INF FAKULTÄT FÜR INFORMATIK

Intelligente Systeme

Spiele

Prof. Dr. R. Kruse **C. Braune**

{rudolf.kruse,christian,braune}@ovgu.de

Institut für Intelligente Kooperierende Systeme

Fakultät für Informatik

Otto-von-Guericke Universität Magdeburg

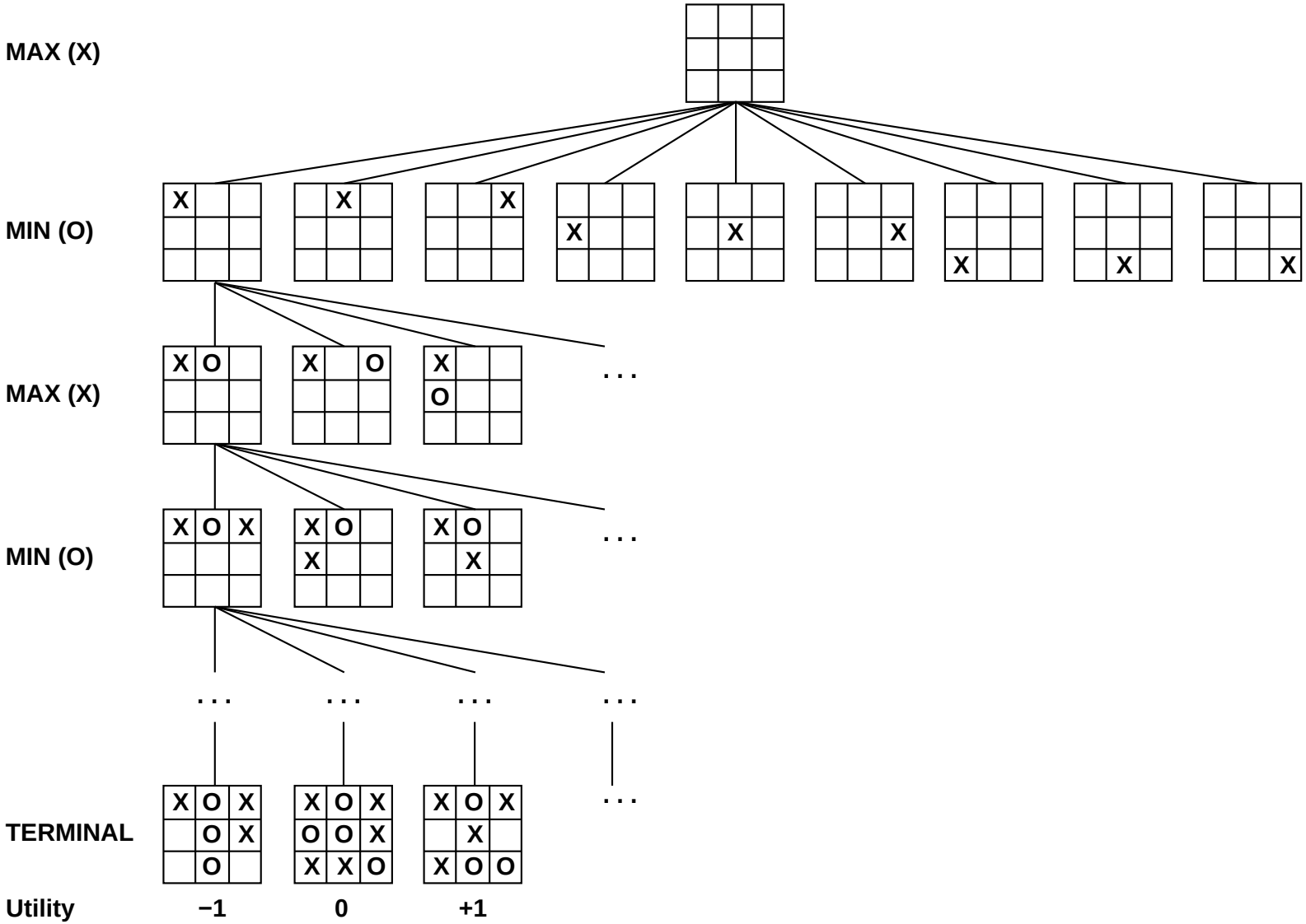
Arten von Spielen

	deterministisch	**Glücksspiel**
vollständige Informationen	Schach, Dame, Go, Othello	Backgammon, Monopoly
unvollständige Informationen	Schiffe versenken, blindes Tic-Tac-Toe	Bridge, Poker, Scrabble, Nuklearer Krieg

Blindes Tic-Tac-Toe:

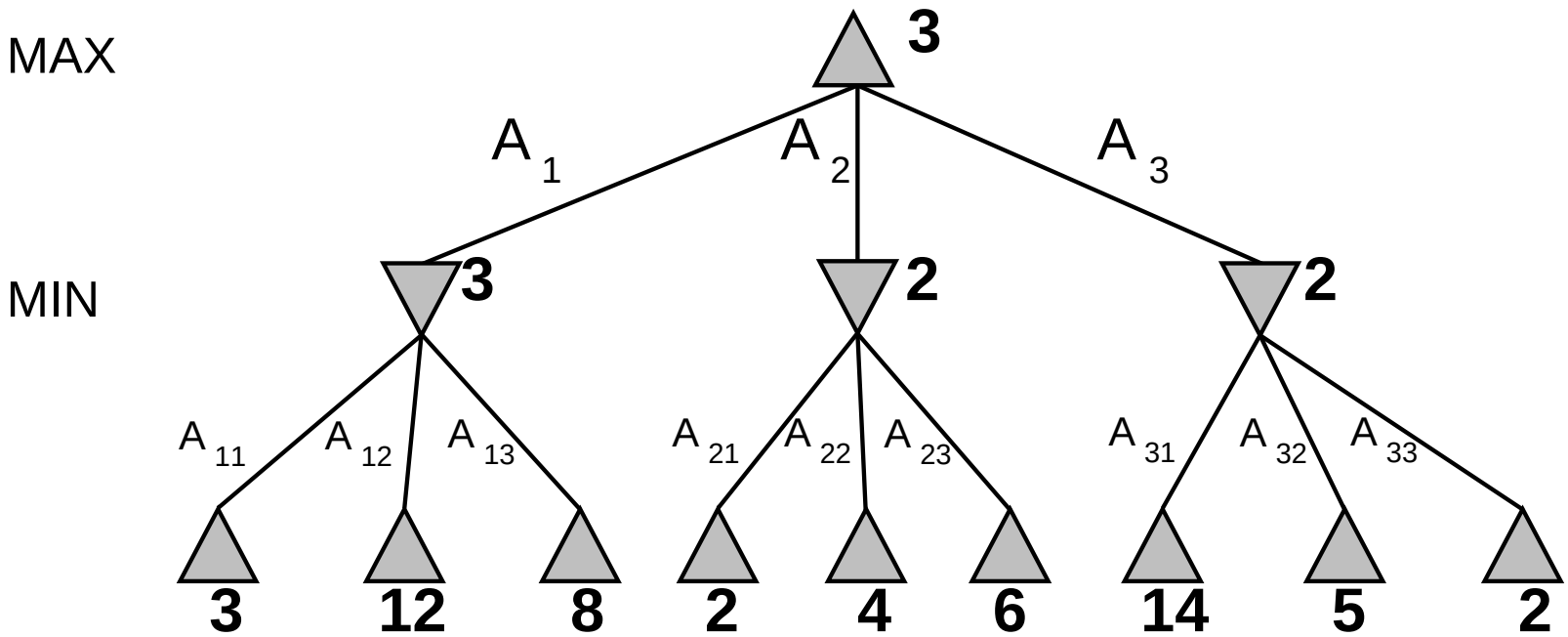
- Unvollständige Variante des Standardspiels
- Jeder Spieler kann X *und* O setzen
- Gegner erfährt nur welches Feld, aber nicht ob X oder O
- Spieler mit erster Linie mit 3 gleichen Zeichen gewinnt

Spielbaum bei Tic-Tac-Toe



Minimax

Minimax-Algorithmus ist ein Algorithmus zur Ermittlung der optimalen Spielstrategie für endliche Zwei-Personen-Nullsummenspiele mit perfekter Information.



Minimax-Algorithmus

MINIMAX-DECISION

Eingabe: *state*, momentaner Zustand im Spiel
Ausgabe: eine Aktion *action*

1: **return** die Aktion a in ACTIONS(*state*), die MIN-VALUE(RESULT(a, *state*)) maximiert

MAX-VALUE

```
1: if TERMINAL-TEST(state) {
2:     return UTILITY(state)
3: }
4: v ← −∞
5: for each a, s in SUCCESSORS(state) {
6:     v ←MAX(v,MIN-VALUE(s))
7: }
8: return v
```

MIN-VALUE

```
1: if TERMINAL-TEST(state) {
2:     return UTILITY(state)
3: }
4: v ← ∞
5: for each a, s in SUCCESSORS(state) {
6:     v ←MIN(v,MAX-VALUE(s))
7: }
8: return v
```

Eigenschaften des MiniMax-Algorithmus

Zeit- und Speicherkomplexität gemessen anhand von

b: maximalem Verzweigungsfaktor des Suchbaums
m: maximaler Tiefe des Zustandsraums (eventuell ∞)

