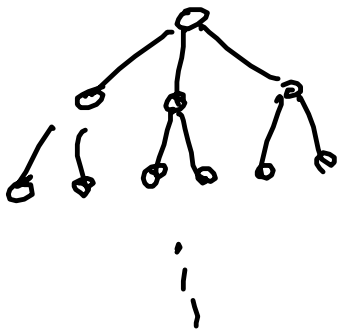
Suchraumproblematik

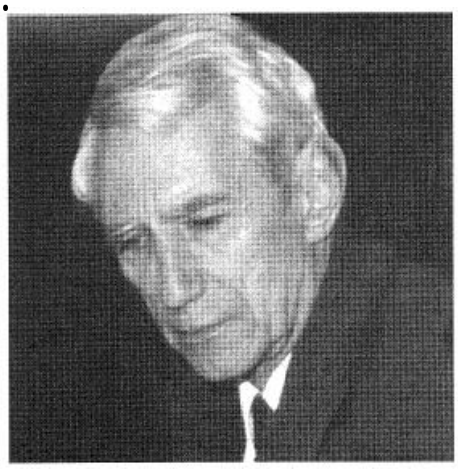
Schach



Halbzugtiefe	#Stellungen
1	40
2	1600
⋮	
7	164 Milliarden
⋮	
11	419 Billionen

Verzweigungsfaktor ca. 33–40

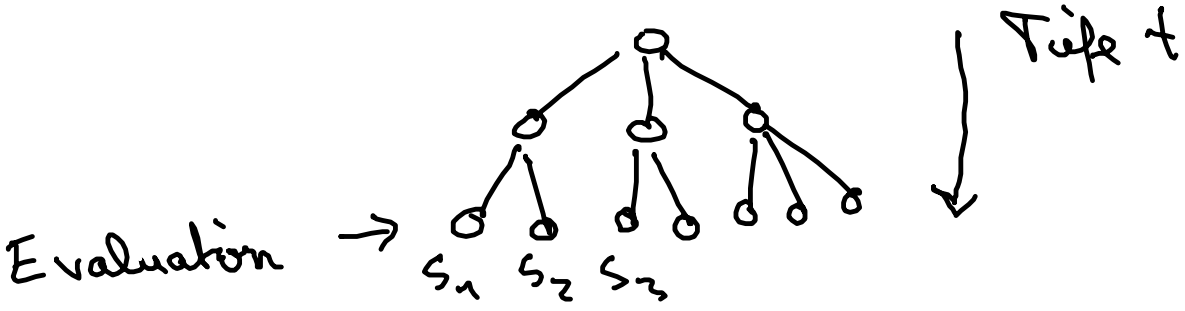
Claude Elwood Shannon



- Mathematiker, New Jersey
- Vortrag, 9. März 49
  → Min Max + Evaluation
- Informationstheorie

IDEE

- feste Suchtiefe
- Abschätzung

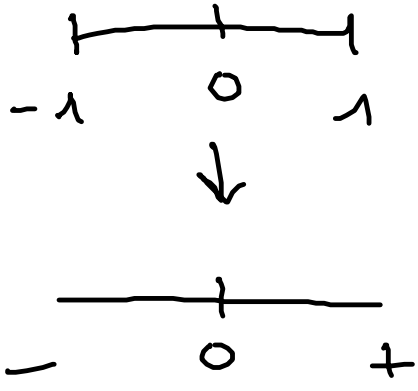


Bewertungsfunktion

meistens linear

S zu beurteilende Stellung

$f : S \to \mathbb{R}$



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

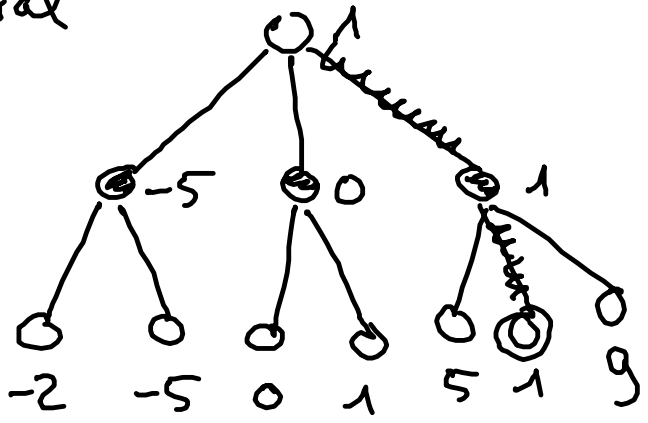
Spiel-wissen

$f_1(s) =$ material (W_s) − material (S_s)

