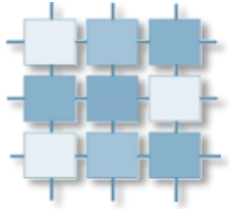
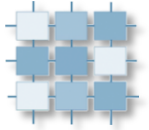




**OPPA European Social Fund
Prague & EU: We invest in your future.**



Two-players Games

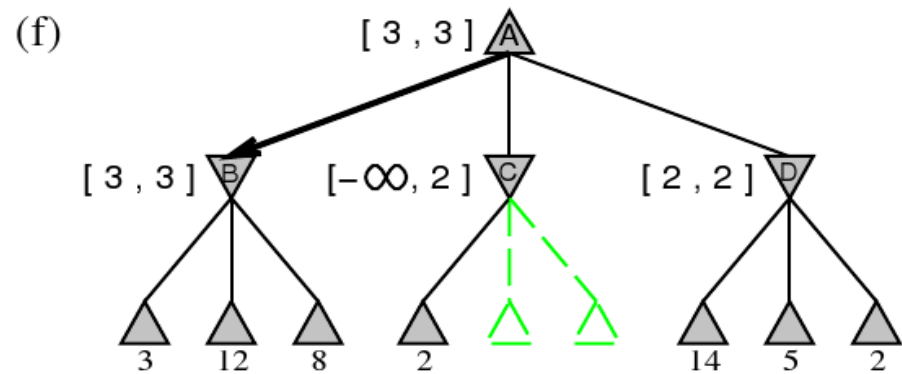
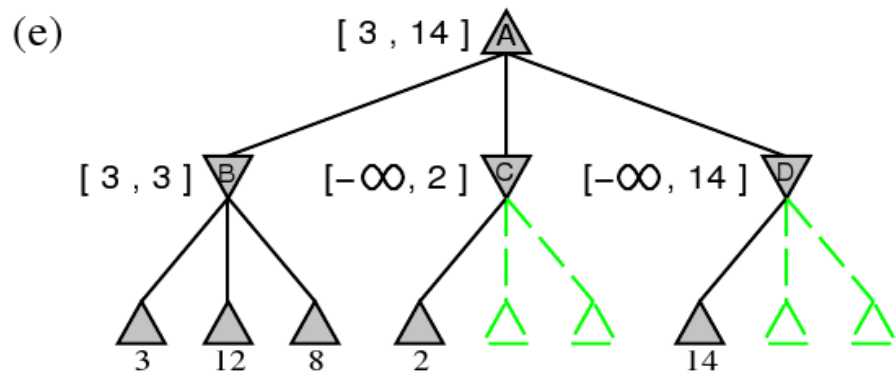
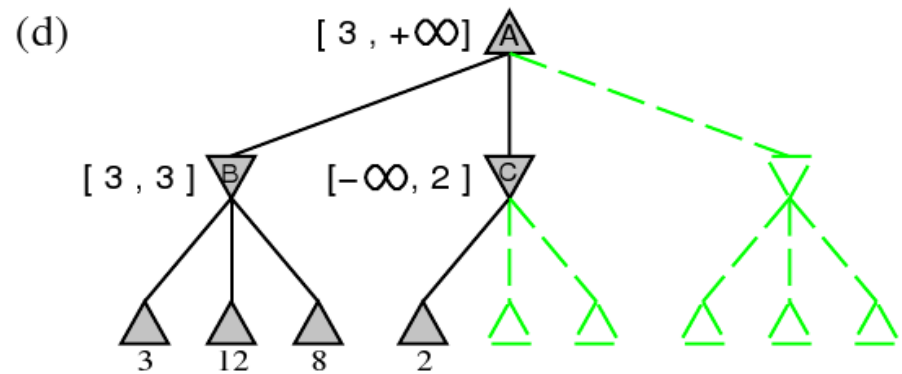
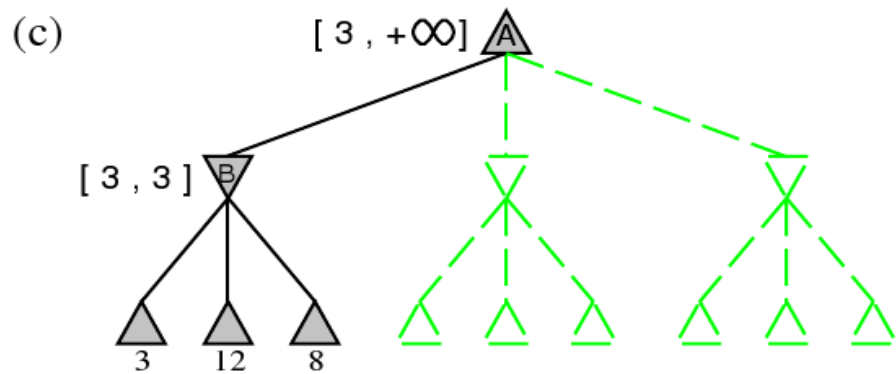
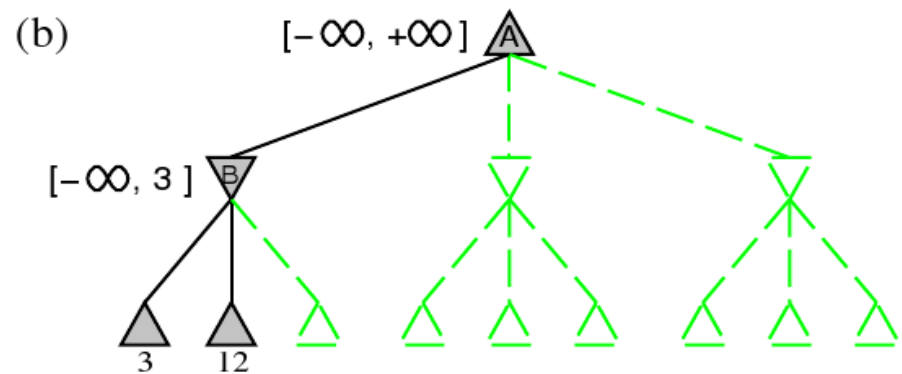
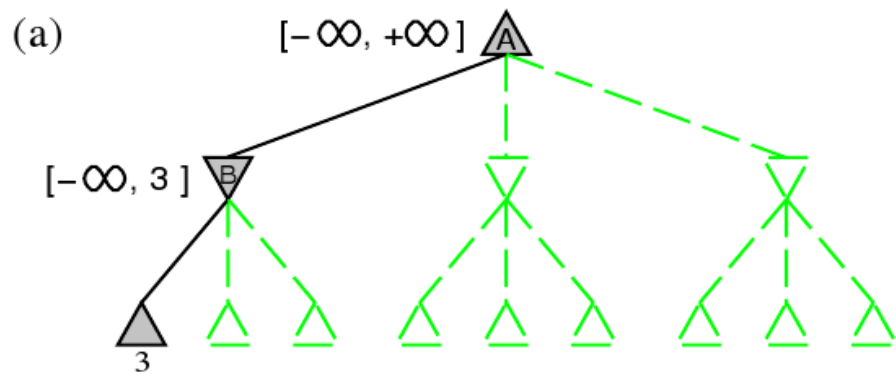