Vollständig: ja, falls Baum endlich

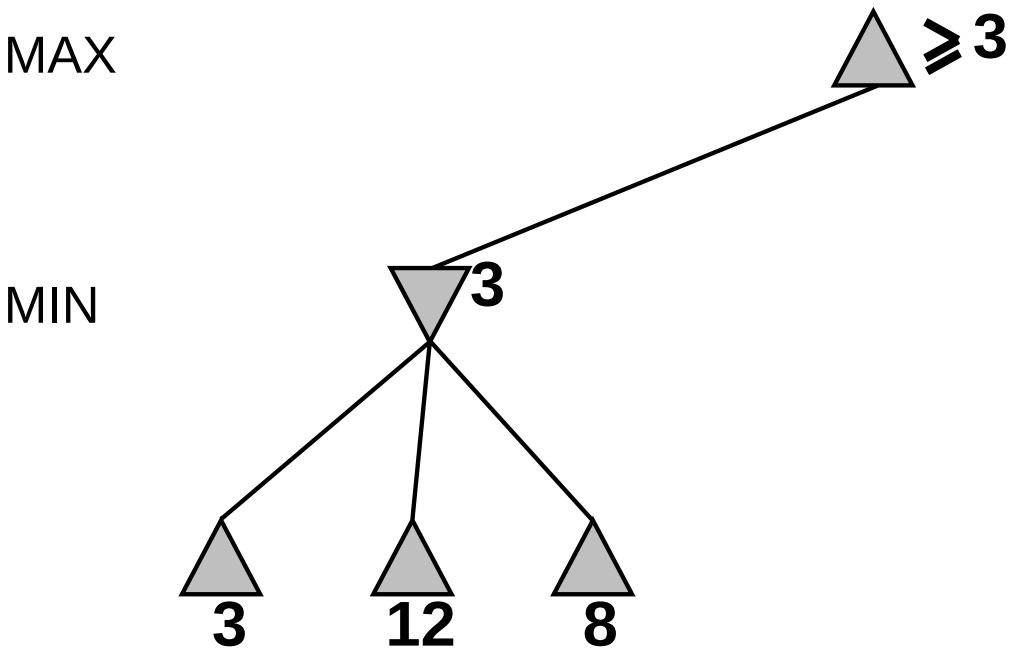
Optimal: ja, gegen optimalen Gegner

Zeit: $O(b^m)$

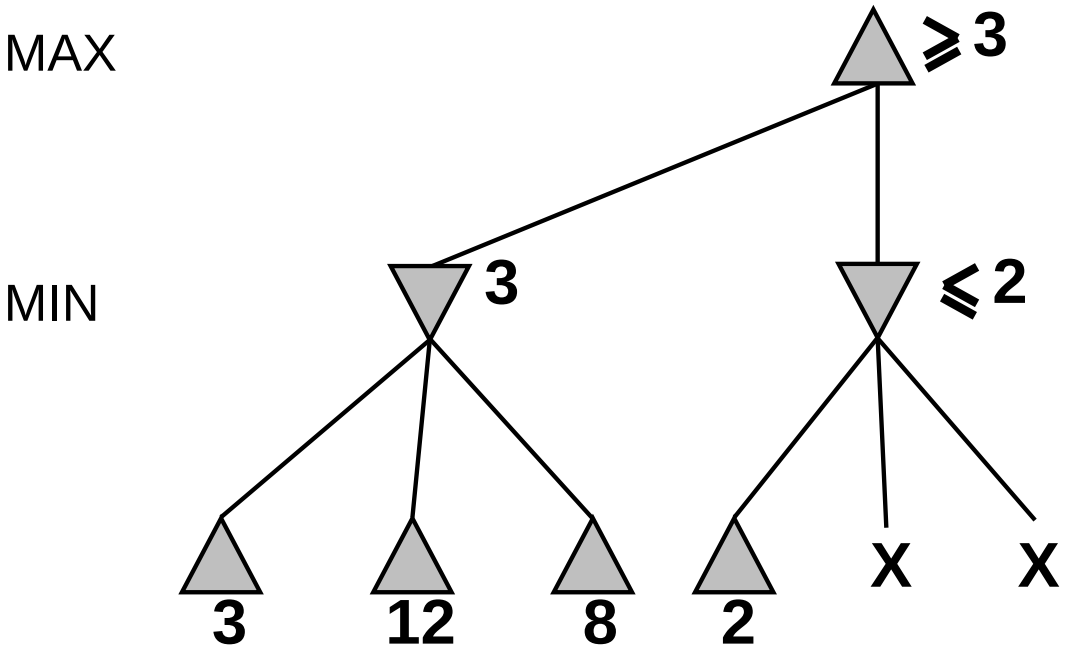
Speicher: $O(b \cdot m)$ (Tiefensuche)

Für Schach: $b \approx 35, m \approx 100$ bei „realistischen“ Spielen.
Somit ist die exakte Lösung absolut nicht berechenbar.
Aber: muss jeder Pfad exploriert werden?

