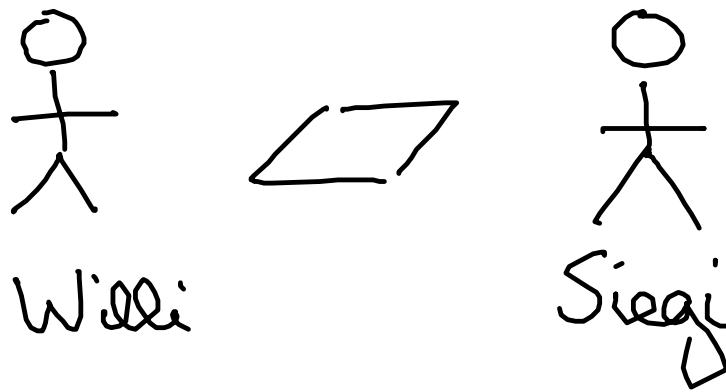


Spieltheorie – Optimale Spielstrategie

Spiele als Suchprobleme. Im Unterschied zu den bisher betrachteten Suchproblemen muss das mögliche Verhalten des Gegners in Betracht gezogen werden.



- 2-Spieler
- abwechselnd am Zug
- endliches Spiel (W, 0, S)
- endlich viele Aktionen
- der einen Gewinn ist der anderen Verlust

⇒ Nullsummenspiel mit vollständiger
Information (Perfect Knowledge Game)

Bsp: TicTactoe, Schach, Go, Dame, ...

Herausforderung

- Verhalten des Gegners ungewiss
- meistens kontraproduktiv
- Mangel an Zeit und Speicher

