# Schach-algorithmen

## und die Zukunft mit AI



Maximilian Hofmann, Denise Katritschenko, Florian Pögl

Einleitung

# Schach der letzten 80 Jahre

1943 - 1945

Konrad Zuse schreibt erstes Schachprogramm

1948

Alan Turing und David Champernowne entwickeln den “Turochamp”

1949

Claude Shannons Minimax-Algorithmus

1961

Dietrich Prinz’ Programm zur Lösung von Schachaufgaben

1983

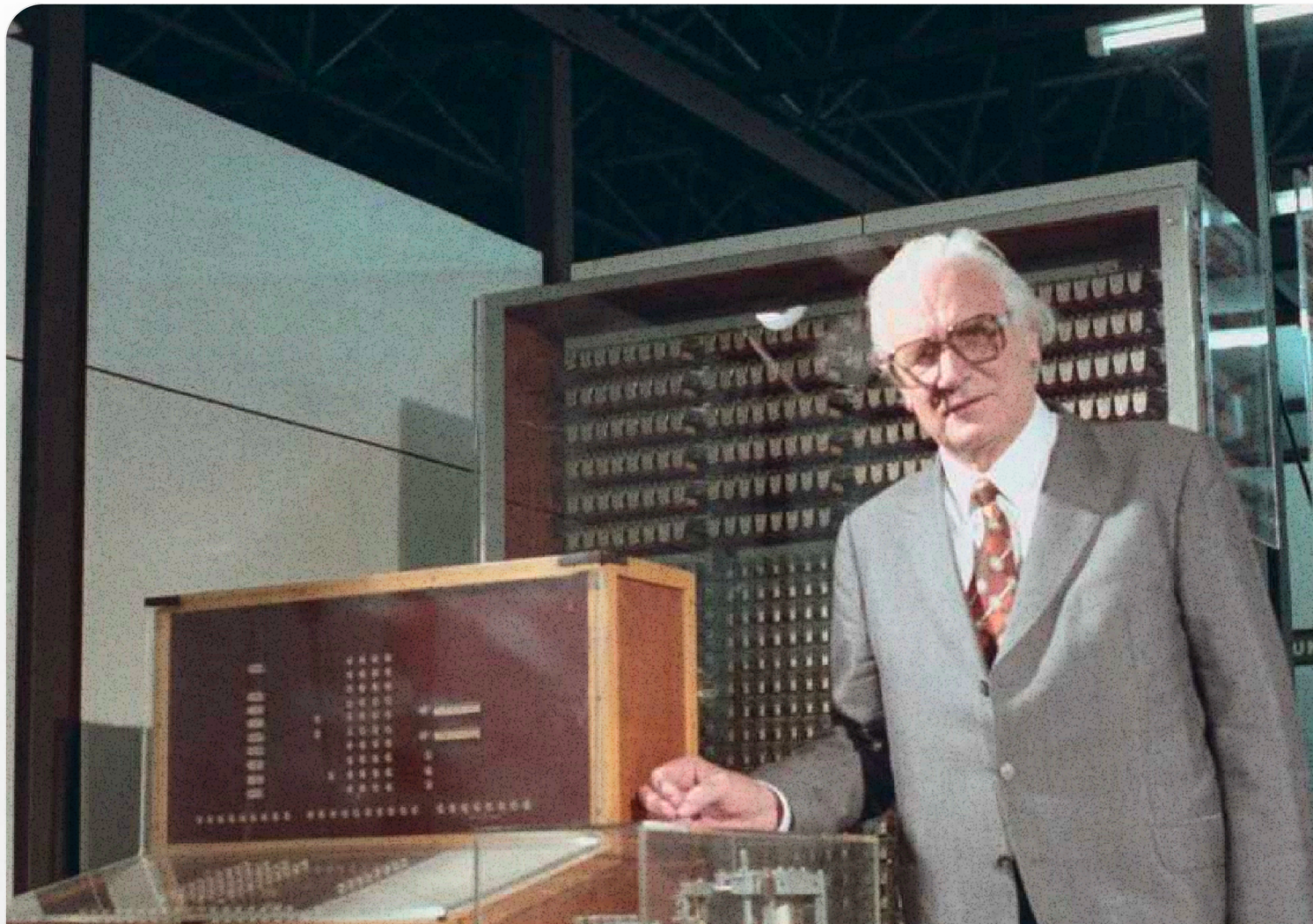
Schachprogramm „Belle“ (Ken Thompson und Joe Condon) erreicht eine Wertung von 2363

