Adversariale Suche für optimales Spiel: Der Minimax-Algorithmus und die Alpha-Beta-Suche

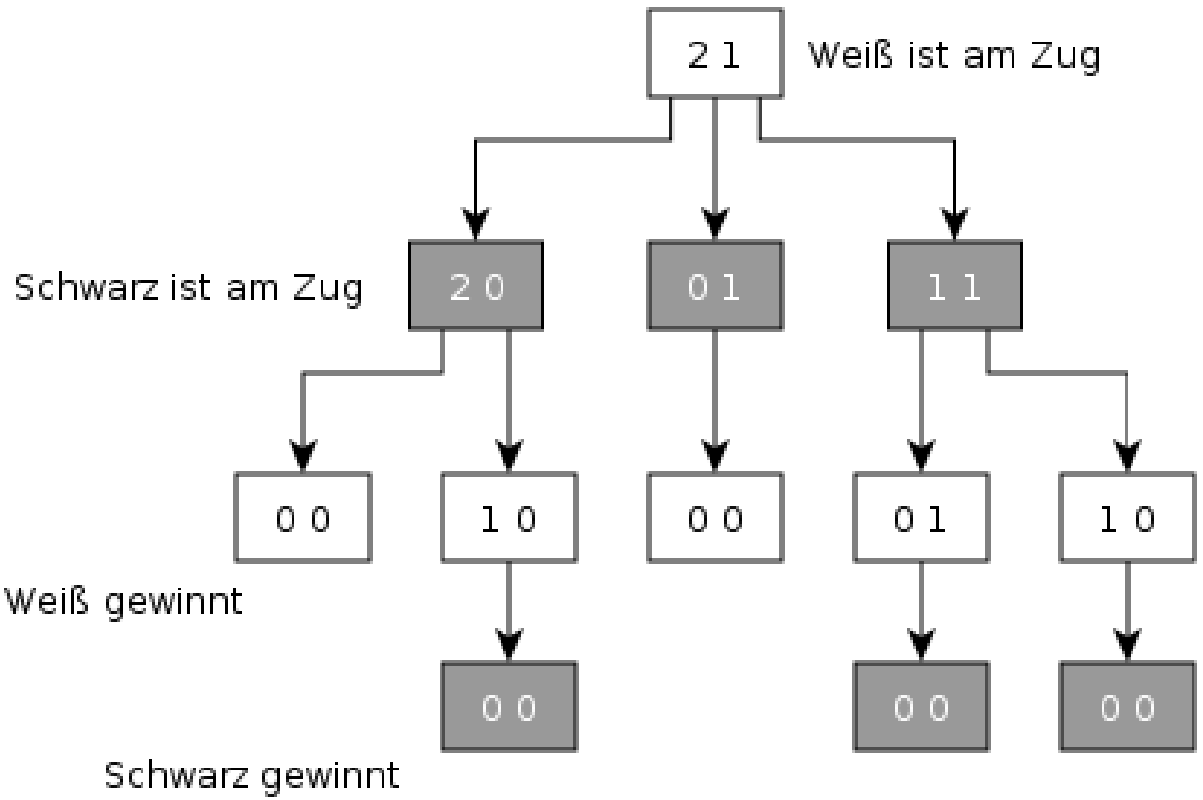
Julius Adorf

Fakultät für Informatik
Technische Universität München

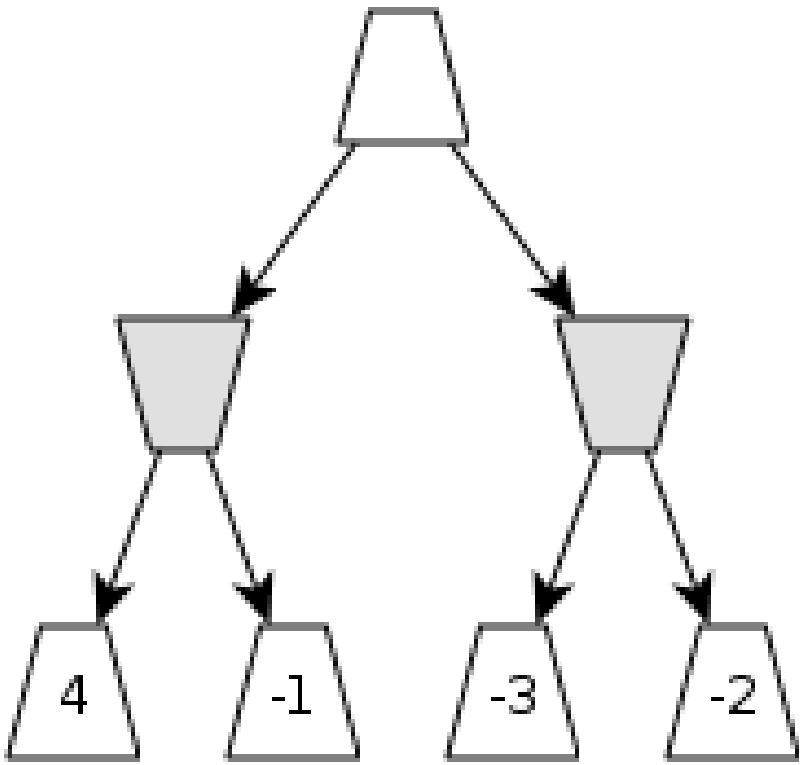
9.12.2009 / Proseminar Künstliche Intelligenz

