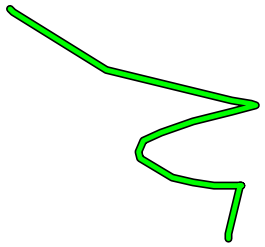
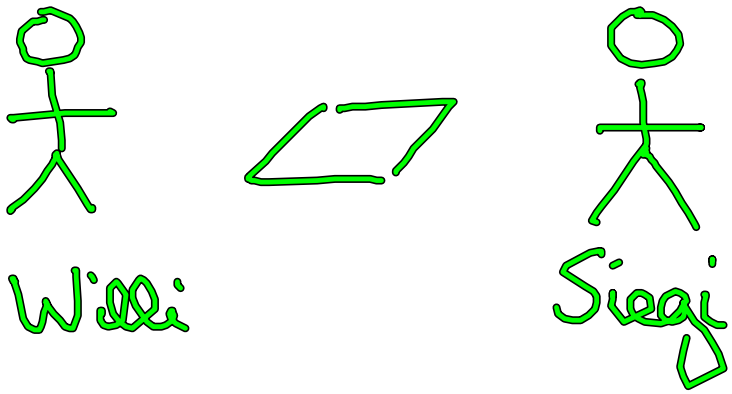


Spieltheorie – Optimale Spielstrategie

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



- 2-Spieler
- als wechselnd am Zug
- endliches Spiel (W, O, S)
- endlich viele Aktionen
- des einen Gewinn ist des anderen Verlust

⇒ Nullsummenspiel mit vollständiger Information (Perfect Knowledge Game)

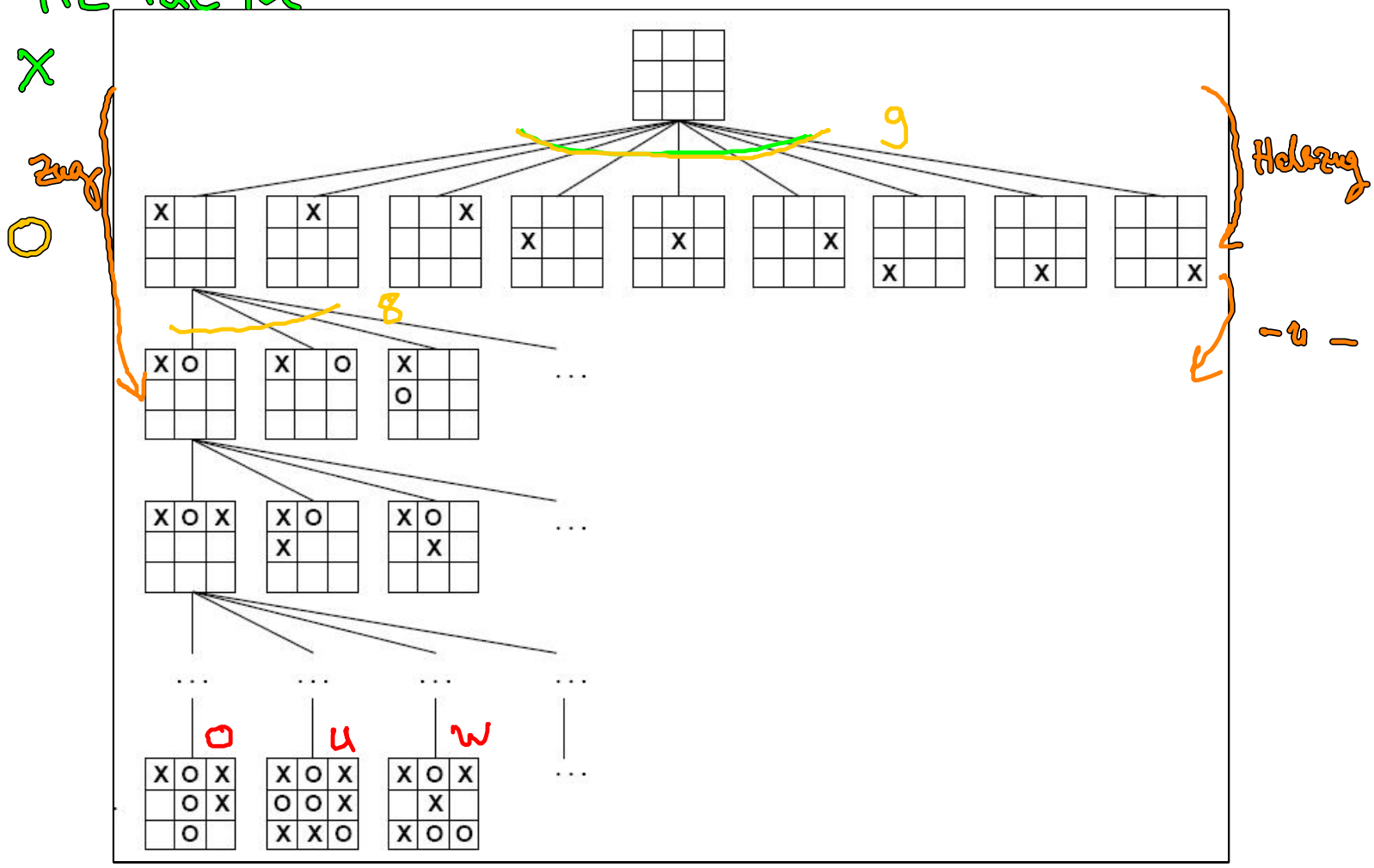
Bsp: TicTacToe, Schach, Go, Dame, ...