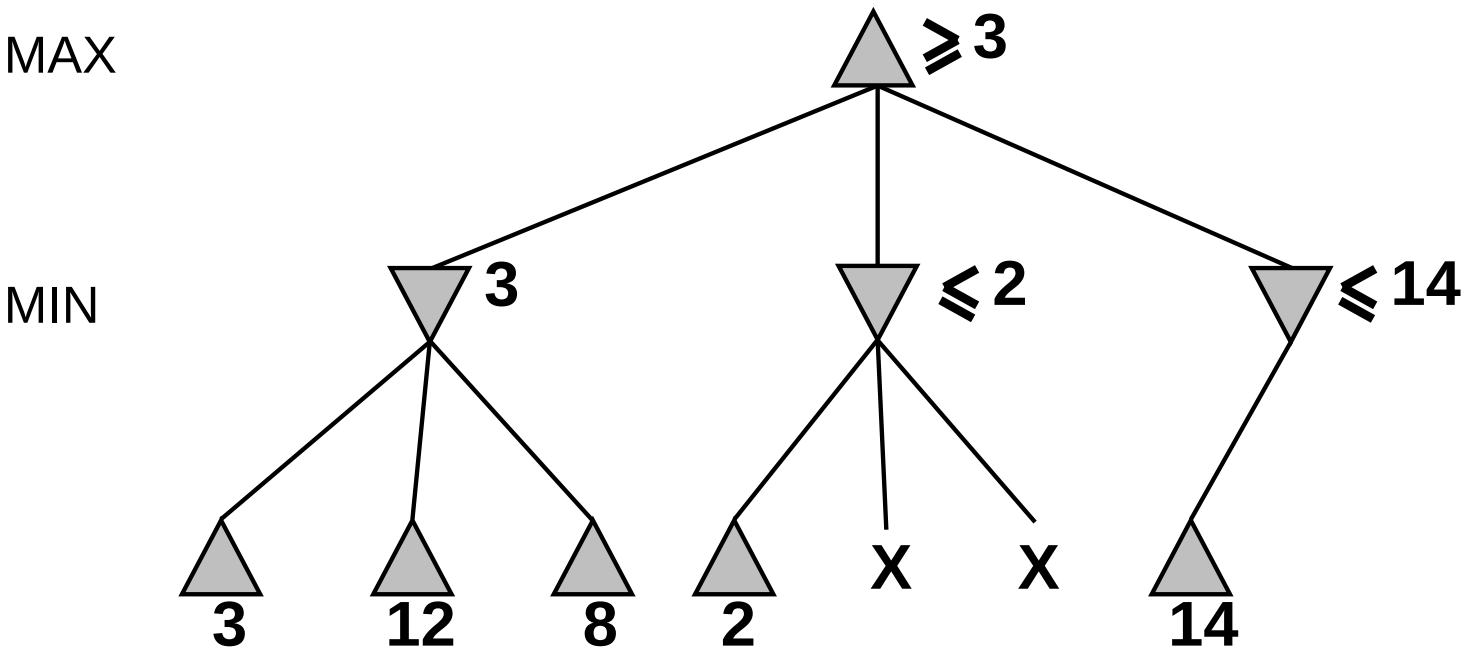
α–β-Stutzen



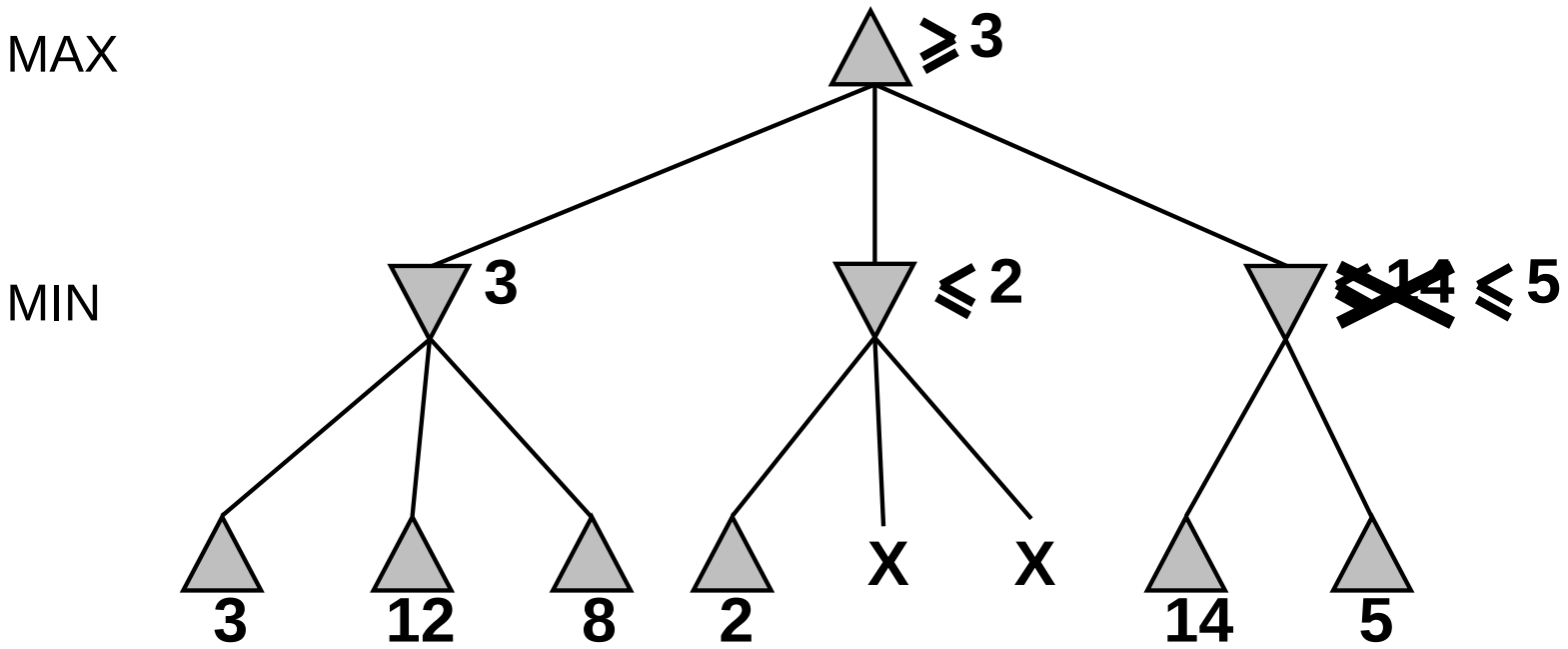
α–β-Stutzen



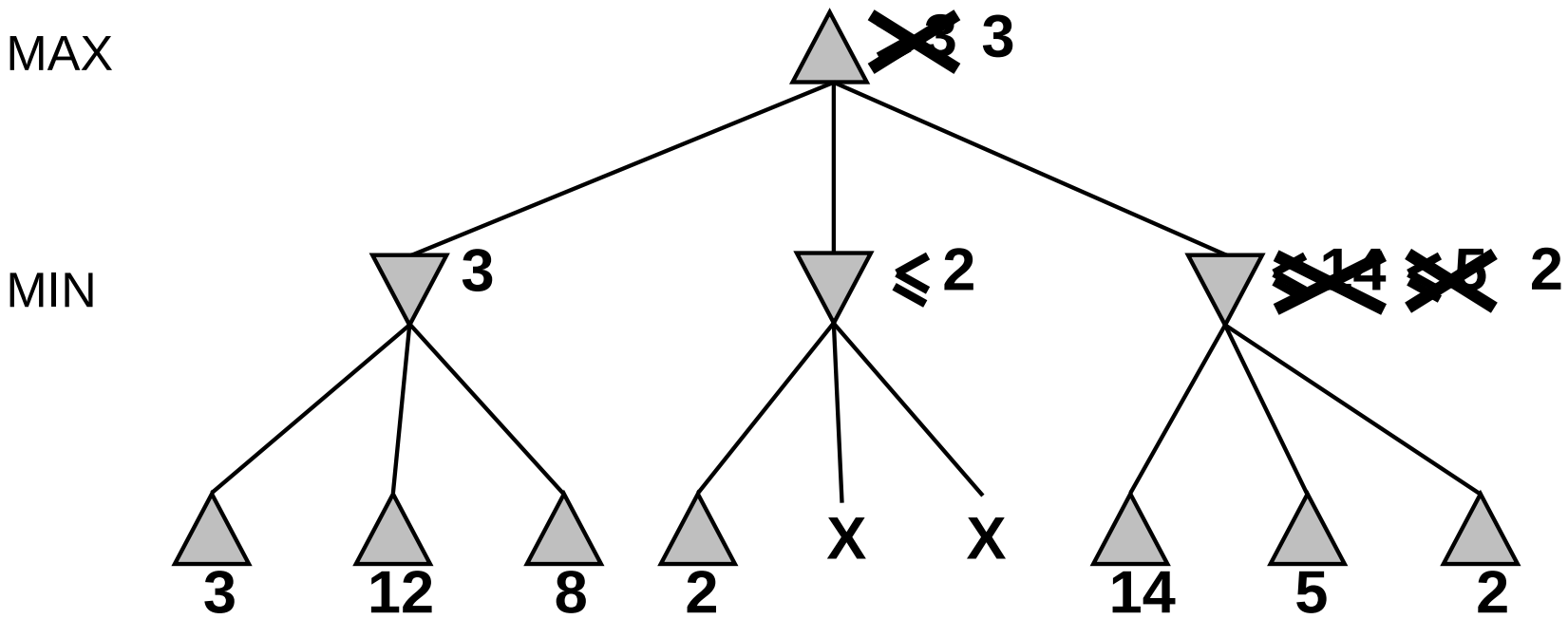
α–β-Stutzen



α–β-Stutzen



α–β-Stutzen



Der α–β-Algorithmus

ALPHA-BETA-DECISION

1: **return** die Aktion *a* in ACTIONS(*state*), die MIN-VALUE(RESULT(*a*, *state*)) maximiert

MAX-VALUE

Eingabe: *state*, momentaner Zustand im Spiel
α, bester MAX-Wert auf Pfad zu *state*
β, bester MIN-Wert auf Pfad zu *state*

```
1: if TERMINAL-TEST(state) {
2:     return UTILITY(state)
3: }
4: v ← −∞
5: for each a, s in SUCCESSORS(state) {
6:     v ←MAX(v,MIN-VALUE(s, α, β))
7:     if v ≥ β {
8:         return v
9:     }
10:    α ← MAX(α, v)
11: }
12: return v