Outline

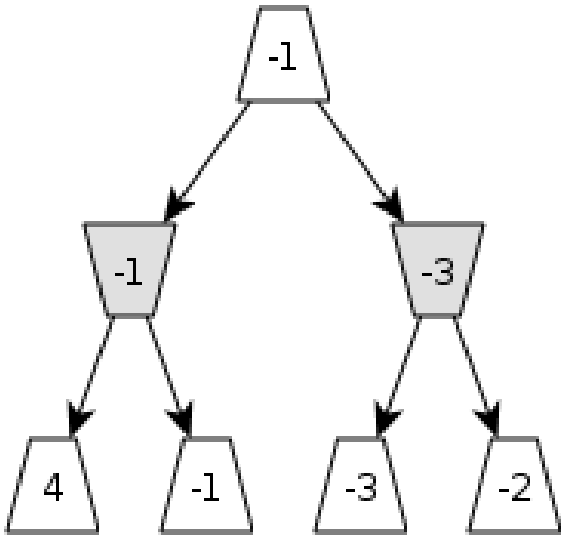
Spielbaum des Nim-Spiels



Spielbaum eines Zwei-Personen-Nullsummenspiels

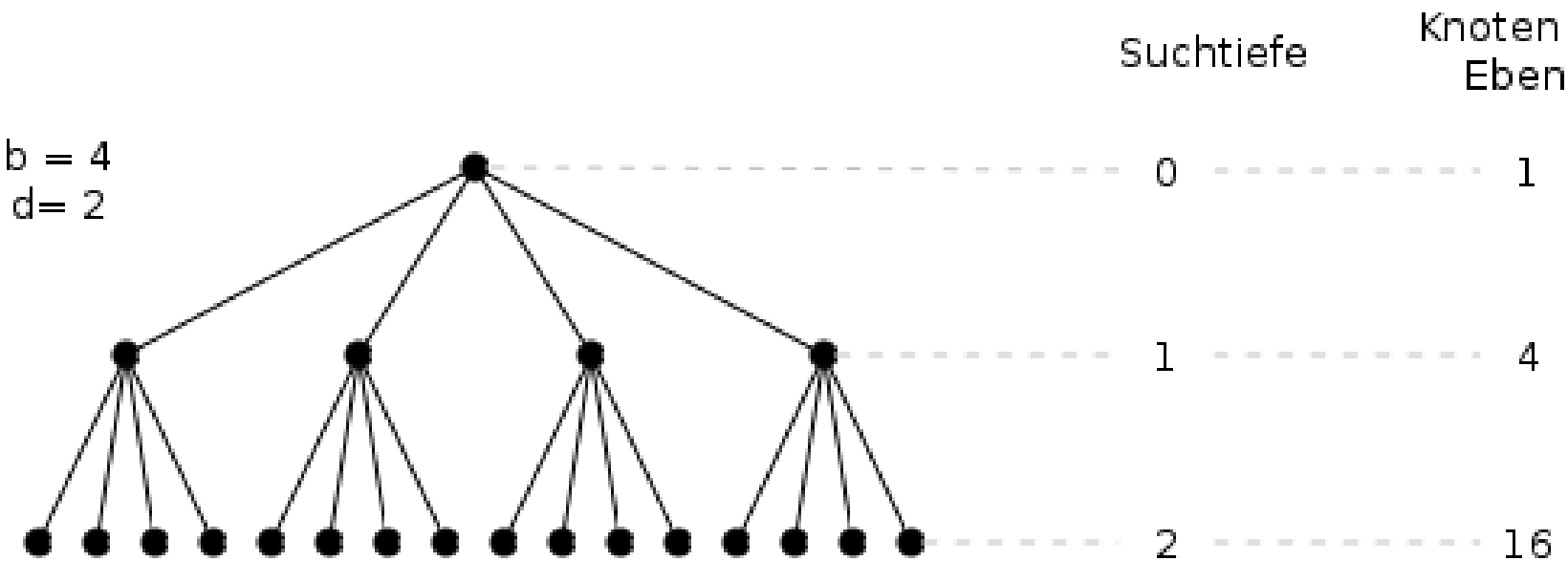


Minimax-Algorithmus



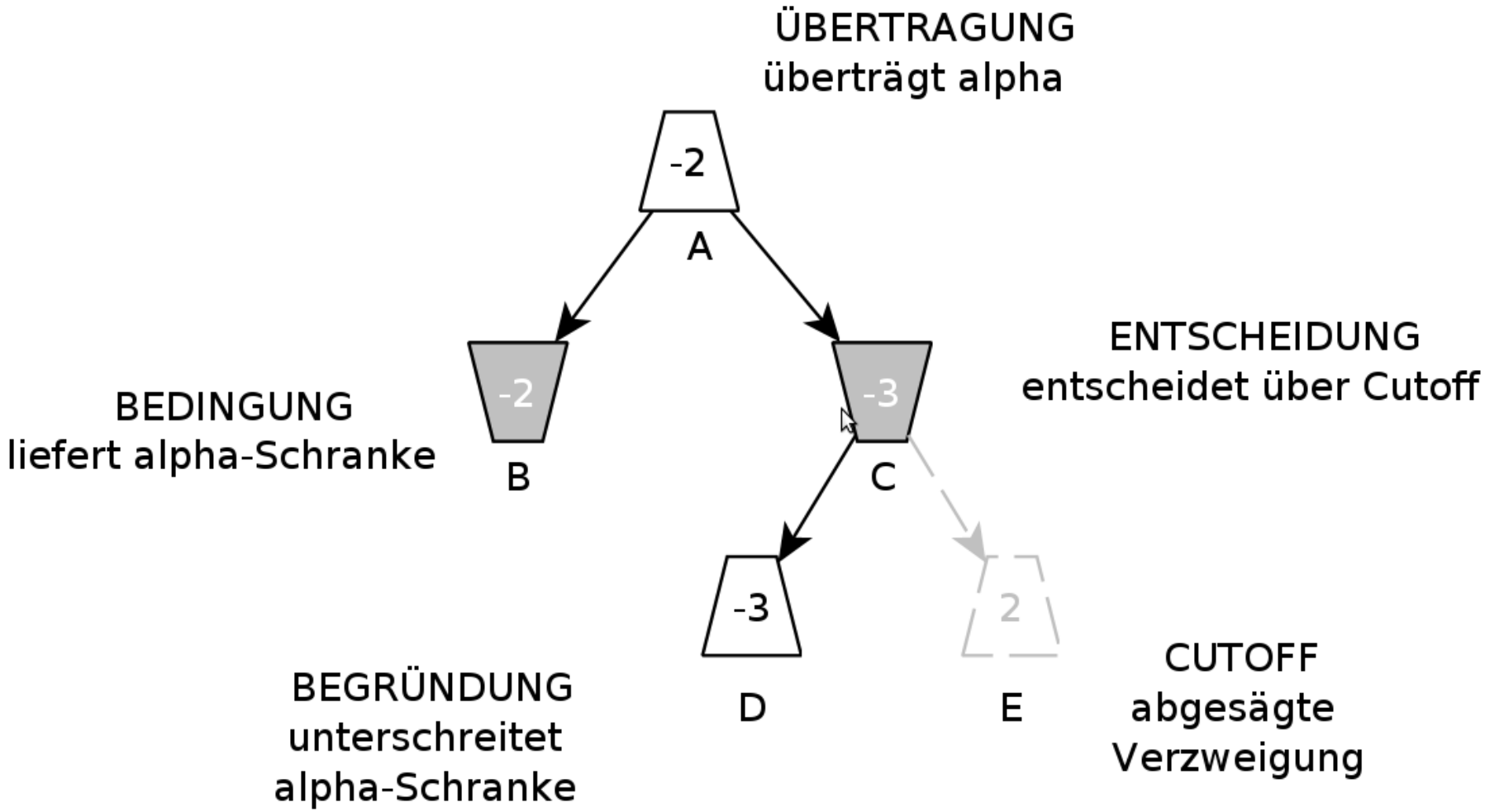
$$\text{minmax}(r) = \begin{cases} \text{utility}(r), & \text{falls r Endzustand} \\ \max\{\text{minmax}(s) \mid s \in \Gamma(r)\}, & \text{falls r MAX-Knoten} \\ \min\{\text{minmax}(s) \mid s \in \Gamma(r)\}, & \text{falls r MIN-Knoten} \end{cases}$$