Herausforderung

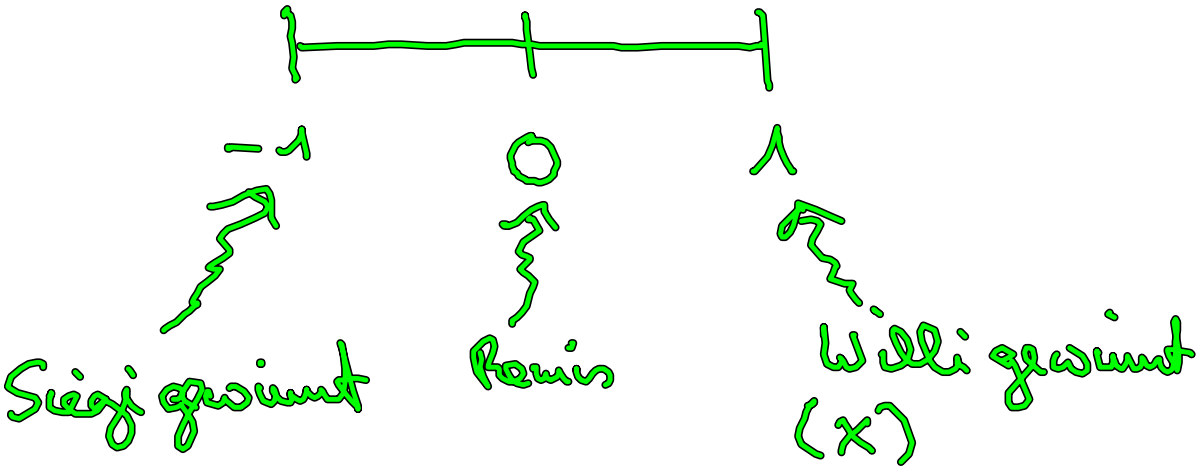
- Verhalten des Gegners ungewiss
- meistens kombinatorisch
- Mangel an Zeit und Speicher

Ermittlung der besten Spielstrategie

Tic Tac Toe :