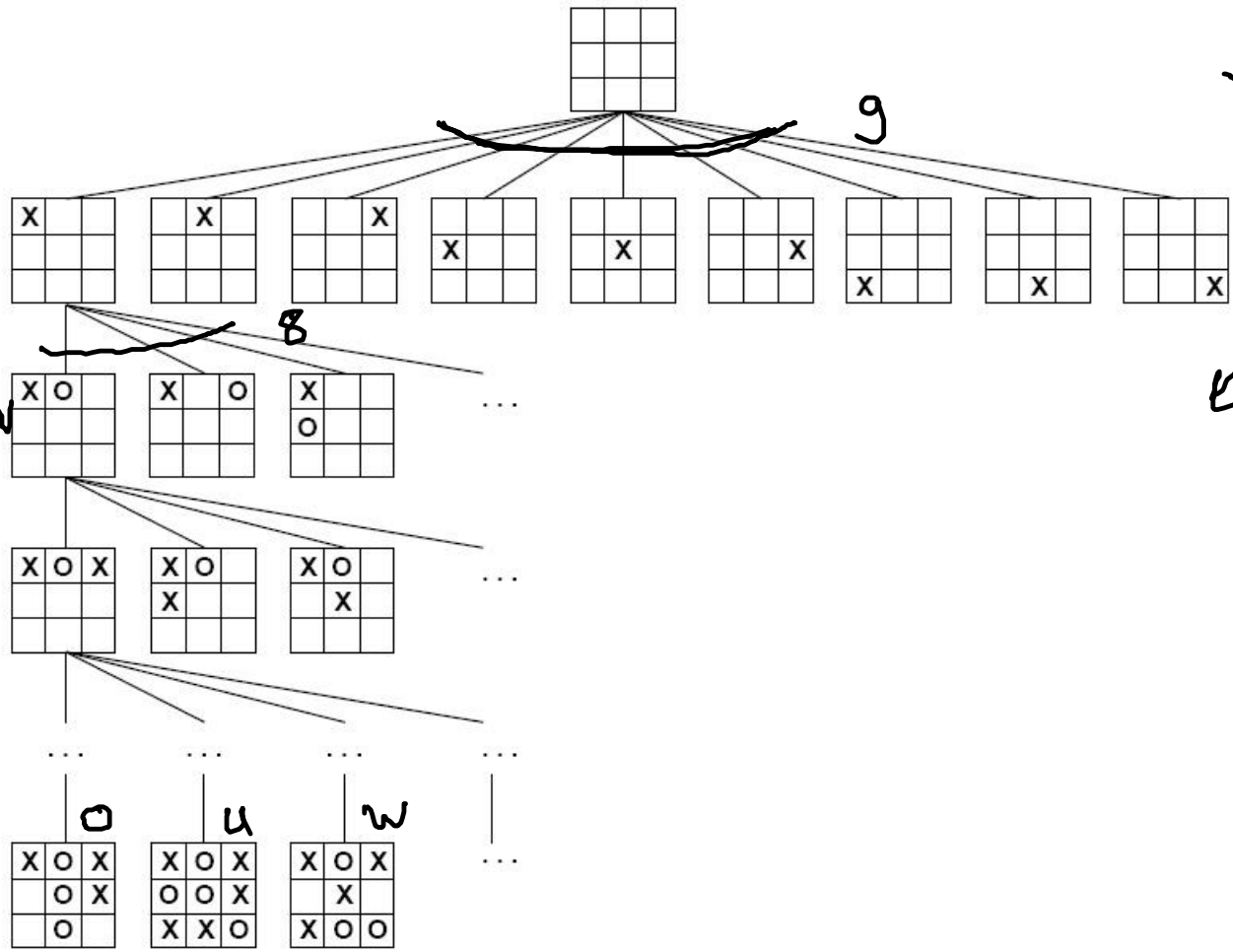
Ermittlung der besten

Spielstrategie

Tic Tac Toe :