```

MIN-VALUE

1: genau wie MIN-VALUE aber mit vertauschten Rollen von α, β

α–β-Algorithmus: Eigenschaften

Stutzen hat **keine** Auswirkung auf Endergebnis

- Gute Zugordnung verbessert Effektivität des Stutzens.
- Mit „perfekter“ Ordnung, Zeitkomplexität $= O(b^{m/2})$
- Somit doppelte Suchtiefe
- Für Schach: immer noch 35^{50} möglich

α–β-Algorithmus: Grenzen

Standard-Ansatz:

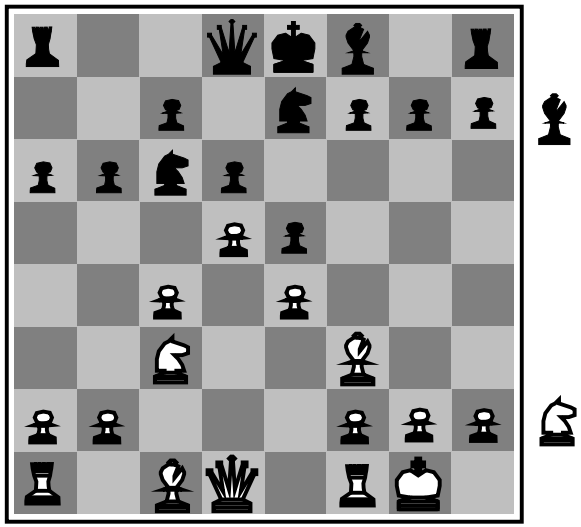
Für gewöhnlich: Lösung zu tief im Suchbaum

Utility mit Werten +1, 0 oder -1 nicht berechenbar

Nutze Cutoff-Test anstelle von Terminal-Test
z.B. Tiefenbegrenzung (u.U. mit „Ruhesuche“ – verfolgt aktive Pfade (z.B. nachdem Figur im Schach geschlagen wurde) tiefer als inaktive)