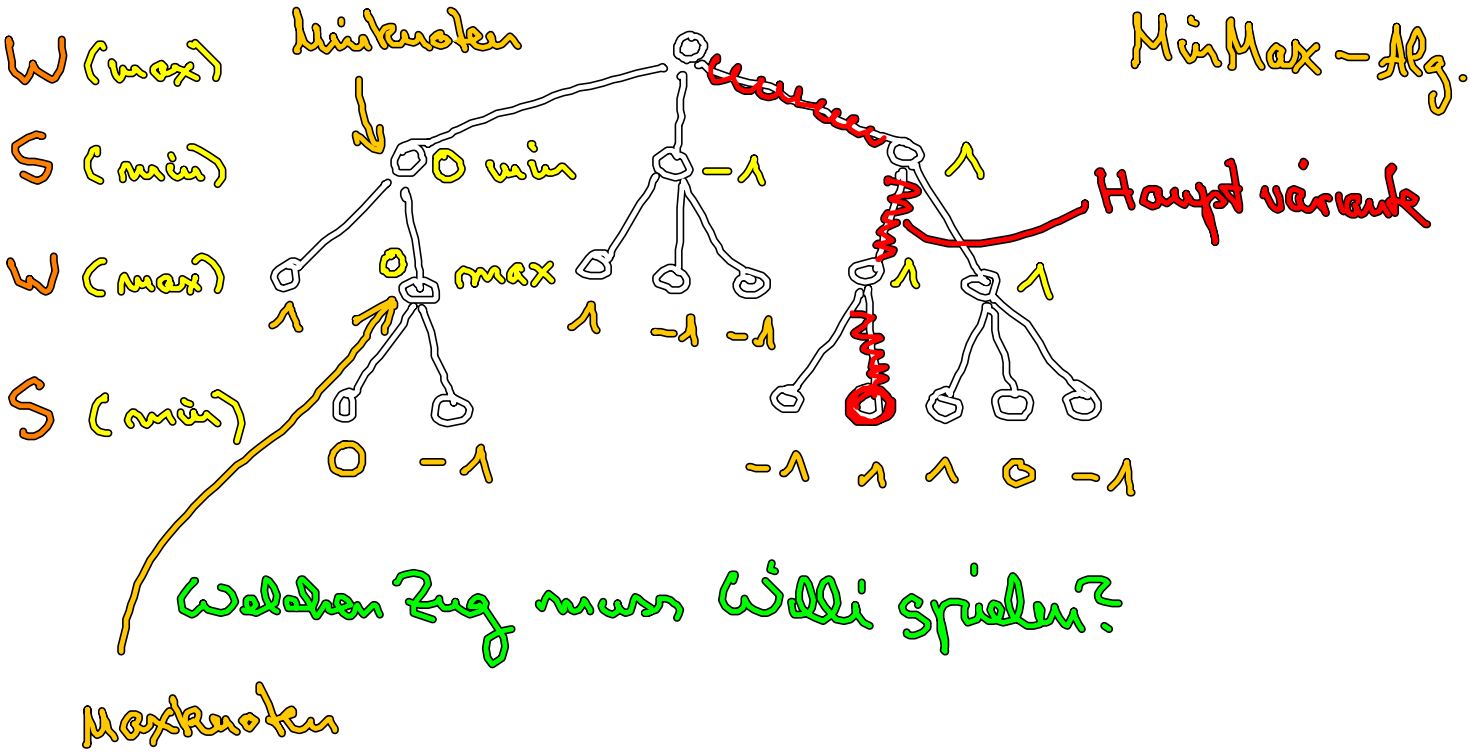


Definition



Hauptidee:

Gegner ist unfehlbar, spielt immer den besten Zug (optimale Spielweise).



MinMax-Prinzip:

n Zustand

S Menge der Nachfolger von n

$$minimax(n) = \begin{cases} Wert(n) & n \text{ ist term. Zst.} \end{cases}$$

$\max_{s \in S}$ minmax(s) ist Maxknoten

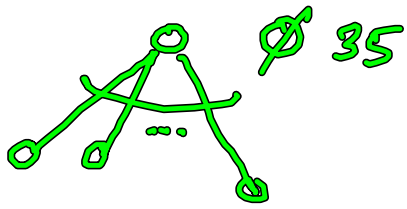
$\min_{s \in S}$ minmax(s) ist Minknoten

Erreichen wir immer die terminalen Zustände?

Wie groß ist der Suchraum?

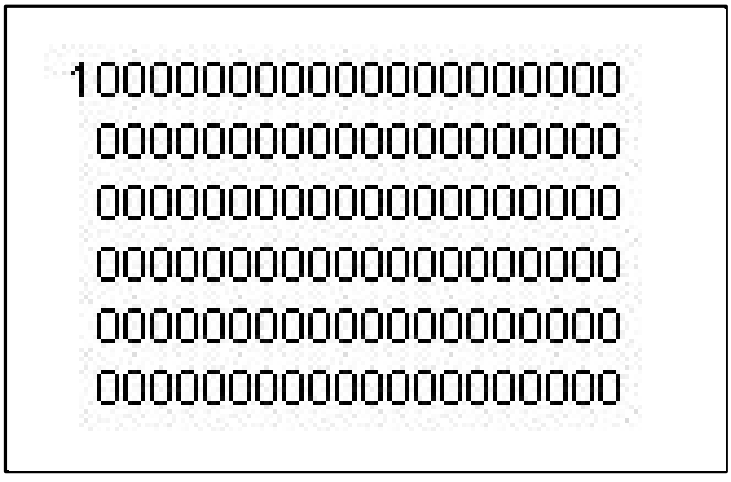
Suchraumprobleme

Schach



- es gibt schätzungsweise $2{,}28 \cdot 10^{46}$ legale Schachpositionen
- Partien → ca 50 Züge