X

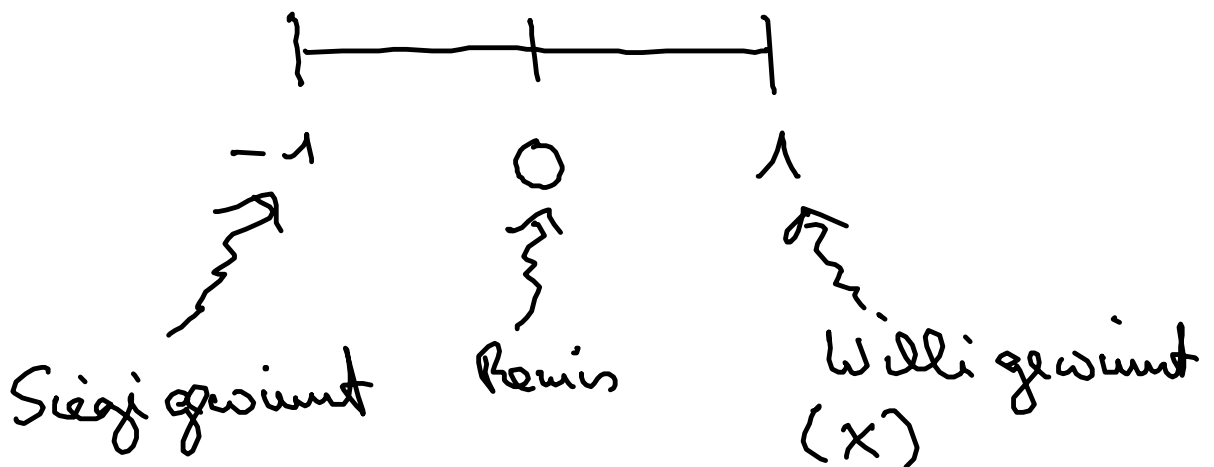
O

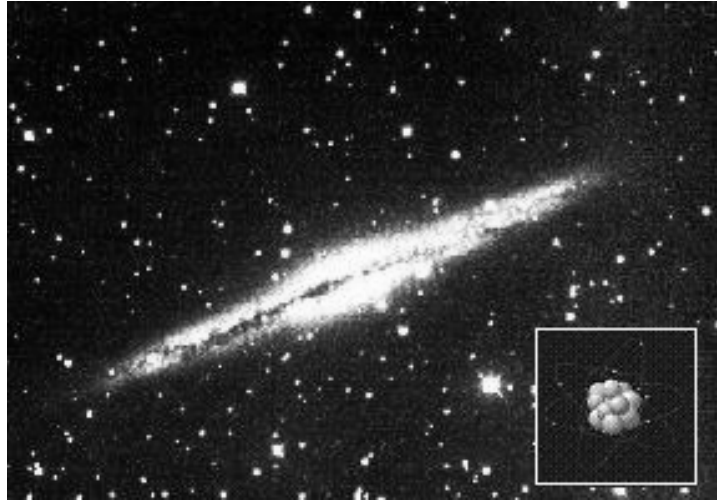


Heberzug
-u-

terminale Stellung

Definition

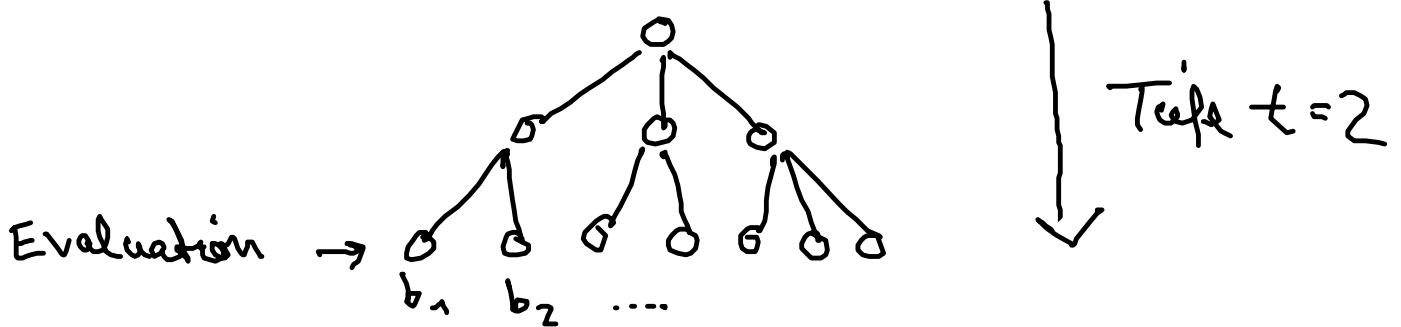




Atome im Weltall

$$\Rightarrow \sim 10^{80}$$

IDEE: - feste Suchtiefe
- Abschätzungsfunktion



Claude Edward Shannon



- Vortrag 1949
- Minimax + Evaluation

Bewertungsfunktion

Qualität des Spielprogramms hängt entscheidend von der Qualität dieser Funktion ab.

Achten auf:

- Effizienz (oft sehr teuer)
- möglichst unabhängige Kriterien
- Wiedergabe der tatsächlichen Gewinnchancen

