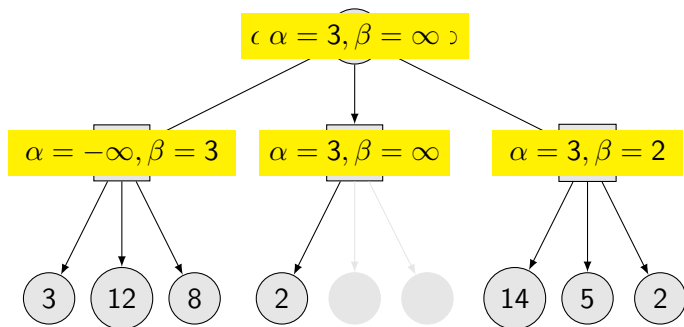
function MAX-VALUE(*state*, α , β) **returns** a utility value v
 if $\text{TERMINAL-TEST}(\text{state})$ **return** $\text{UTILITY}(\text{state})$
 $v \leftarrow -\infty$
 for all $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{max}(v, \text{MIN-VALUE}(\text{RESULT}(\text{state}, a), \alpha, \beta))$
 if $v \geq \beta$ **return** v
 $\alpha \leftarrow \text{max}(\alpha, v)$
 end for
end function

function MIN-VALUE(*state*, α , β) **returns** a utility value v
 if $\text{TERMINAL-TEST}(\text{state})$ **return** $\text{UTILITY}(\text{state})$
 $v \leftarrow \infty$
 for all $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{min}(v, \text{MAX-VALUE}(\text{RESULT}(\text{state}, a), \alpha, \beta))$
 if $v \leq \alpha$ **return** v
 $\beta \leftarrow \text{min}(\beta, v)$
 end for
end function

Move ordering

α highest value choice found so far for any choice along MAX

β lowest value choice found so far for any choice along MIN



Imperfect but real-time decisions

$$\begin{aligned} \text{H-MINIMAX}(s, d) = & \\ & \text{EVAL}(s) \quad \text{if } \text{CUTOFF-TEST}(s, d) \\ \max_{a \in \text{ACTIONS}(s)} \text{H-MINIMAX}(\text{RESULT}(s, a), d + 1) & \quad \text{if } \text{PLAYER}(s) = \text{MAX} \\ \min_{a \in \text{ACTIONS}(s)} \text{H-MINIMAX}(\text{RESULT}(s, a), d + 1) & \quad \text{if } \text{PLAYER}(s) = \text{MIN} \end{aligned}$$

Cutting off search

Replace

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

with:

if CUTOFF-TEST(s,d) **then return** EVAL(s)

Cutting depends on d only, why we need s as the input parameter?

function ALPHA-BETA-SEARCH(*state*) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, \infty)$
 return the action in $\text{ACTIONS}(\text{state})$ with value v
end function

function MAX-VALUE(*state*, α , β) **returns** a utility value v
 if $\text{TERMINAL-TEST}(\text{state})$ **return** $\text{UTILITY}(\text{state})$
 $v \leftarrow -\infty$
 for all $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{max}(v, \text{MIN-VALUE}(\text{RESULT}(\text{state}, a), \alpha, \beta))$
 if $v \geq \beta$ **return** v
 $\alpha \leftarrow \text{max}(\alpha, v)$
 end for
end function

function MIN-VALUE(*state*, α , β) **returns** a utility value v
 if $\text{TERMINAL-TEST}(\text{state})$ **return** $\text{UTILITY}(\text{state})$
 $v \leftarrow \infty$
 for all $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{min}(v, \text{MAX-VALUE}(\text{RESULT}(\text{state}, a), \alpha, \beta))$
 if $v \leq \alpha$ **return** v
 $\beta \leftarrow \text{min}(\beta, v)$
 end for
end function