⇒ $\sim 10^{120}$ verschiedene Partieverläufe

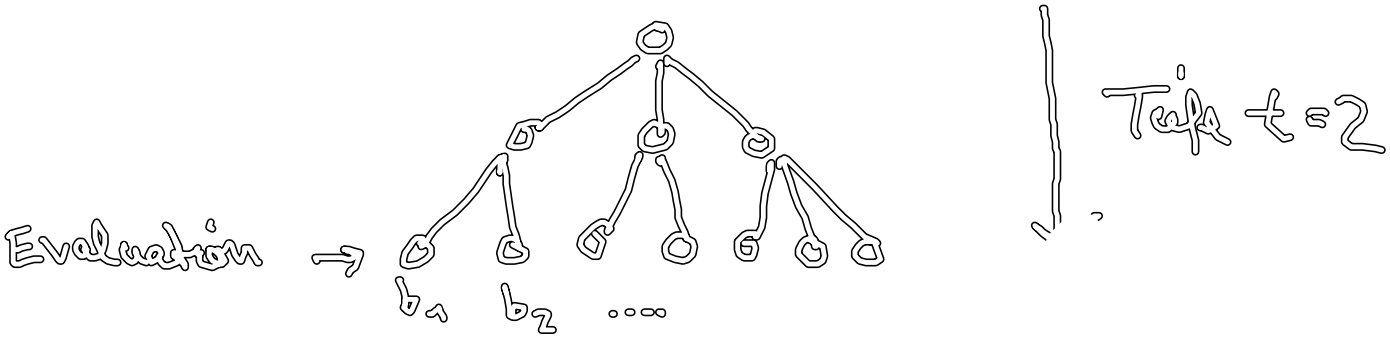


Atome im Weltall

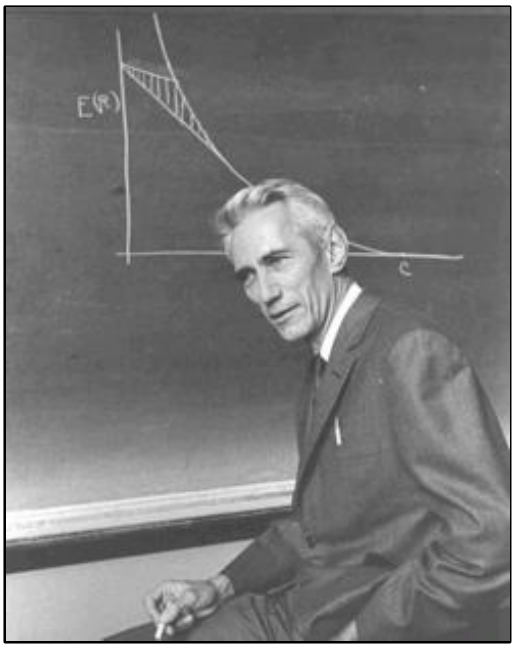
$\Rightarrow \sim 10^{80}$

IDEE: - feste Suchtiefe

- Abschätzungsfunktion



Claude Elwood Shannon



- Vortrag 1949
- Minmax + Evaluation

Bewertungsfunktion:

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

Achten auf:

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

meistens gewichtete, lineare Funktionen

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

... $f_1(s)$ = materialWeiss(s) − materialSchwarz(s)

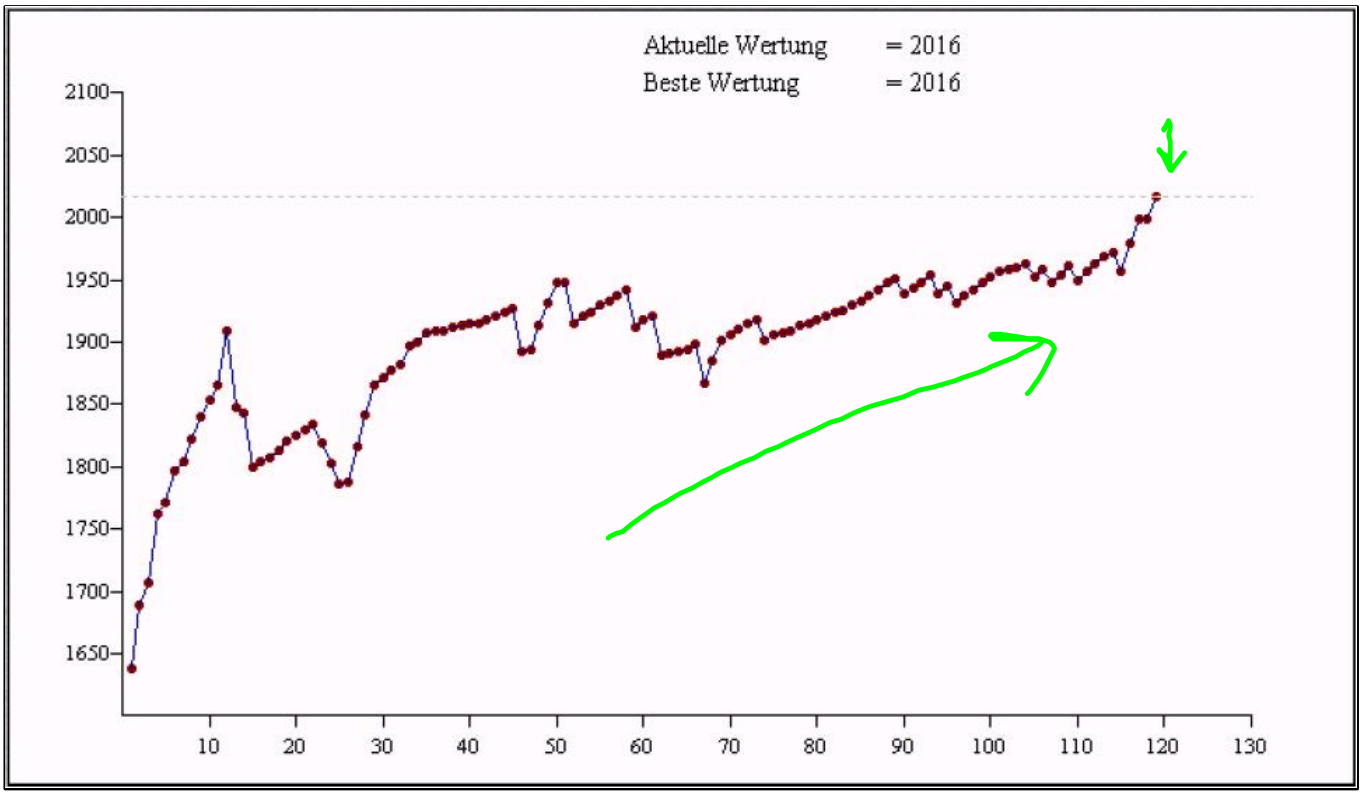
... $f_2(s)$ = mobilitätW(s) − mobilitätS(s)