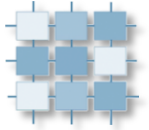
Reminder and

- two-players games
- zero-sum games
- no chance nodes
- perfect information

- chess, checkers, ...

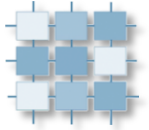
- note – the algorithms return a value, but we need a solution
- backup function





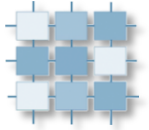
Reminder (2)

- **function** alphabeta(node, depth, α , β , Player)
- **if** (depth = 0 or node is a terminal node) **return** the heuristic value of node
- **if** (Player = MaxPlayer)
- **for each** child of node
- $\alpha := \max(\alpha, \text{alphabeta}(\text{child}, \text{depth}-1, \alpha, \beta, \text{not}(\text{Player}))$)
- **if** ($\beta \leq \alpha$) **break**
- **return** α
- **else**
- **for each** child of node
- $\beta := \min(\beta, \text{alphabeta}(\text{child}, \text{depth}-1, \alpha, \beta, \text{not}(\text{Player}))$)
- **if** ($\beta \leq \alpha$) **break**
- **return** β



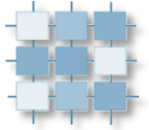
Step 1 - Negamax

- **function** **negamax**(node, depth, α , β , color)
- **if** (depth = 0 or node is a terminal node) **return** the heuristic value of node
- **if** (Player = MaxPlayer)
- **for each** child of node
- $\alpha := \max(\alpha, \text{negamax}(\text{child}, \text{depth}-1, -\beta, -\alpha, \text{color}))$
- **if** ($\beta \leq \alpha$) **break**
- **return** α
- **else**
- **for each** child of node
- $\beta := \min(\beta, \text{alphabeta}(\text{child}, \text{depth}-1, \alpha, \beta, \text{not}(\text{Player})))$
- **if** ($\beta \leq \alpha$) **break**
- **return** β