1997

“Deep Blue” bezwingt den Weltmeister Garry Kasparow mit 3,5 zu 2,5 Punkten

heute

Konrad Zuse vor dem Nachbau seiner Z3 im Deutschen Museum in München, Foto: Deutsches Museum



# Grundlagen von Schachalgorithmen

Grundlagen

# Grundidee des Spielbaums

Im Schach kann man sich das Spielfeld und alle möglichen Züge als einen Entscheidungsbaum vorstellen, den sogenannten Spielbaum

⚠ Problem: Spielbaum wird sehr groß!

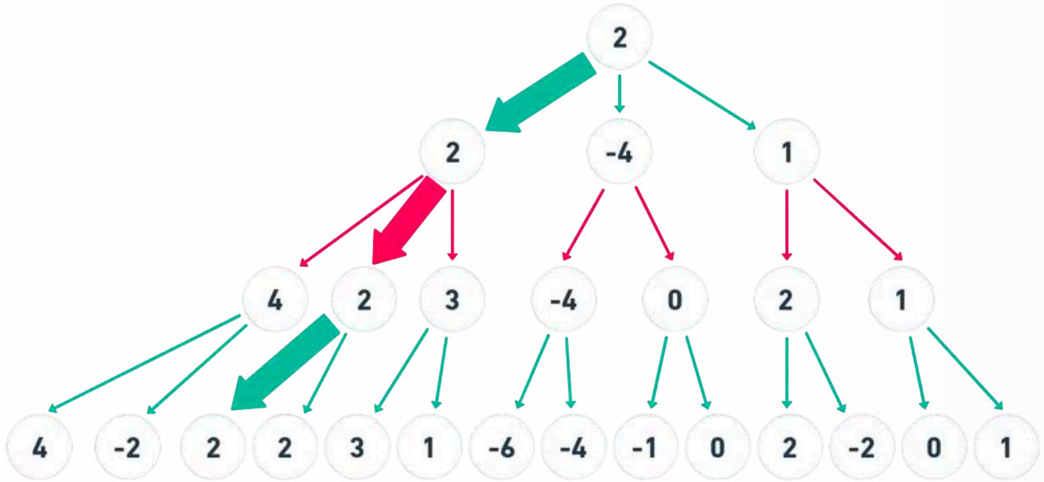



Bild: pexels.com

Grundlagen

# Komplexität des Schachspiels

10,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000

Anzahl der Atome im beobachtbaren Universum

Anzahl möglicher Schachstellungen (sog. Shannon-Zahl)

Grundlagen

# Bewertungsfunktionen und Heuristiken

## 1 Materialvorteil

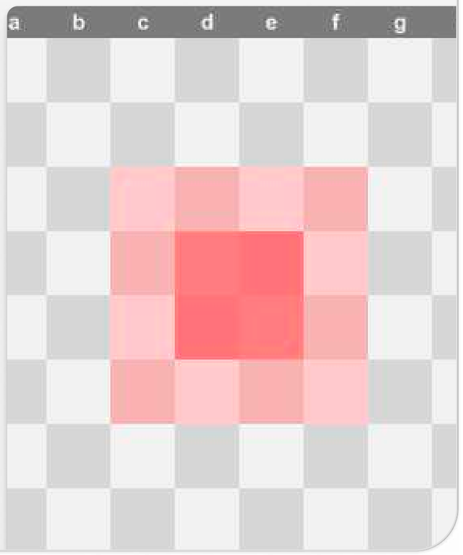
Differenz der Figurenwerte beider Seiten (z. B. Dame = 9 Punkte, Bauer = 1 Punkt)


## 2 King Safety

Ist der eigene König gut geschützt?