$\vec{a}_1$ Eröffnung $\vec{a}_2$ Mittelspiel $\vec{a}_3$ Endspiel

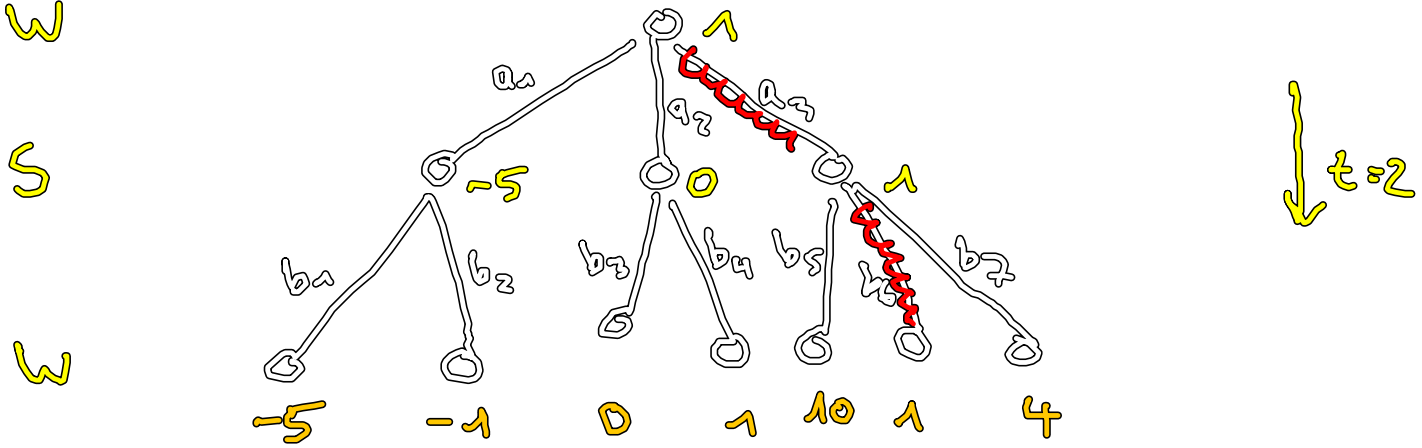
- König Zentrum
- Möglichst Dame

$\vec{a}_1 \dots \vec{a}_n$

Reinforcement Learning



MinMax-Algorithmus mit Bewertungsfunktion



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

Pseudocode

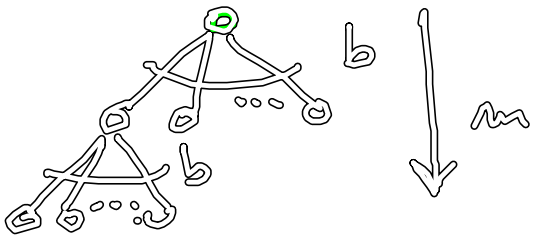
Effizienz

```
maxKnoten ( Knoten x) → Integer {
    if (x ist Blatt)
        return evaluation (x)
    else {
        // x_1 ... x_k seien Kinder von x
        ω := -∞
        for i := 1 to k do {
            v := minKnoten (x_i)
            if (v > ω)
                ω := v
        }
        return ω
    }
}
```