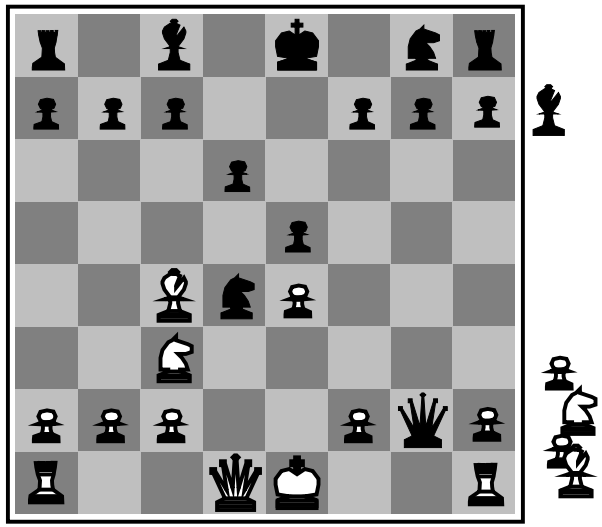
Nutze Eval (Güteschätzung des Zustands) anstatt Utility:
Bewertungsfunktion schätzt Güte einer Stellung

Bewertungsfunktionen



Black to move

White slightly better



White to move

Black winning

Für Schach: typischerweise linear gewichtete Summe von n **Merkmalen**

$$\text{Eval}(s) = \sum_{i=1}^{n} w_i \cdot f_i(s)$$

Deterministische Spiele

Dame:

- 1994 beendete Chinook die 40-jährige Herrschaft des Weltmeisters Marion Tinsley
- Endspieldatenbank mit perfekten Spielen aller Stellungen mit ≤ 8 Steinen ($\geq 443 \cdot 10^9$ Stellungen)

Schach:

- Deep Blue besiegte 1997 Weltmeister Garri Kasparow in 6 Spielen
- $200 \cdot 10^6$ Stellungen/Sekunde, sehr komplizierte Bewertung
- Bis zu 40 Halbzüge tief (nicht veröffentlichte Methoden)

Go:

- Wettbewerb bis 2016 nur unter menschlichen Meistern
- $b > 300$, daher Datenbanken mit Mustern für plausible Züge
- Beste Spieler weltweit werden inzwischen von AlphaGo geschlagen