$f_2(s) =$ mobilität (W_s) − material (S_s)

$\vdots$

Min Max + Eval



PseudoCode

```
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) then ω := v
        }
        return ω
    }
}

minKnoten(Knoten x) -> Integer
    if (x Blatt)
        return evaluation(x)
    else {
        ω := +∞
        for i := 1 to k do {
            v := maxKnoten(x_i)
            if (v < ω) then ω := v
        }
        return ω
    }
```

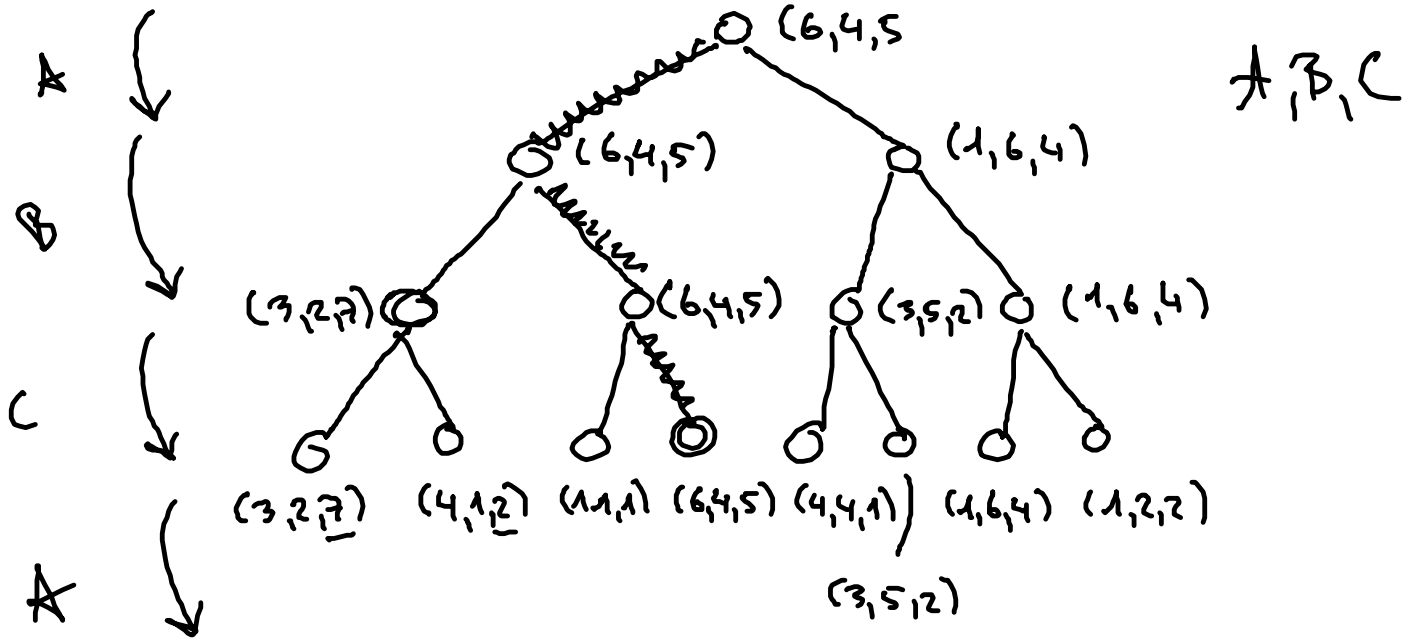
Min Max in Prolog

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

best([Pos], Pos, Val) :-
    minmax(Pos, _, Val).

best([Pos1 | PosList], BestPos, BestVal) :-
    minmax(Pos1, _, Val1),
    best(PosList, Pos2, Val2),
    betterof(Pos1, Val1, Pos2, Val2,
                BestPos, BestVal).