Step 2 – Aspiration Search

- $[\alpha, \beta]$ interval – window
- alphabeta initialization $[-\infty, +\infty]$
- what if we use $[\alpha_0, \beta_0]$
 - $x = \text{alphabeta}(\text{node}, \text{depth}, \alpha_0, \beta_0, \text{player})$
 - $\alpha_0 \leq x \leq \beta_0$ - we found a solution
 - $x \leq \alpha_0$ - failing low (run again with $[-\infty, x]$)
 - $x \geq \beta_0$ - failing high (run again with $[x, +\infty]$)

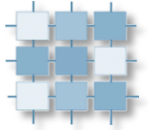


Step 3 – Scout – Idea

- assume we are in a MAX node
- we are about to search a child 'c'
- we already have obtained a lower bound ' α '

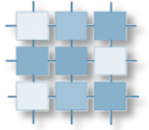
- Is it worth searching the branch 'c'?

- we need to have some test ...



Step 3 – Scout – A Test

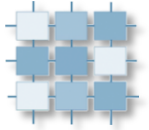
- what we really need at that moment is a bound (not the precise value)
- Remember Aspiration Search?
 - $x \leq \alpha_0$ - failing low (we know, that solution is $\leq x$)
 - $x \geq \beta_0$ - failing high (we know, that solution is $\geq x$)
- What if we use a null-window $[\alpha, \alpha+1]$ (or $[\alpha, \alpha]$)?
 - we obtain a bound ...



Step 3 – NegaScout

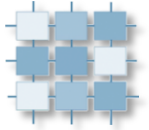
function negascout(node, depth, α , β , color)

- **if** ((depth = 0) or (node is a terminal node)) **return** eval(node)
- $b := \beta$
- **for each** child of node
- $v := -\text{negascout}(\text{child}, \text{depth}-1, -b, -\alpha, -\text{color})$
- **if** (($\alpha < v < \beta$) and (child is not the first child))
- $v := -\text{negascout}(\text{child}, \text{depth}-1, -\beta, -\alpha, -\text{color})$
- $\alpha := \max(\alpha, v)$
- **if** ($\beta \leq \alpha$) **break**
- $b := \alpha + 1$
- **return** α