Kombinatorische Explosion



Outline

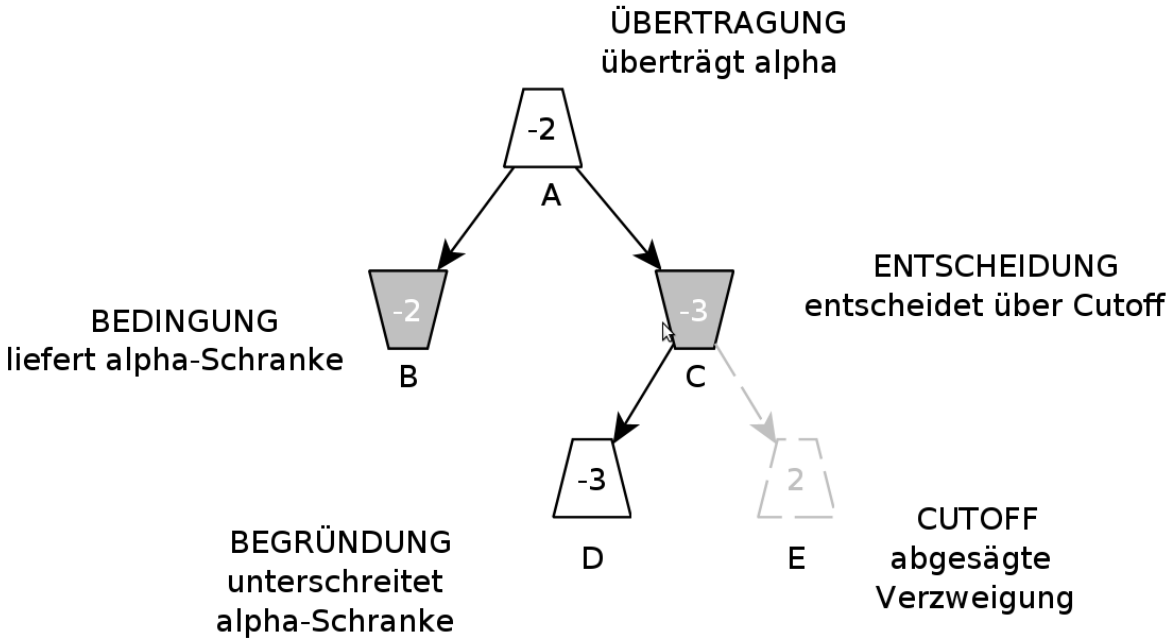
Minimalbeispiel



Alpha-Beta-Algorithmus

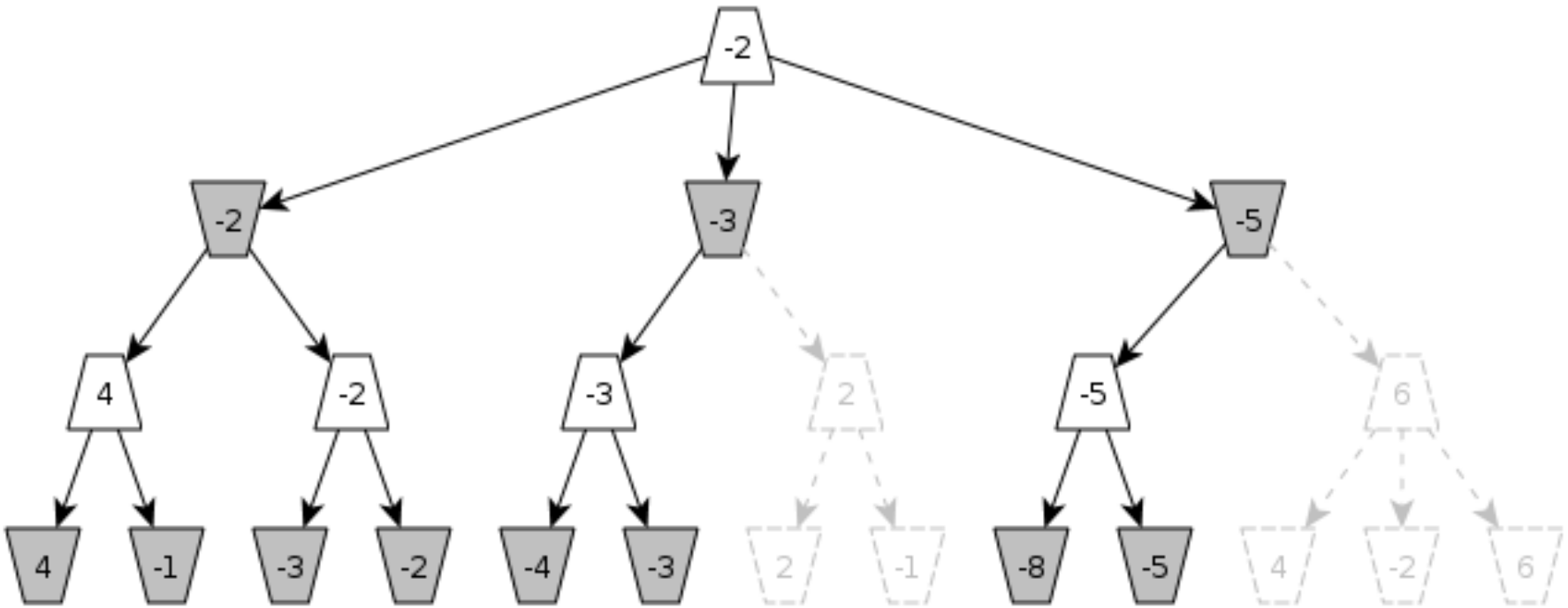
```
def alphabeta_search(node):
    return max_value(node, -sys.maxint - 1, sys.maxint)

def max_value(node, alpha, beta):
    if is_leaf(node):
        return value(node)
    else:
        v = alpha
        for c in children(node):
            v = max(v, min_value(c, v, beta))
            if v >= beta:
                return v
        return v

def min_value(node, alpha, beta):
    if is_leaf(node):
        return value(node)
    else:
        v = beta
        for c in children(node):
            v = min(v, max_value(c, alpha, v))
            if v <= alpha:
                return v
        return v
```

Minimalbeispiel




```
def min_value(node, alpha, beta):
    if is_leaf(node):
        return value(node)
    else:
        v = beta
        for c in children(node):
            v = min(v, max_value(c, alpha, v))
            if v <= alpha:
                return v
        return v
```