betterof(Pos0, Val0, Pos1, Val1, Pos0, Val0) :-
    min_to_move(Pos0),
    Val0 > Val1, !
    ;
    max_to_move(Pos0),
    Val0 < Val1, !.
```

betterof(Pos0, Val0, Pos1, Val1, Pos1, Val1).

Ausblick

Minmax bei MehrSpielerSpielen



(X, Y, Z)

↑ ↑ ↑

A B C

z.B. Siedler

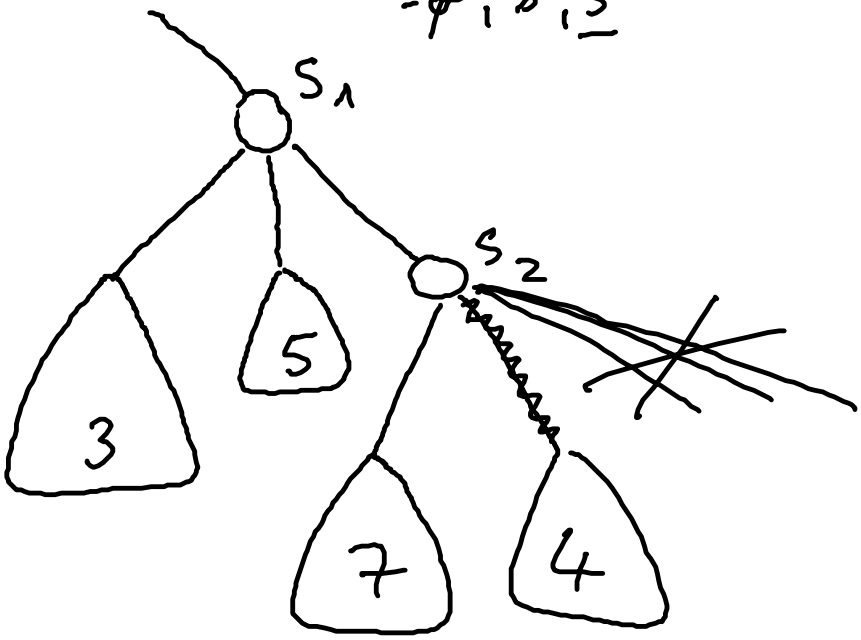
→ Allianzen (Zusammenschlüsse)

Alan Newell, John Shaw und Herbert Simon

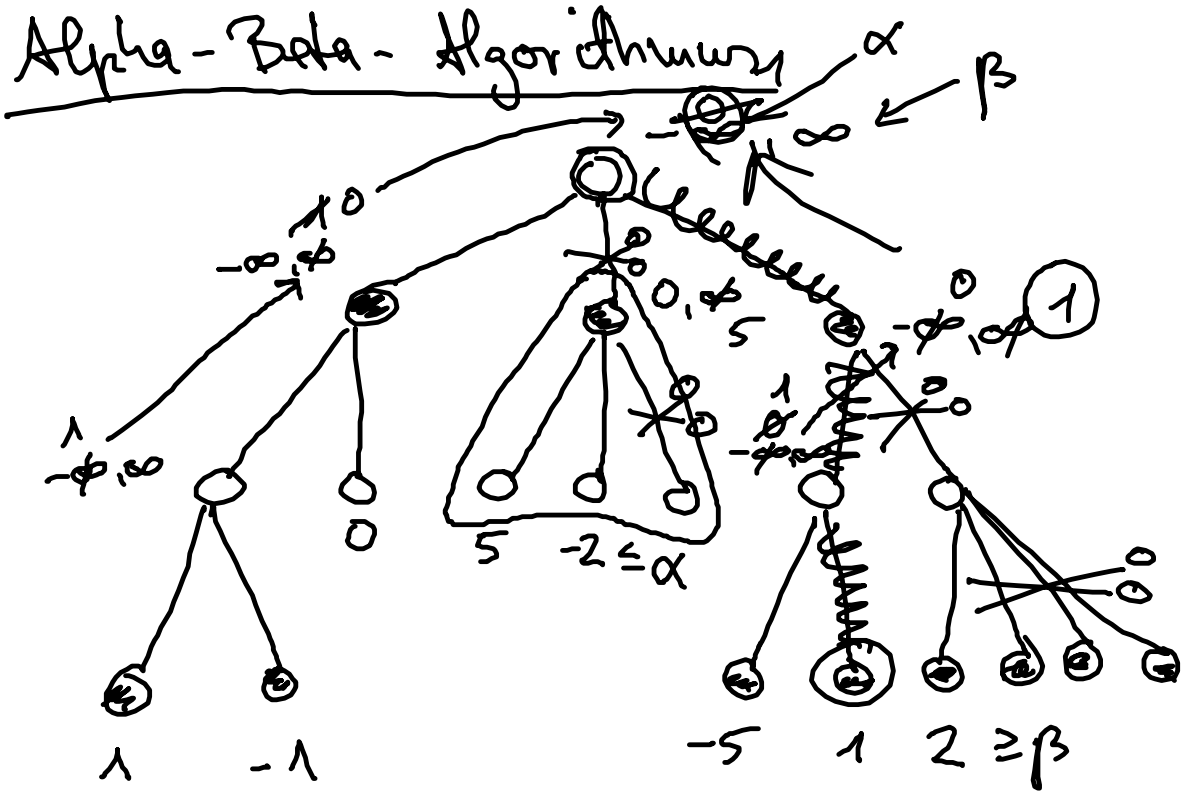
- „Computerwissenschaftler", Pittsburgh
- 1958
  → Alpha-Beta

lokale Betrachtung

$-\phi, 3, \underline{5}$



Alpha-Beta-Algorithmus



Pseudocode