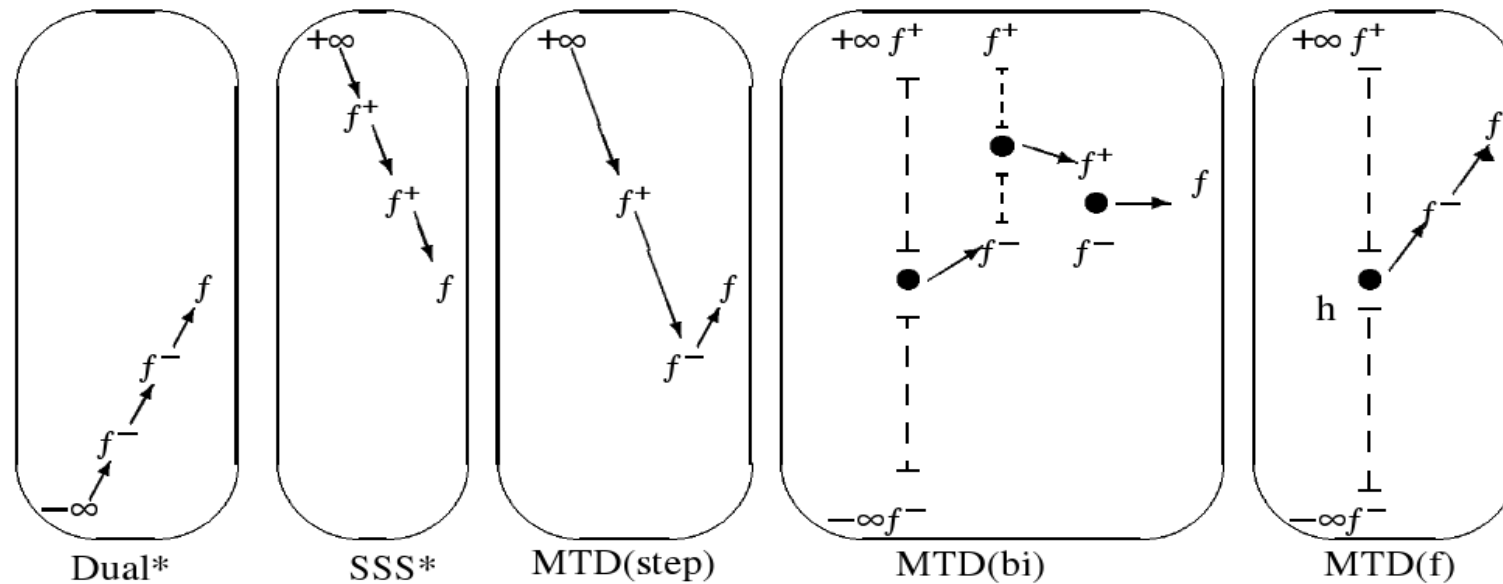
Step 3 – NegaScout

- also termed Principal Variation Search (PVS)
- dominates alphabeta (never evaluates more nodes than alphabeta)
- depends on the move ordering
- can benefit from transposition tables
- generally 10-20% faster compared to alpha-beta