⇒ minimaxen entsprechend

Aufwandsabschätzung

Sei m maximale Spielbaumtiefe und
b die durchschnittliche Anzahl der Spielzüge



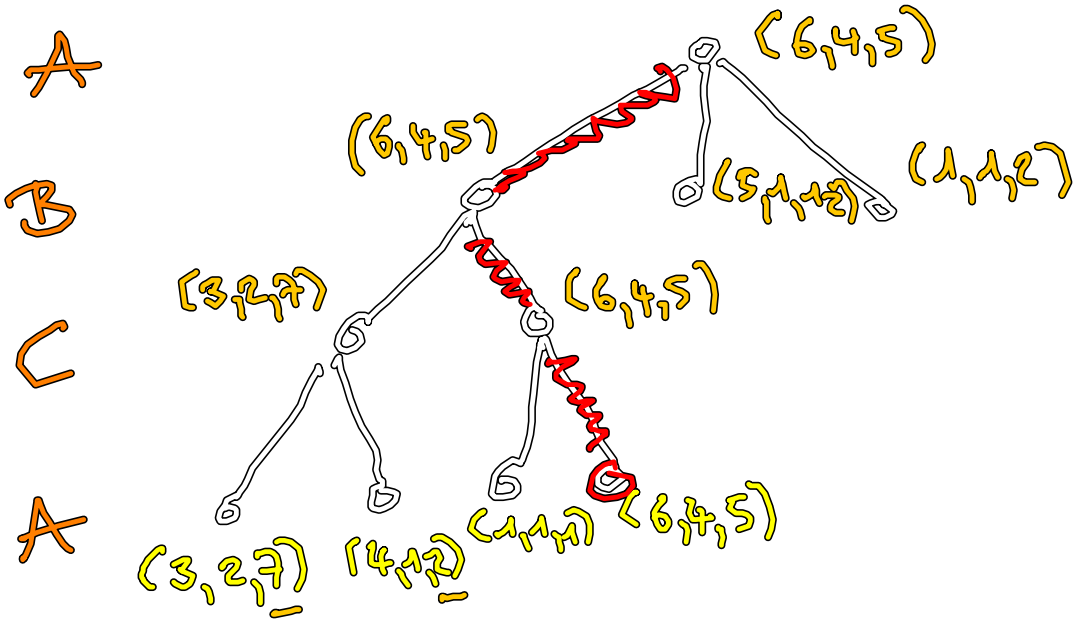
$\Rightarrow O(b^m)$

MinMax in Prolog

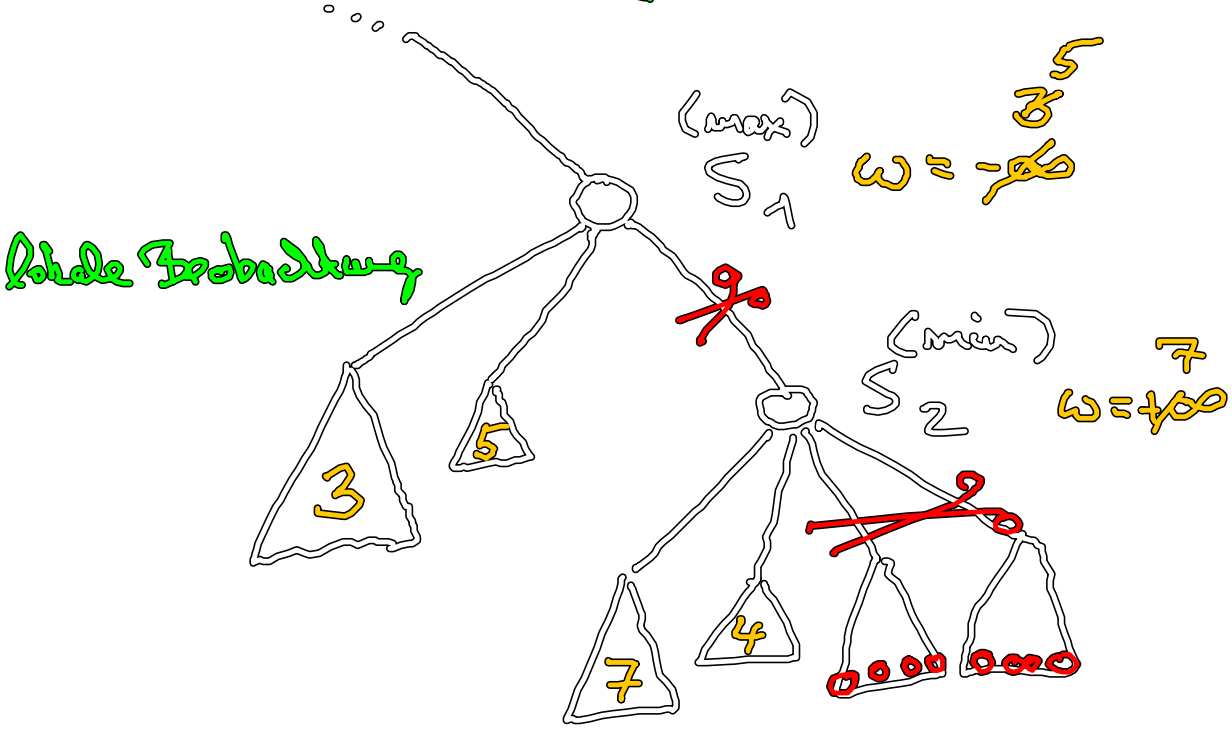
```
minmax(Pos, BestSucc, Val) :-
    (moves(Pos, PosList), !,
      best(PosList, BestSucc, Val))
    ;
    evaluation(Pos, Val).

... (Tut)
```