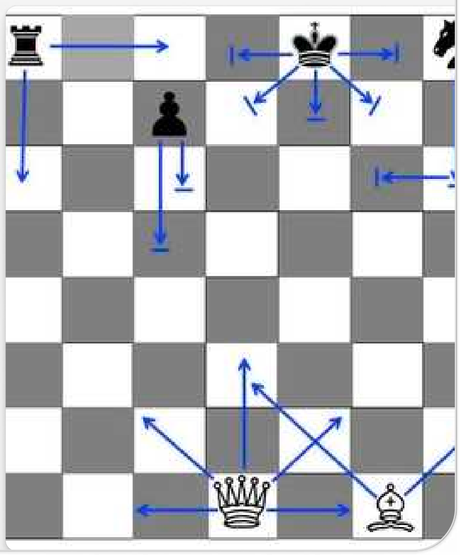
## 3 Kontrolle

Wie viele zentrale Felder werden kontrolliert?




## 4 Mobilität

Wie viele legale Züge sind verfügbar?



Bilder: pexels.com

Grundlagen

# Klassische Bewertungsfunktion

$$f(\text{Stellung}) = w_1 \times \text{Material}(\text{Stellung}) + w_2 \times \text{King Safety}(\text{Stellung}) + w_3 \times \text{Zentrumsbeherrschung}(\text{Stellung})$$

$w_n$ ist dabei ein sog. “weight” (Gewicht), das den Spielstil bestimmt

Horizont-Effekt

Grundlagen

# Grundlegende Suchprobleme

Optimierung des Spielbaums

# Monte Carlo Search Tree

(MCTS)