AlphaGo: Problemstellung Go

2 Spieler (Schwarz, Weiß)

Legen abwechselnd Steine

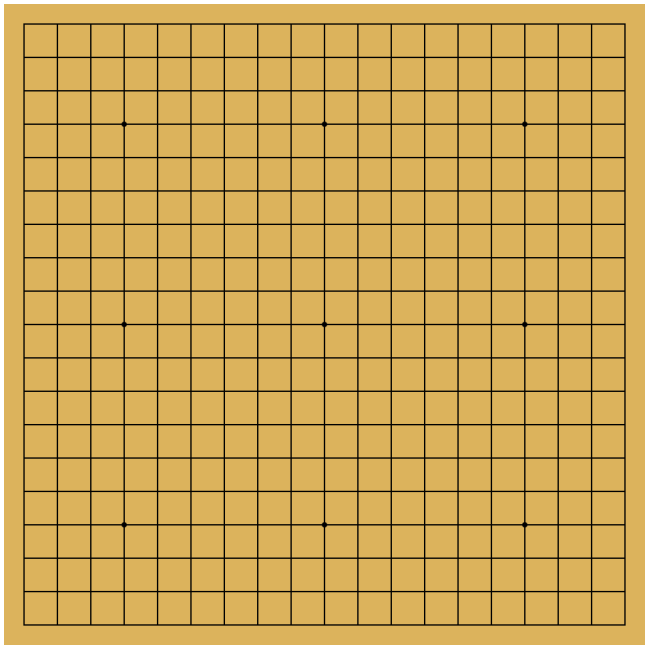
auf einem 19 x 19 Gitter

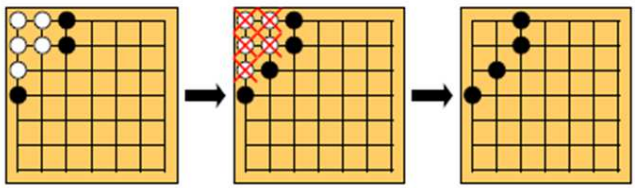
Ziel: Die Größte Fläche einkreisen

eingekreiste Steine werden weggenommen

Anzahl der Möglichkeiten: 250^{150}

Vergleich zu Schach: 35^{80}



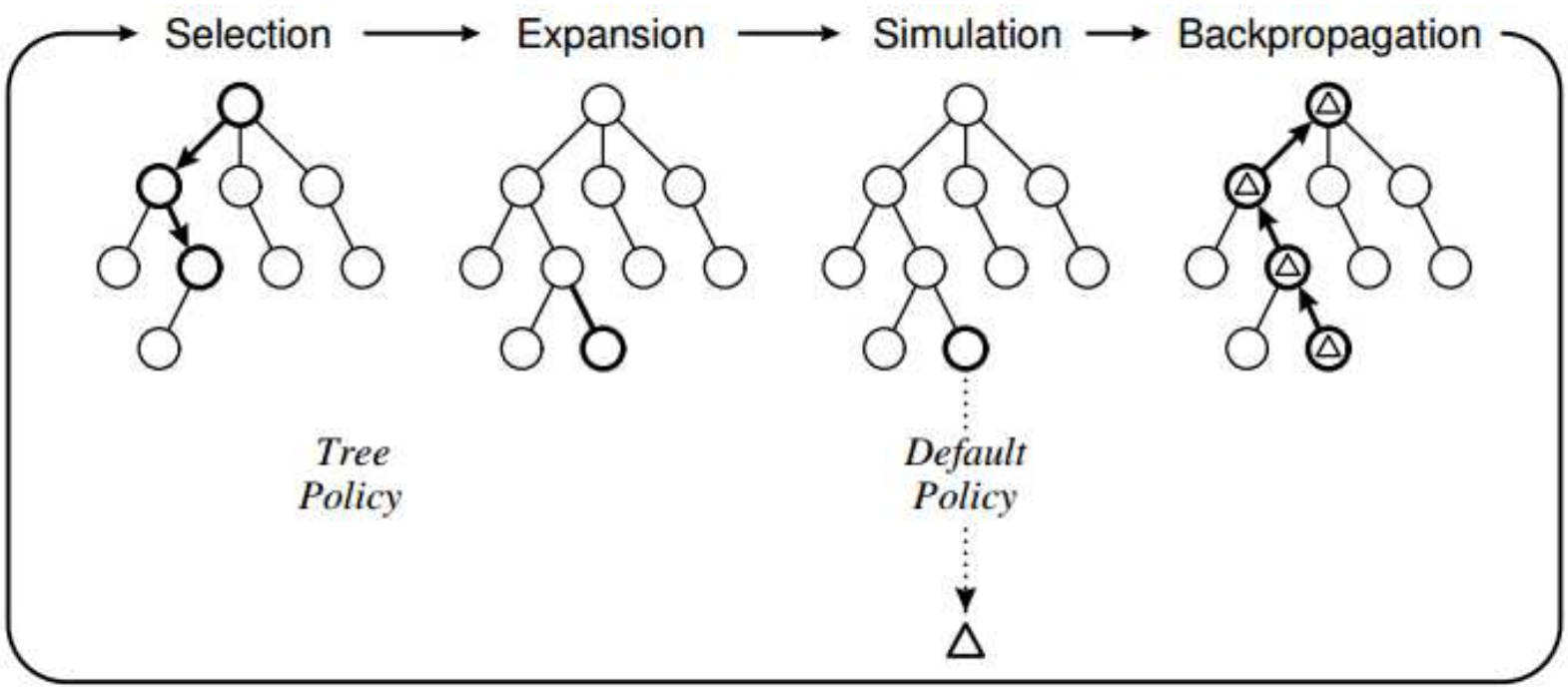


AlphaGo: Ansatz Monte Carlo Suche

Ansatz: Suche im Spielbaum

Lerne Netz 1 für menschenähnliche nächste Züge

Lerne Netz 2 zum Bewerten von Stellungen



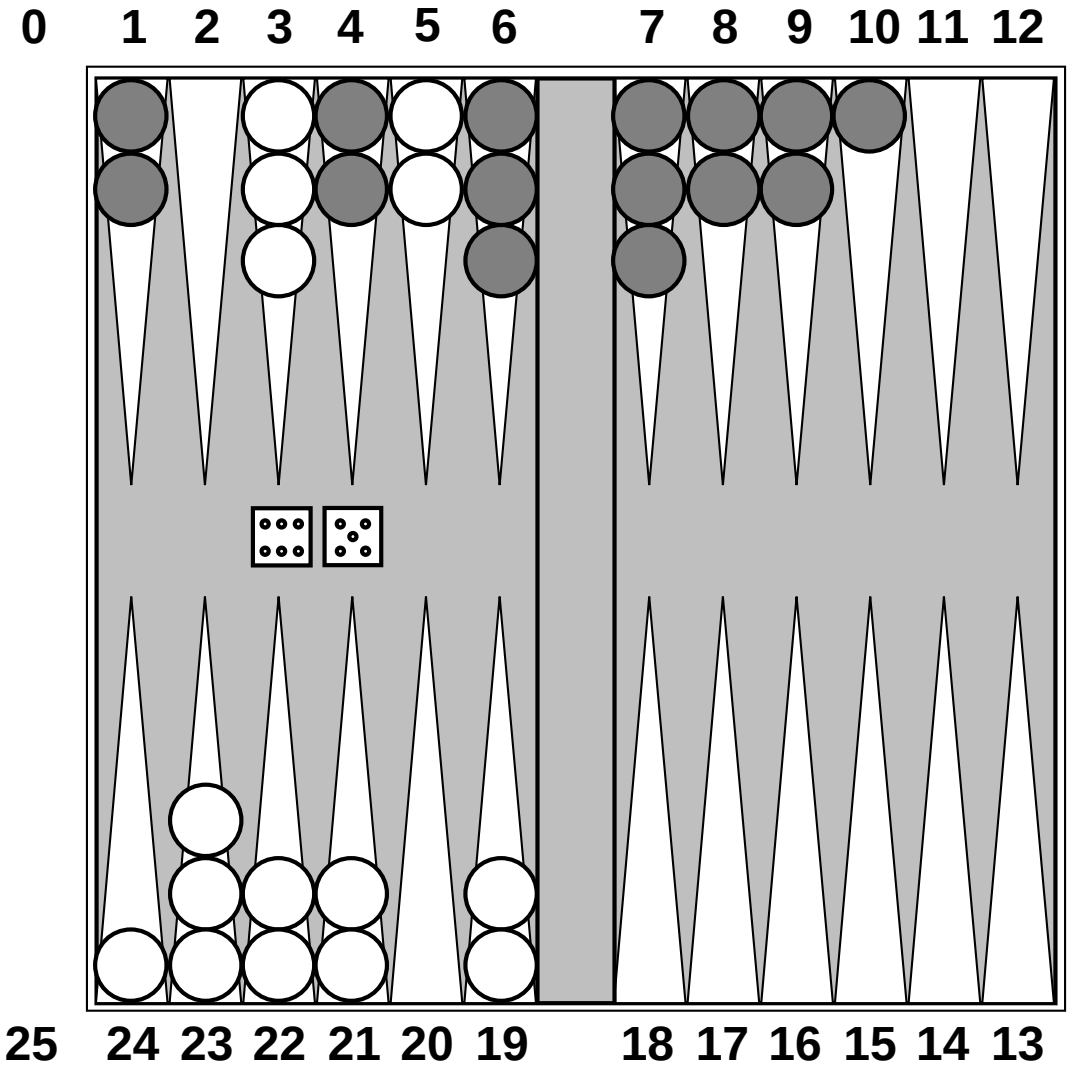
AlphaGo: Ergebnisse

Sieg gegen Europameister, Fan Hui: 5 zu 0

Sieg gegen Top10 der Weltrangliste, Lee Sedol: 4 zu 1

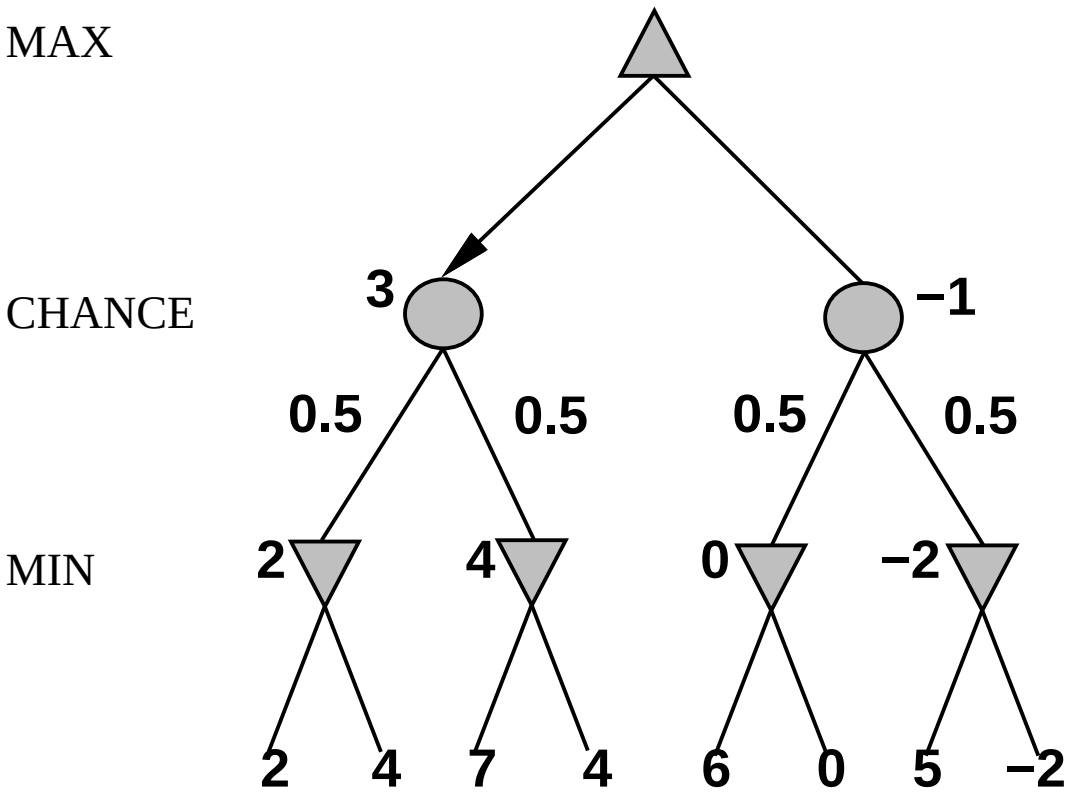
AlphaGo	Search threads	CPUs	GPUs	Elo
Asynchronous	1	48	8	2203
Asynchronous	2	48	8	2393
Asynchronous	4	48	8	2564
Asynchronous	8	48	8	2665
Asynchronous	16	48	8	2778
Asynchronous	32	48	8	2867
Asynchronous	40	48	8	2890
Asynchronous	40	48	1	2181
Asynchronous	40	48	2	2738
Asynchronous	40	48	4	2850
Distributed	12	428	64	2937
Distributed	24	764	112	3079
Distributed	40	1202	176	3140
Distributed	64	1920	280	3168

Glücksspiele: Backgammon



Glücksspiele allgemein

Zufall aufgrund von Würfeln, Mischen von Karten, etc.
Vereinfachtes Beispiel mit Münzwurf:



Algorithmus für nichtdeterministische Spiele

ExpectiMinimax liefert perfektes Spiel

Wie Minimax, nur Zufallsknoten müssen behandelt werden:

```
...
if state is a Max node {
    return  highest ExpectiMinimax-Value of Successors(state)
}
if state is a Min node {
    return  lowest ExpectiMinimax-Value of Successors(state)
}
if state is a chance node {
    return  average of ExpectiMinimax-Value of Successors(state)
}...
```

Spiele mit unvollständigen Informationen

Die meisten Kartenspiele (wie Bridge, Doppelkopf, Hearts, Mau-Mau, Poker, Siebzehn und vier, Skat) sind Nullsummenspiele mit unvollständigen Informationen.

Auch einige Brettspiele (Schiffe versenken, Kriegspiel-Schach).