Ausblick

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

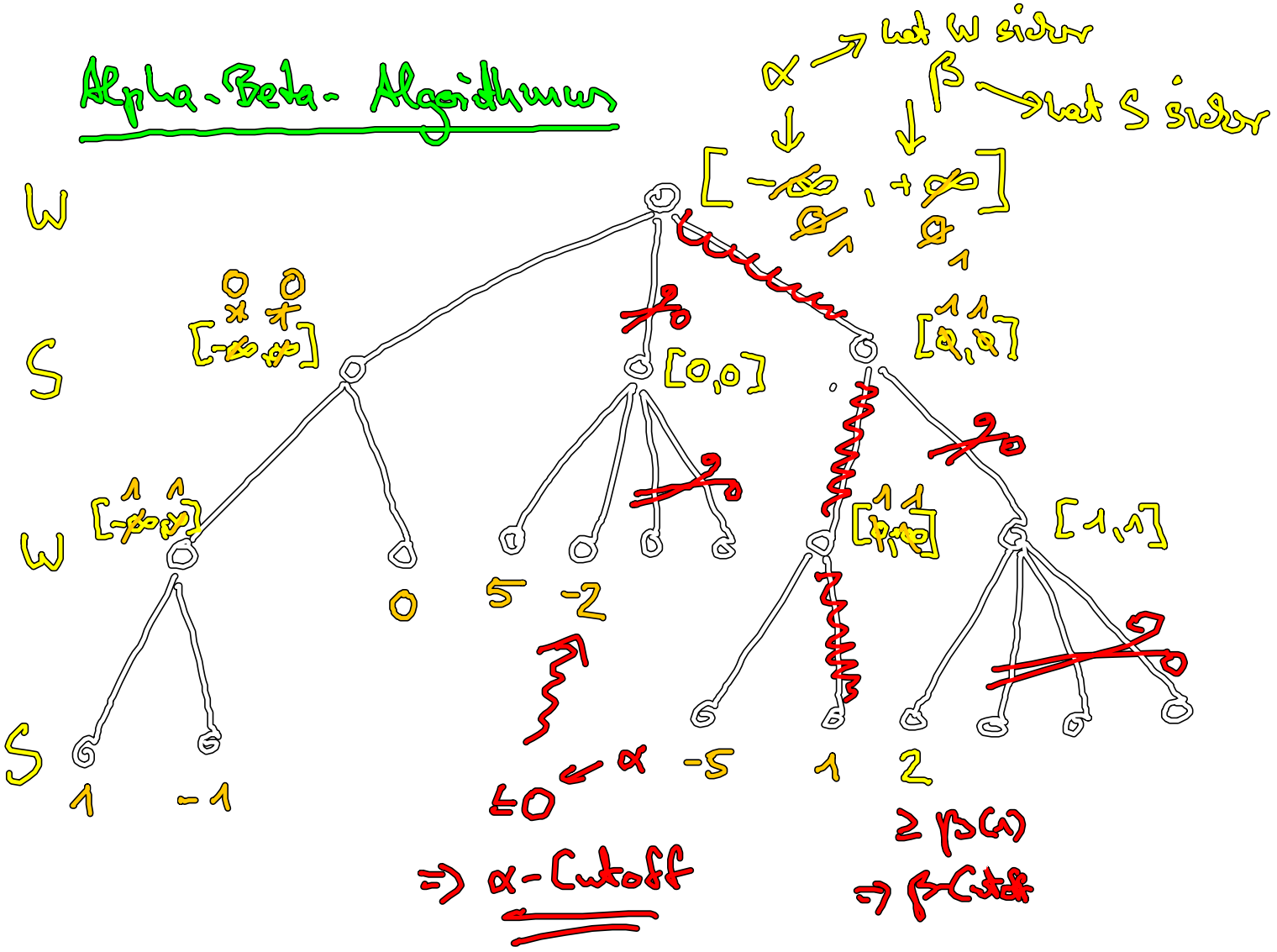


Alpha-Beta-Pruning



Newell, Shaw, Simon 1958

Alpha-Beta-Algorithmus



Pseudocode:

```
αβ maxKnoten(Knoten x, Integer α, β) -> Int {
    if (x istBlatt)
        return evaluation(x)
    else {
        ω := α
```

```
for i:= 1 to k do {
    v := αβmin Knoten(x_i, ω, β)
    if (v > ω)
        ω := v
    if (ω ≥ β)          //β-Cutoff
        return β
}
return ω
}
}
```

⇒ αβ min Knoten entsprechend

Aufwandsabschätzung

worst case → MinMax $O(b^m)$

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

⇒ Verzweigungsfaktor $\sqrt{b}$ statt b

Schach ⇒ 35 $\underline{6}$

zufällig → $O((\frac{b}{\log b})^n)$

Optimierungen + Heuristiken

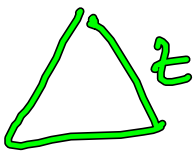
- Zugsortierungen

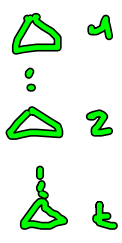
  IDEE: Züge, die sich als gut bewährt haben sollen Vorzug erhalten

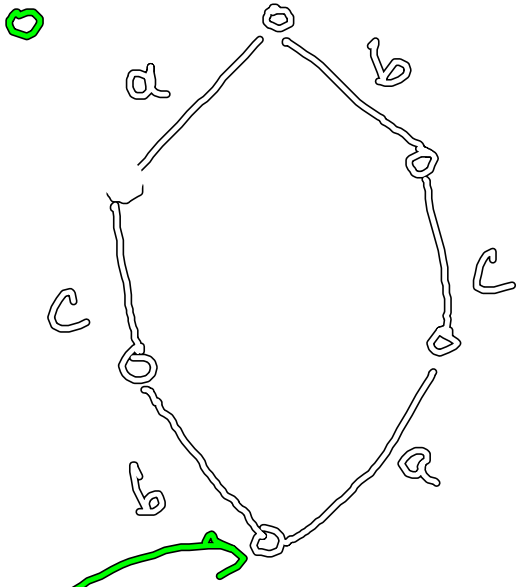
  - Killermoves
  - Historymoves
  - Hauptvariante

- Iterative Suchverfahren

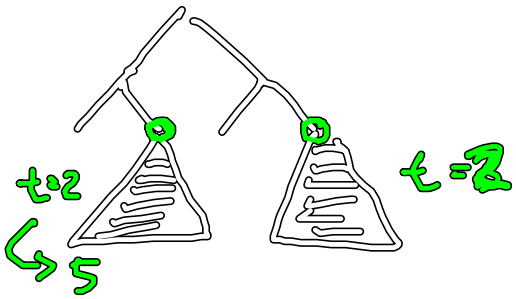
  sehr gut





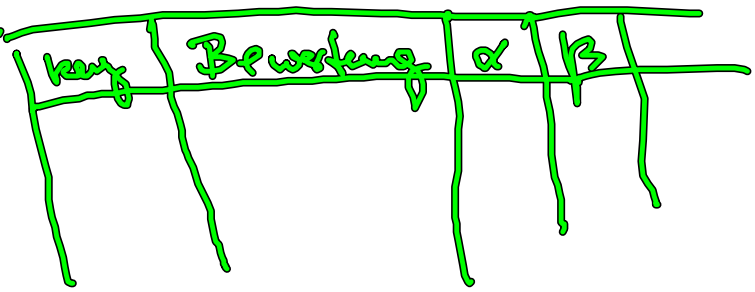


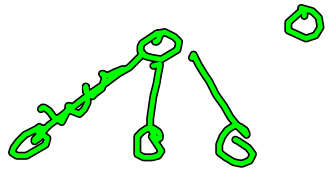
Subgraphen! (keine Bäume)