Effektivität des Alpha-Beta-Prunings

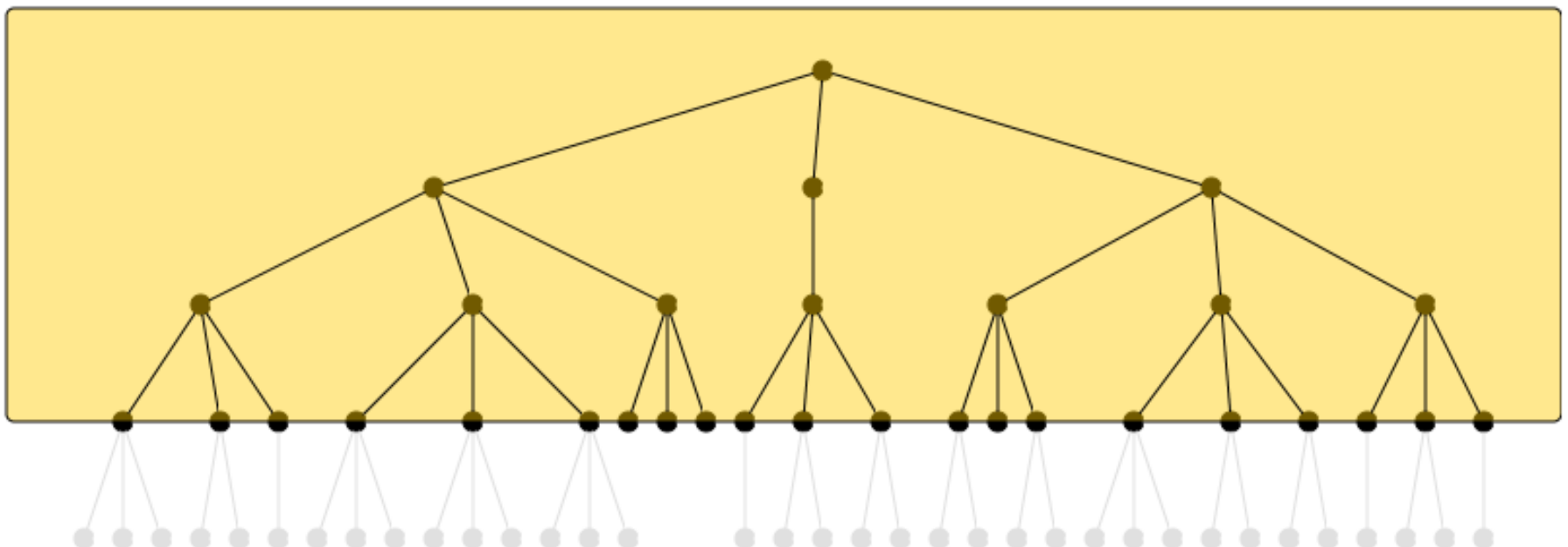


Zufällige Zugreihenfolge:

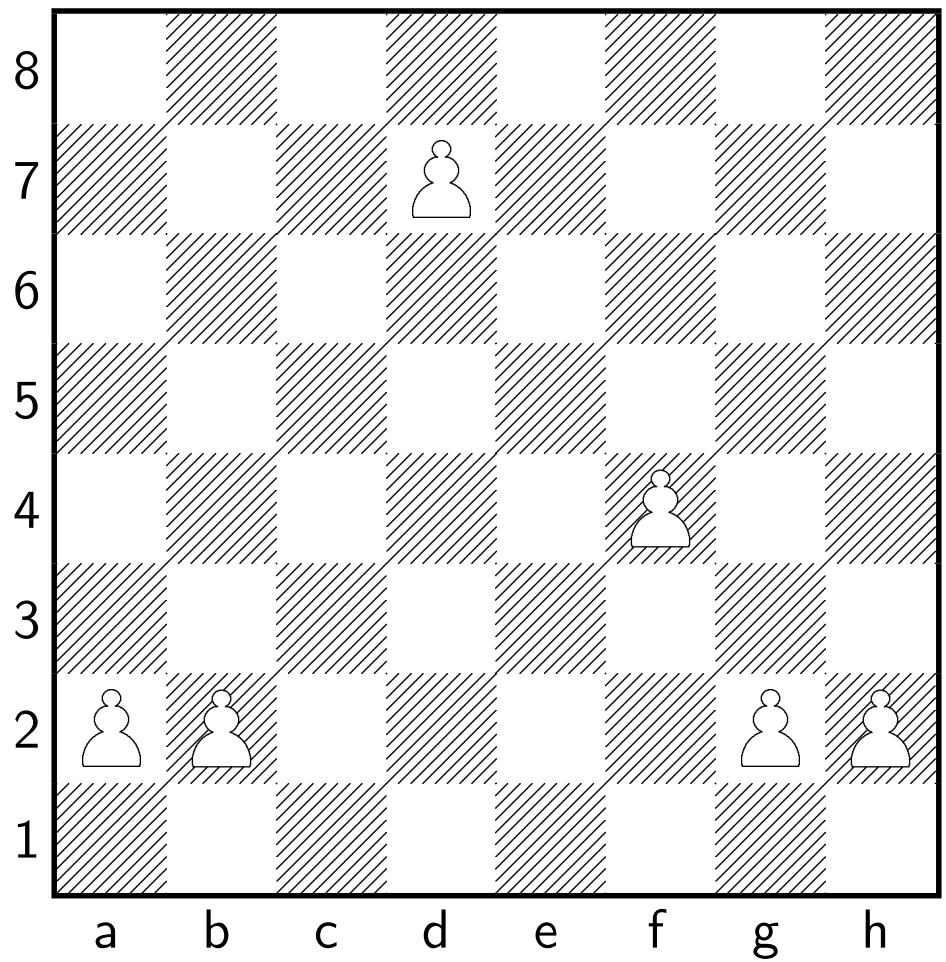
Minimax: $O(b^d)$, Alpha-Beta: $O(b^{\frac{3d}{4}})$