meistens gewichtete, lineare Funktionen

$$f(S) = \sum_{i=1}^n \alpha_i f_i(S)$$

$$\dots f_1(S) = \text{material Weiss}(S) \\ - \text{material Schwarz}(S)$$

$$\dots f_2(S) = \text{mobilität W}(S) \\ - \text{mobilität S}(S)$$

→ α_1
Eröffnung

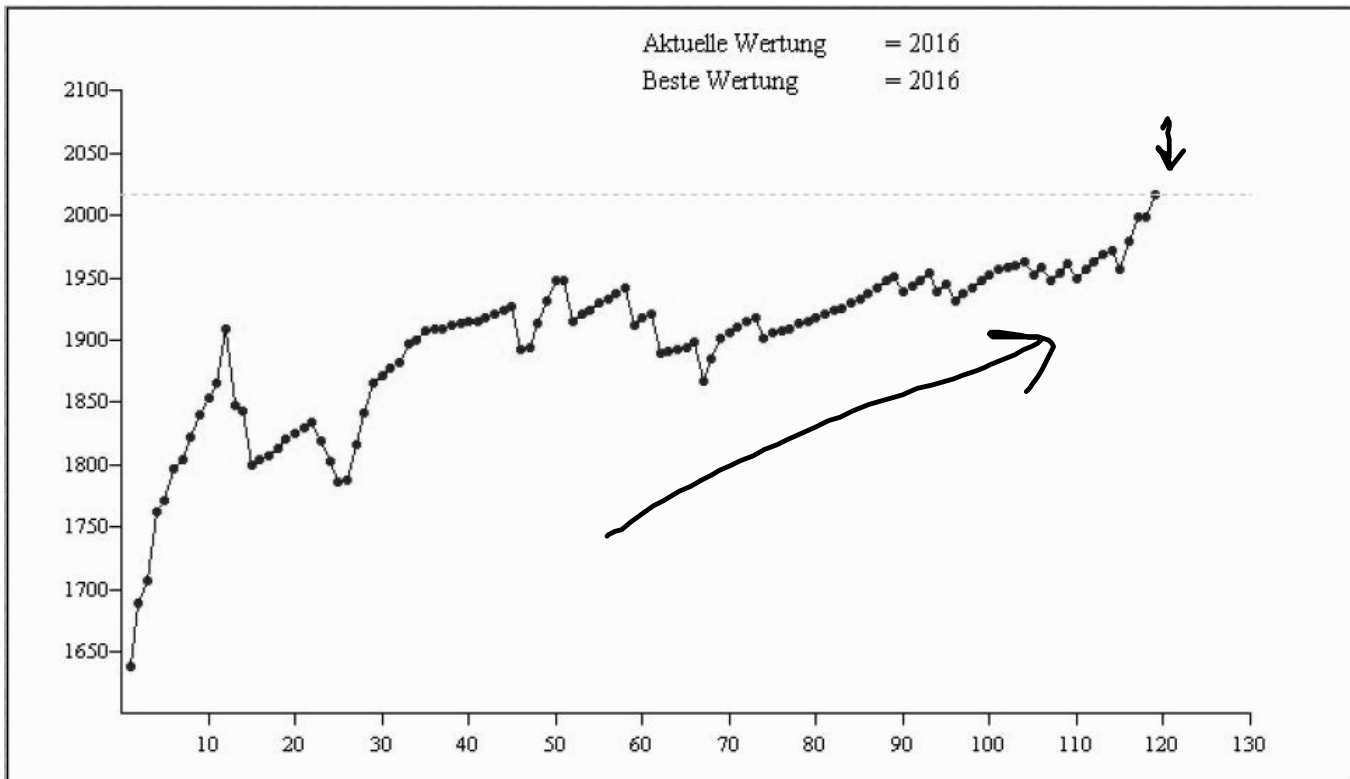
→ α_2
Mittelspiel

→ α_3
Endspiel

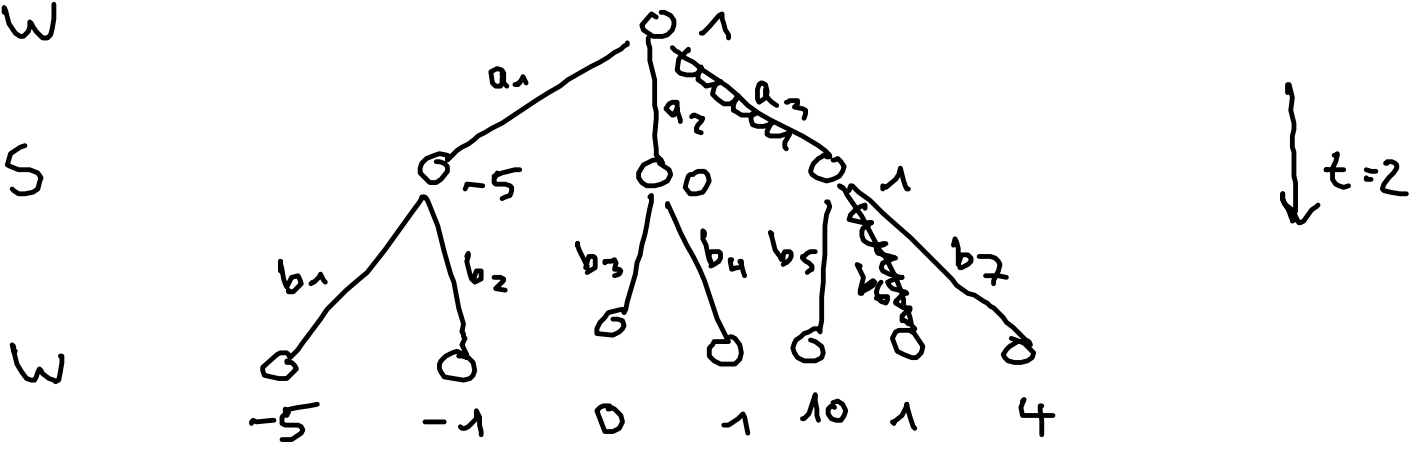
- König Zentrum
- Mobilisiert Dame

→ $\alpha_1 \dots \alpha_n$

Reinforcement Learning



MinMax - Algorithmus mit Bewertungsfunktion



Hauptvariante : $([a_3, b_6], 1)$

Pseudocode

Effizienz

```

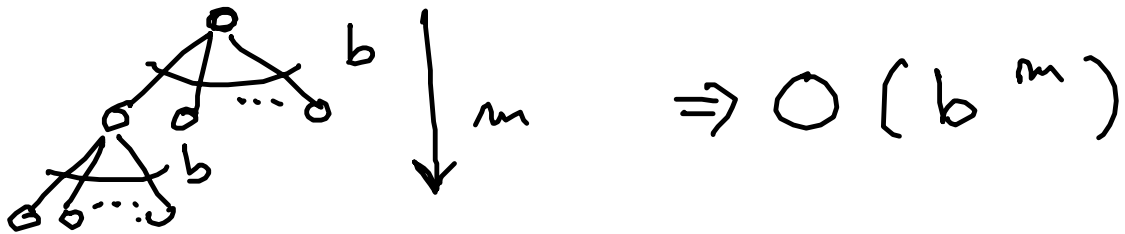
maxKnoten ( Knoten x ) → Integer {
  if ( x ist Blatt )
    return evaluation ( x )
  else {
    // x1 ... xk seien Kinder von x
    ω := -∞
    for i := 1 to k do {
      v := minKnoten ( xi )
      if ( v > ω )
        ω := v
    }
    return ω
  }
}