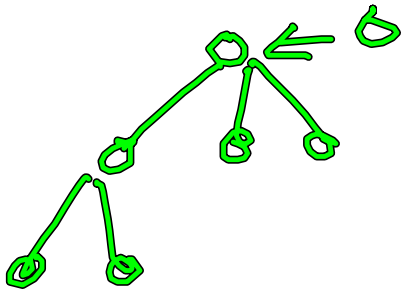


key(S)

Transpositionstabellen (TT)







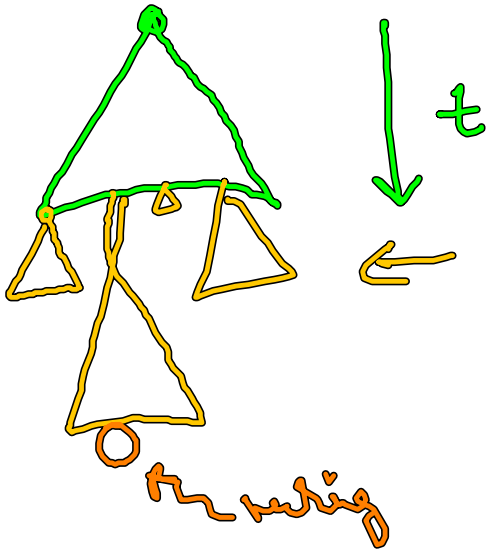
• Aspiration Search

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

αβsuche (..., α=-∞, β=+∞)

$\Rightarrow$ alphabeta(..., $\alpha = b - \varepsilon$, $\beta = b + \varepsilon$)

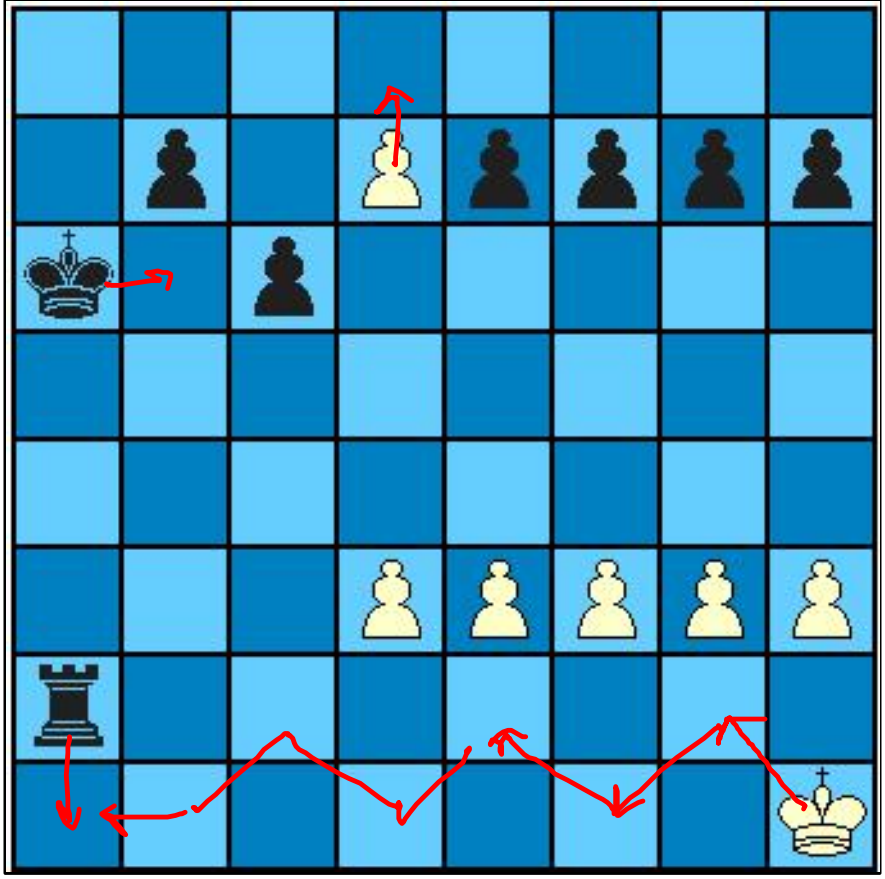
- Ruhesuche



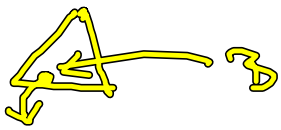
- Eröffnungsbücher

Endspielbücher

Interessanter Problem



-3



Top Programme:

16 Halberuga