Outline

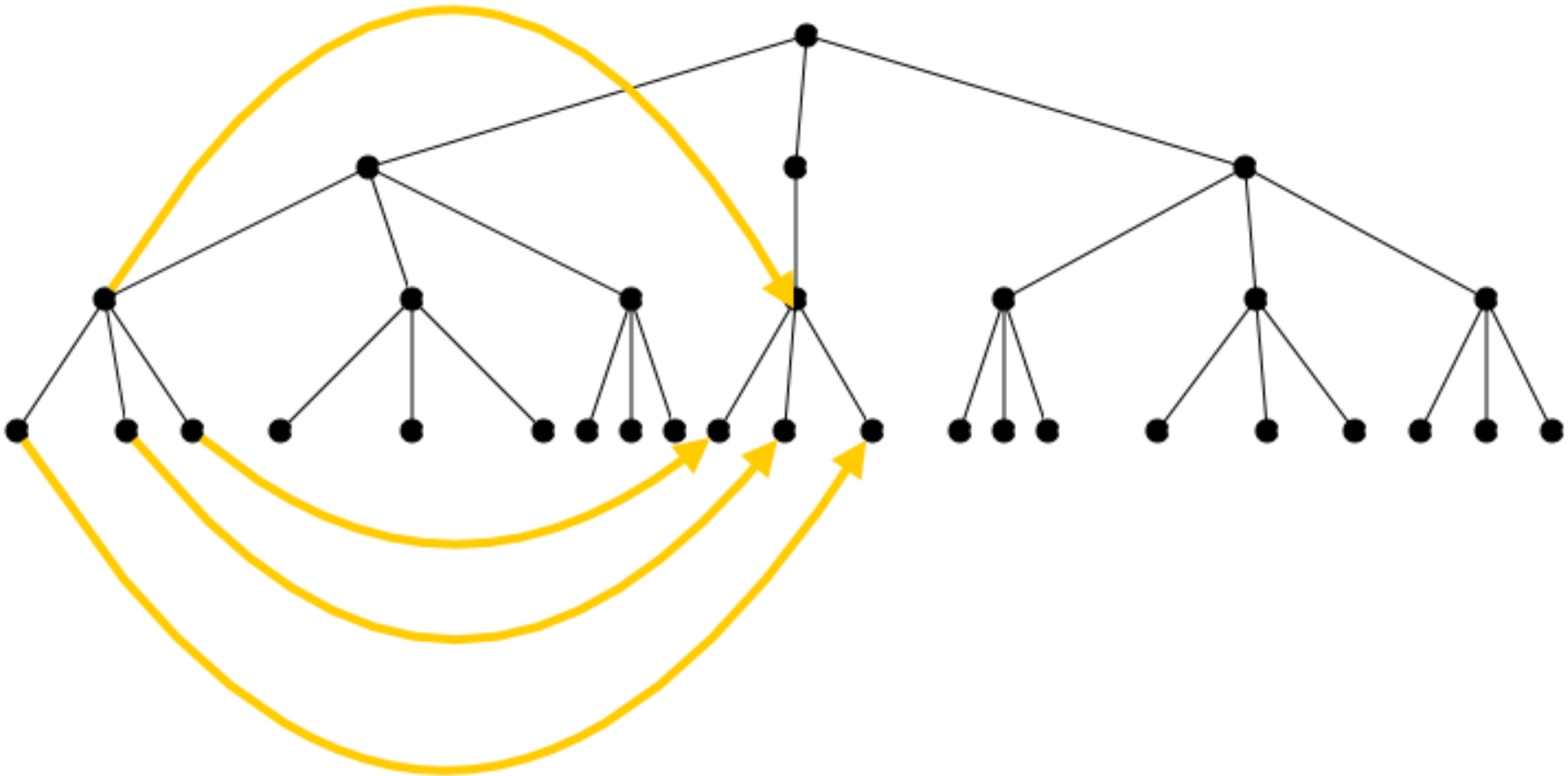
Suchhorizont



Datenstruktur

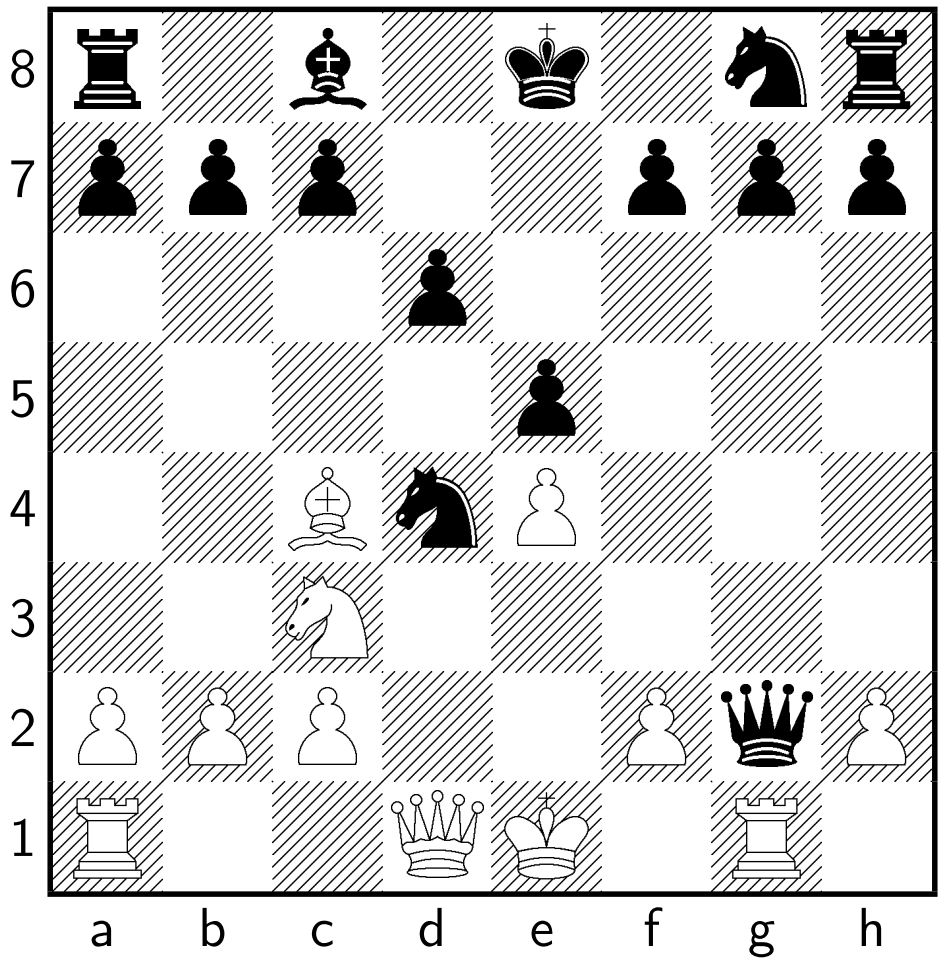


Transpositionstabellen



Ausnahmen: Rochaderecht, en-Passant-Regel!

Suche nach ruhigen Stellungen



Zusammenfassung

- Minimax-Algorithmus theoretisch interessant
- $\alpha\beta$-Suche praxistauglich
- Anwendungen bei Schach, Othello, Go, Go-Bang, Mühle