```

⇒ miniknoten entsprechend

Aufwandsabschätzung

Sei m maximale Spielbaumtiefe und
 b die durchschnittliche Anzahl der Spielzüge



MinMax in Prolog

minmax (Pos, BestSucc, Val) :-

 (moves(Pos, PosList), !,
 best(PosList, BestSucc, Val))

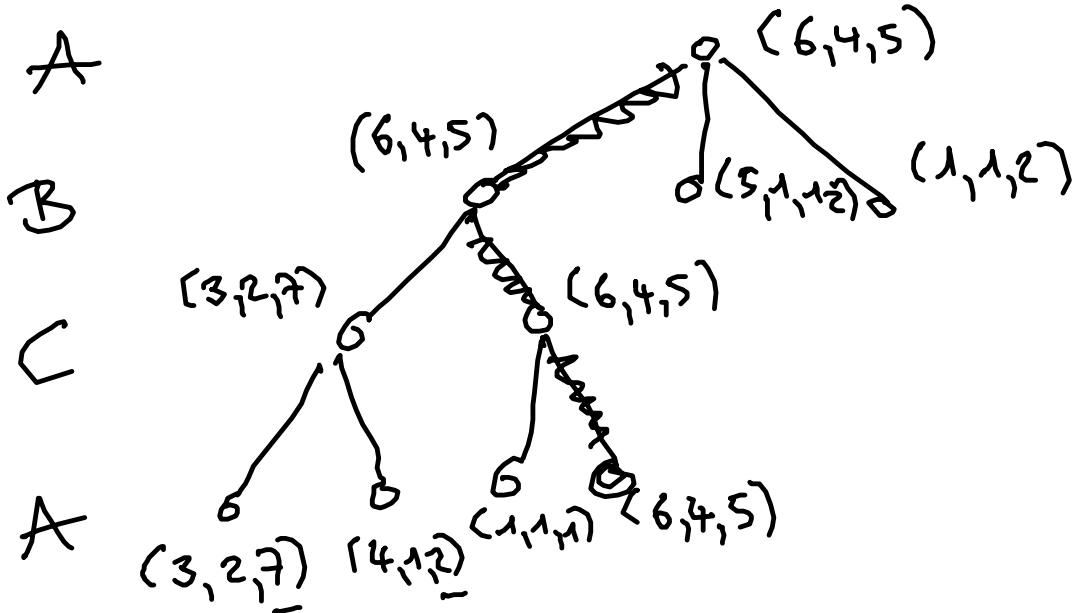
;

 evaluation(Pos, Val).