Monte Carlo Search Tree

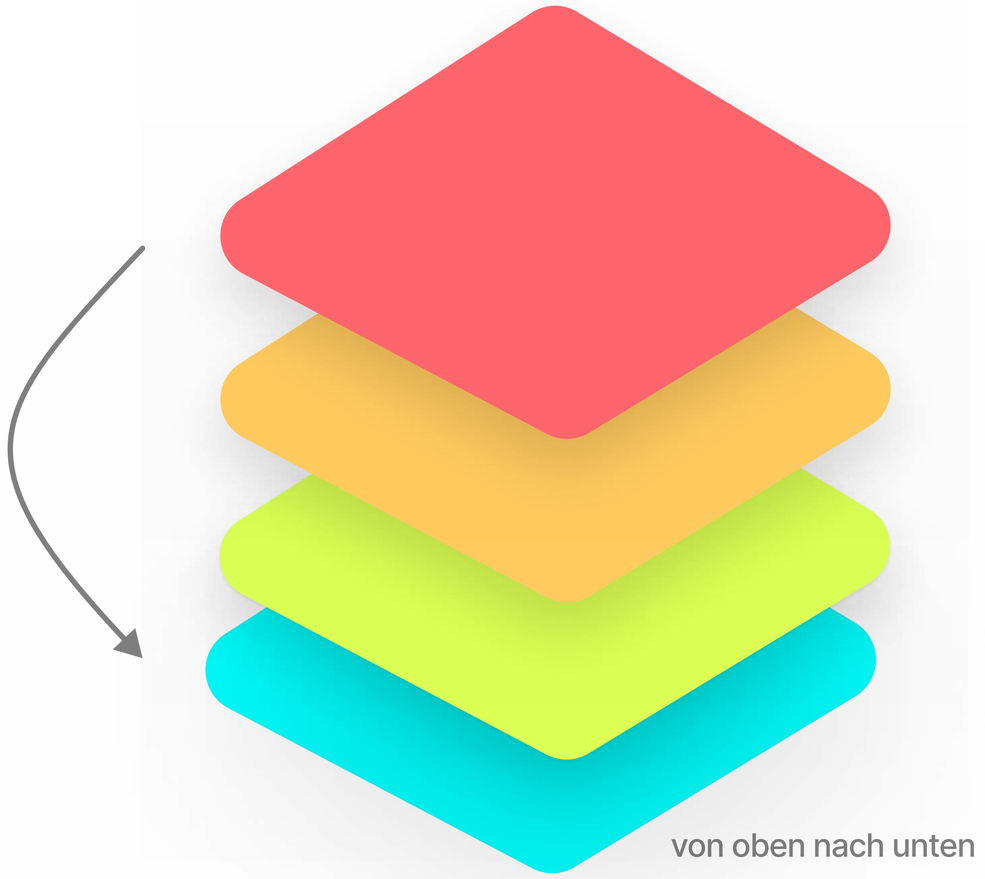
# Ablauf

**Selektion:** Finde den vielversprechensten Knoten für die weitere Analyse

**Expansion:** Füge neue Kindknoten für mögliche Züge ein

**Simulation:** Zufällige Simulation bis zu einem Endstand durchgeführt

**Rückpropagieren:** Die Simulation wird entlang des Pfades zur Wurzel rückpropagiert



von oben nach unten

Monte Carlo Search Tree

# 1. Selektion

- Suche nach einem Knoten zwischen neuen und vielversprechenden Zügen.
- Balance zwischen wenig getesteten und bewährten Zügen
- Auswahl mittels Upper Confidence Bound(UCB):

$$UCB = Winrate + C \times \sqrt{\frac{\ln(N)}{n}}$$

C = Konstante, die Verhältnis aus neu und bewährt steuert

N = Anzahl der Simulationen in übergeordneten Knoten

n = Anzahl der Simulationen im aktuellen Knoten

Monte Carlo Search Tree

# 2. Expansion

- Nach Knotenauswahl füge neue Kindknoten für mögliche Züge ein
- Neue Knoten repräsentieren bisher nicht simulierte Züge

Monte Carlo Search Tree

# 3. Simulation

- Um grobe Bewertung zu erhalten, wird zufällige Simulation bis zu einem Endstand durchgeführt
- Auswahl durch Zufall oder einfache Heuristik
- Bewertung durch Endzustand (Sieg, Niederlage, Unentschieden)

# 4. Rückpropagieren

- Ergebnis der Simulation wird entlang des Pfades von der Wurzel zum simulierten Knoten propagiert
- Folgende Daten werden für jeden Knoten aktualisiert
  - Anzahl der Besuche (n)
  - Anzahl der Siege (w)

Monte Carlo Search Tree

# Warum MCTS?

| Vorteile | Nachteile |
| --- | --- |
| • Bei großen Spielbäumen<br>  • muss nicht der gesamte Baum durchsucht werden<br>  • Algorithmus findet automatisch die besten Teilbäume | • Benötigt viele Simulationen für hohe Präzision, was bei Spielen mit knapper Zeit zu Problemen und sogar Fehlern führen kann (z.B.: Blitzschach) |
| • Kann dynamisch mehr Ressourcen auf kritische Spielpositionen konzentrieren | • Falls die Simulationen keine realistischen Züge finden, entstehen grobe Spielfehler |
| • Funktioniert ohne kompliziertere Bewertungsfunktionen oder Heuristiken | • Man braucht es eine Bewertungsfunktion für Konkurrenzfähigkeit (Wahl der Konstante C beeinflusst das Ergebnis stark |