```
αβmaxKnoten(Knoten x, Integer α,β) → Integer
    if (x ist Blatt)
        return evaluation(x)
    else {
        ω := α
        for i:= 1 to k do {
            v := αβminKnoten(x_i, ω, β)
            if (v > ω) then ω := v
            if (ω ≥ β) then return ω //β-Cutoff
        }
        return ω
```

```
}

αβminKnoten(Knoten x, Integer α, β) → Integer
  if (x ist Blatt)
      return evaluation(x)
  else {
      w := β
      for i := 1 to k do {
          v := αβmaxKnoten(x_i, α, w)
          if (v < w) then w := v
          if (w ≤ α) then return w
                          // α-Cutoff
      }
      return w
}
```

Effektivität von Alpha-Beta

im schlechtesten Fall wie Min Max

Zugreihenfolge

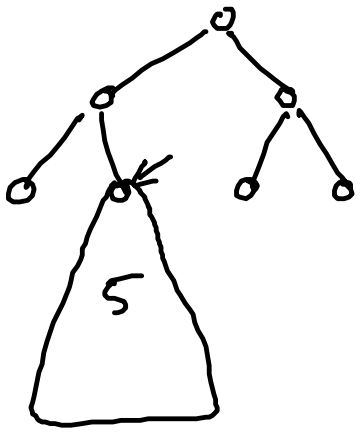
optimal

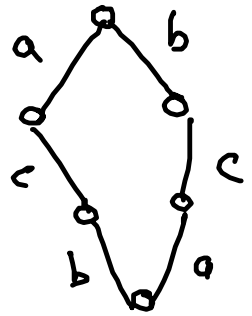
Bsp Schach Verzweigungsgrad (branchfaktor b)

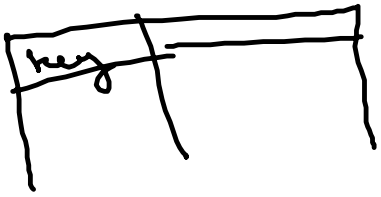
Tb

Optimierung

- Transpositionstabellen

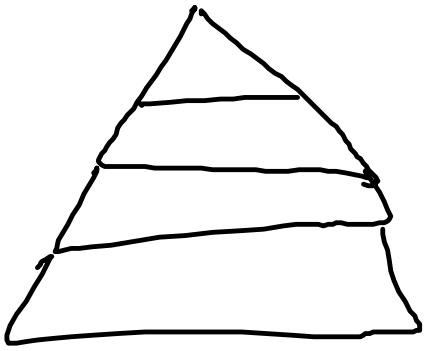






- Zugsortierung

  Schach: Killermoves
  Historymove
  Hauptvariante



↓ 2, 3, 4

iterativen Suchverfahren