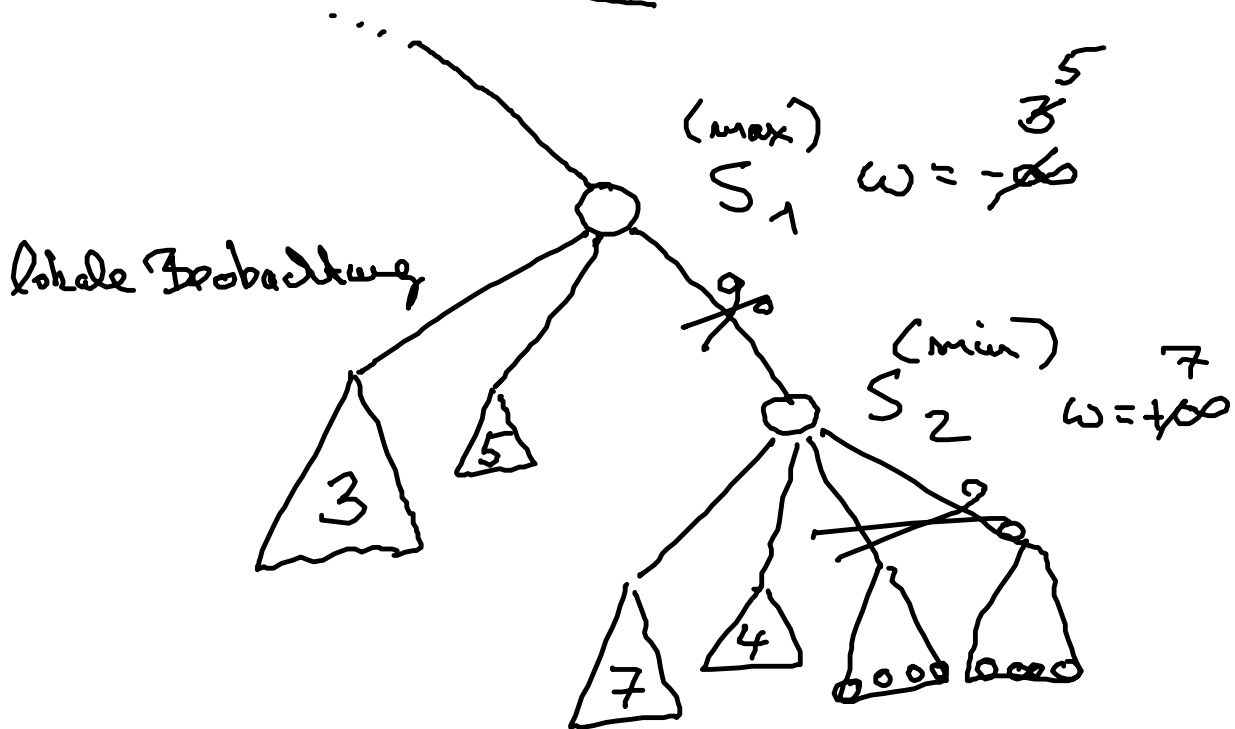
... (Tut)

Ausblick

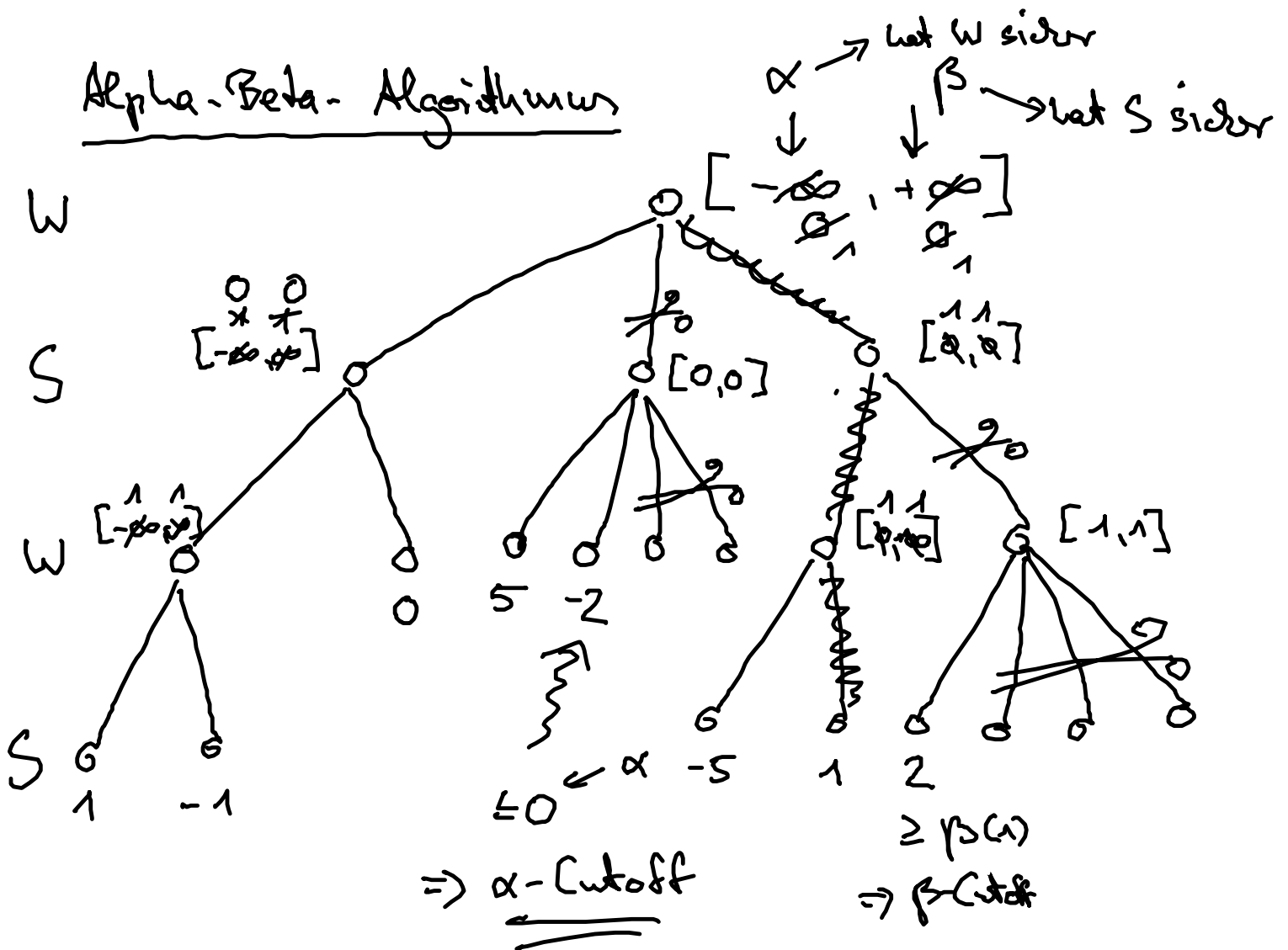
Min Max-Prinzip auch für rundenbasierte Spiele mit mehr als 2 Spielern anwendbar.