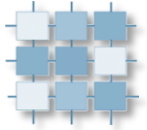


Step 4 – MTD

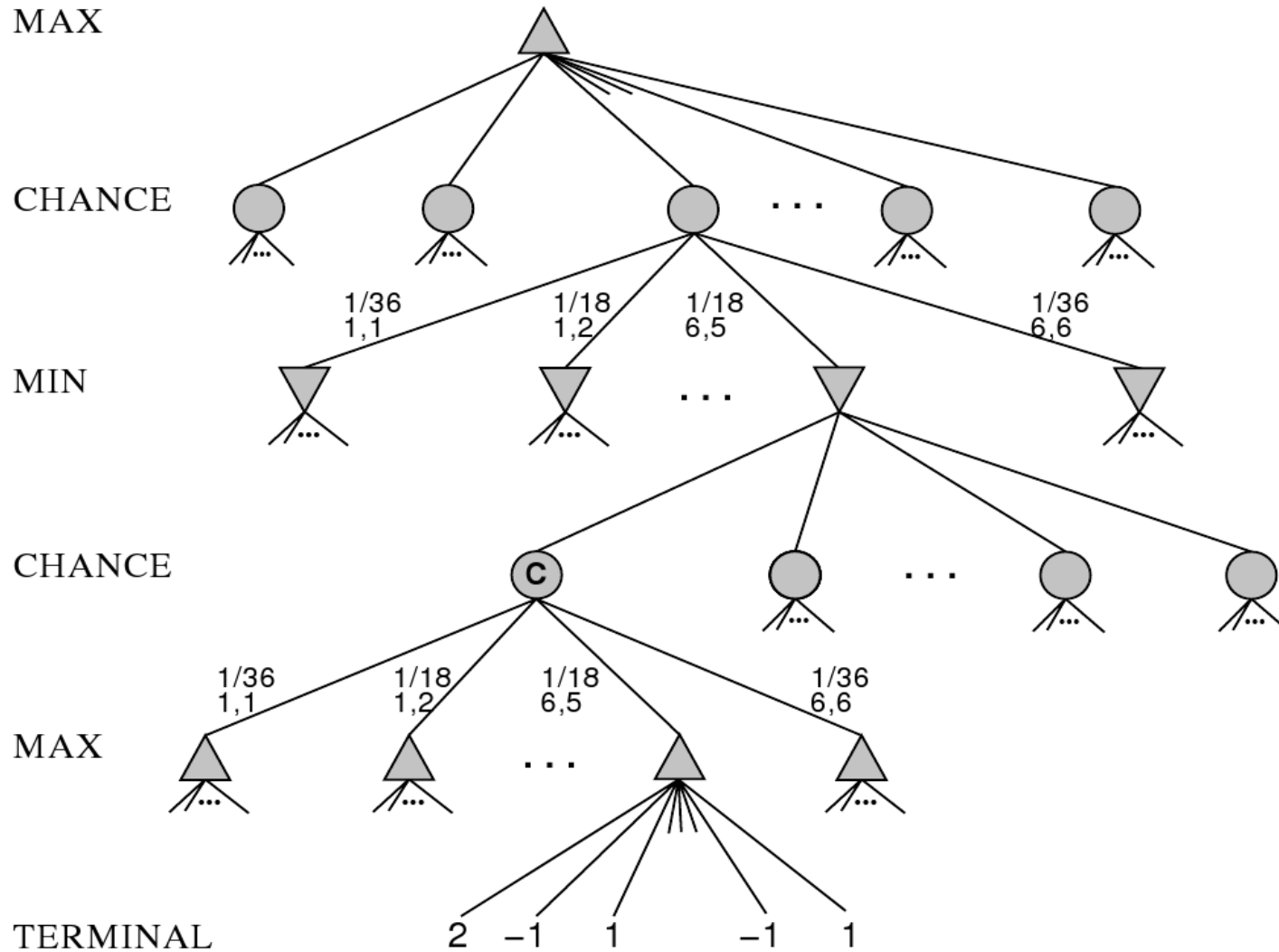
- Memory-enhanced Test Driver

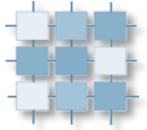


- Best-first fixed-depth minimax algorithms. Plaat et. al. , In *Artificial Intelligence*, Volume 87, Issues 1-2, November 1996, Pages 255-293

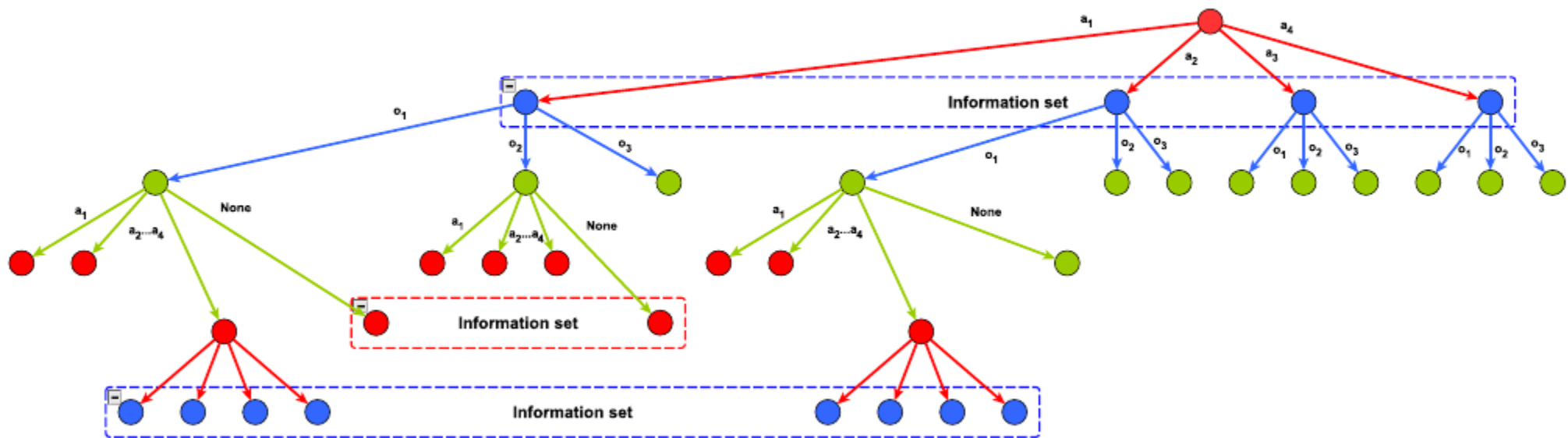


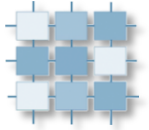
Other Games - Chance nodes





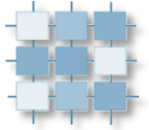
Other Games – Imperfect Information





Games and AI

- checkers – 1994 Chinook ... now a solved game (the program cannot lose)
 - chess – 1997 Deep Blue, ..., computers are now too strong
 - go – best human players are still undefeated, but ... (see <http://www.computer-go.info/h-c/index.html>)
 - poker – 2008 best program (Polaris) can beat a human master
 - ... and many, many others (Hex, Havannah, ...)
-
- University of Alberta
 - Computer Olympiad



Challenges?

- simultaneous moves, imperfect information
- durative moves (asynchronous chess, Google AI Challenge, ...)
- General Game Playing
 - an algorithm receives rules of the game and has to play
- ARIMAA (created in 2002)
 - $BF \approx 17,000$; no opening books; very few patterns
 - easy for people, very difficult for an algorithm
- using a 'real-AI-algorithms' in computer video-games
 - very few examples: F.E.A.R., World In Conflict, ...



**OPPA European Social Fund
Prague & EU: We invest in your future.**