Alpha-Beta-Pruning



Alpha-Beta-Algorithmus



Pseudocode :

```

αβ maxKnoten (Knoten x, Integer α, β) → Int {
  if (x ist Blatt)
    return evaluation(x)
  else {
    w := α
  }
}
    
```

```

for i := 1 to k do {
  v :=  $\alpha \beta$  min Knoten( $x_i, \omega, \beta$ )
  if ( $v > \omega$ )
     $\omega := v$ 
  if ( $\omega \geq \beta$ ) //  $\beta$ -Cut-off
    return  $\beta$ 
}
return  $\omega$ 
}
}
}

```

$\Rightarrow \alpha \beta$ min Knoten entsprechend

Aufwandsabschätzung

worst case \rightarrow MinMax $O(b^m)$

best case $\rightarrow O(b^{\frac{m}{2}})$ wenn die
Zugreihenfolge optimal ist

\Rightarrow Verzweigungsfaktor
 \sqrt{b} statt b

Schach \Rightarrow ~~35~~ 6

Zufällig $\rightarrow O\left(\left(\frac{b}{\log b}\right)^m\right)$

Optimierungen + Heuristiken



• Zugsortierungen

IDEA: Züge, die sich als gut bewährt haben sollen Vorrang erhalten

- Killermoves
- Historymoves
- Hauptvariante

• Iterative Suchverfahren

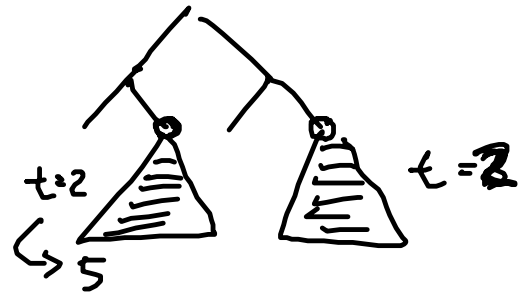
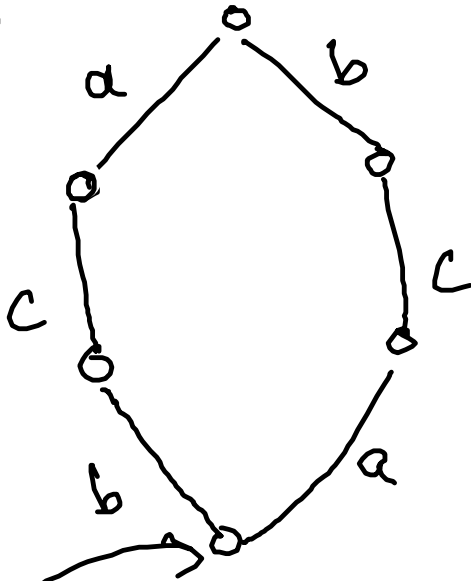
schlecht



gut

Δ 1
:
 Δ 2
:
 Δ t

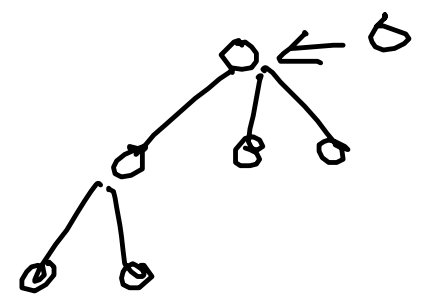
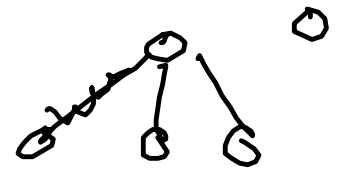
Suchgraphen! (keine Bäume)



key(S)

Transpositionstabelle (TT)

key	Bewertung	α	β



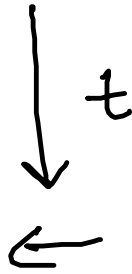
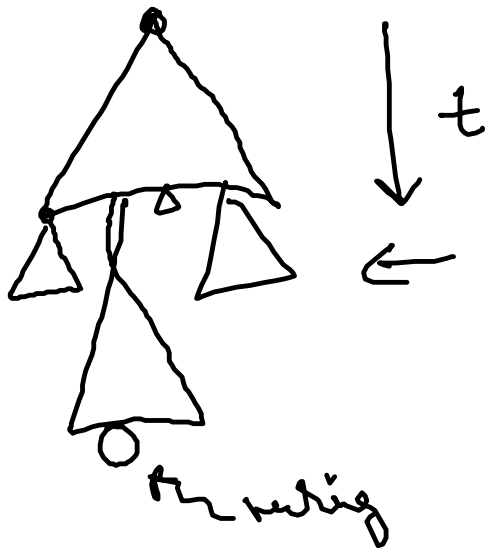
Aspiration Search

IDEA: Bewertung der aktuellen Stellung kann sich nach einem Zug nicht viel ändern

$\alpha\beta$ Suche (... , $\alpha = -\infty$, $\beta = +\infty$)

\Rightarrow α - β -Sude (\dots , $\alpha = b - \epsilon$, $\beta = b + \epsilon$)

o Rulesude



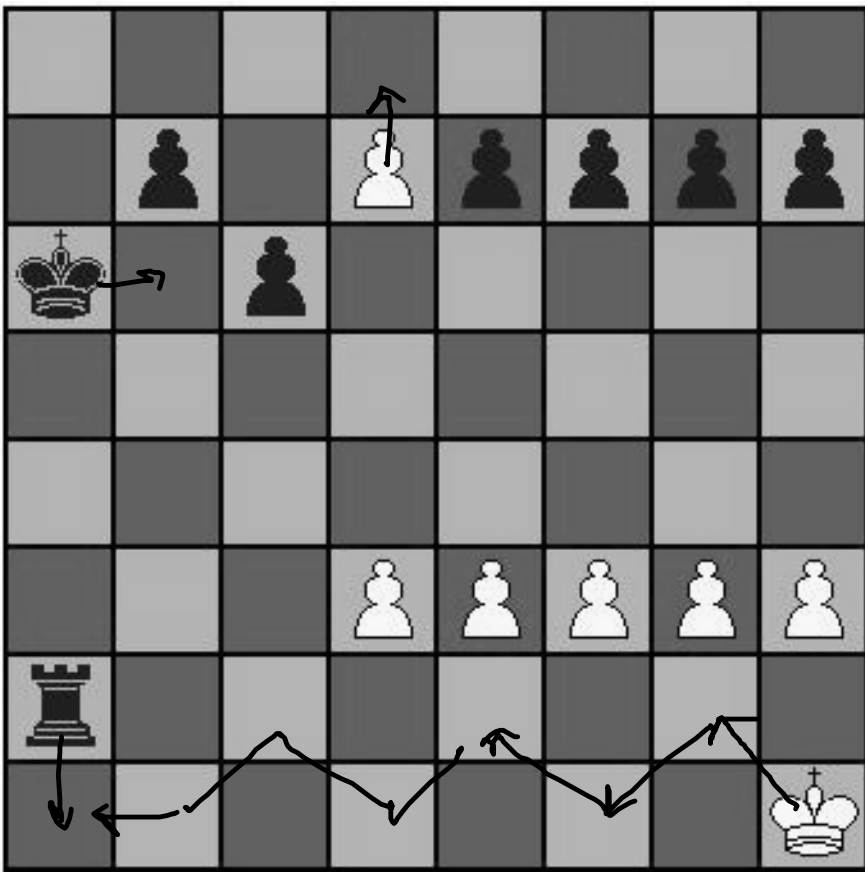
alle Schlag-, oder Schlagzüge
betrachtet

o Eröffnungsbücher

Endspielbücher



Interessanter Problem



-3



Top Programme:

16 Halbzüge