# Minimax-Algorithmus

Minimax-Algorithmus

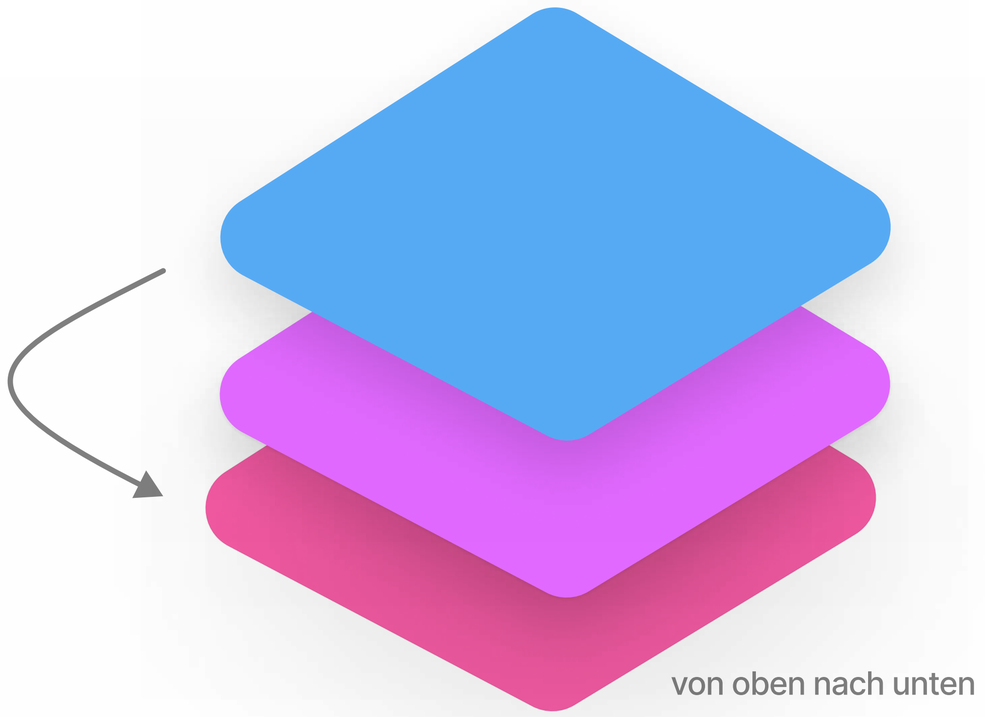
# Ablauf

- Baumstruktur: Knoten = Stellung, Kante = Zug
- Tiefensuche zur Entscheidungsfindung

Berechnung aller legalen Züge

Rekursive Bewertung bis zur maximalen Tiefe oder Spielende

Rückgabe der besten Zugoption



Minimax-Algorithmus

# Warum Minimax?

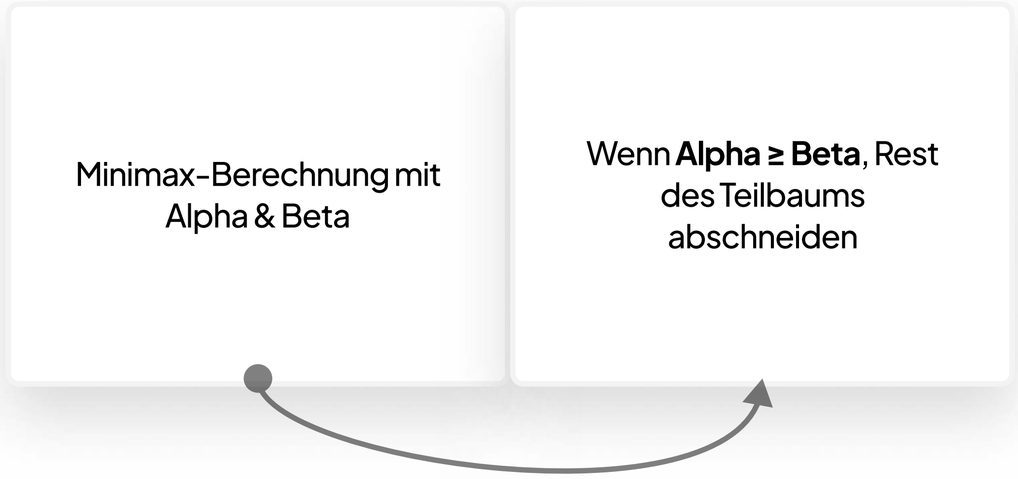
| Vorteile | Nachteile |
| --- | --- |
| • Einfach zu verstehen & implementieren | • Exponentielle Komplexität ($O(b^d)$) |
| • Optimal bei vollständiger Baumsuche | • Begrenzte Genauigkeit durch Suchtiefe |
| • Anwendbar auf 2-Spieler-Spiele | • Abhängig von Bewertungsfunktion |

Minimax-Algorithmus

# Alpha-Beta-Pruning (Optimierung)

**Ziel:** Reduktion der zu bewertenden Knoten durch Abschneiden unnötiger Zweige

- **Alpha:** Maximaler Wert für Spieler Max
- **Beta:** Minimaler Wert für Spieler Min




**Vorteile:**

- Komplexität verbessert von $O(b^d) \rightarrow O(b^{d/2})$
- Reduziert Berechnungen → höhere Effizienz
- Erlaubt tiefere Analysen in gleicher Zeit

# Aktuelle Forschungsansätze in Schach-KI

Probleme

## Energieeffizienz & Hardware

**Lösung**

Effizientere Algorithmen & spezialisierte Hardware

**Beispiel**

Deep Blue → Spezialchips, AlphaZero → GPUs, maßgeschneiderte Chips

## Schachvarianten & Flexibilität

**Lösung**

KI passt sich an Regeländerungen an

**Beispiel**

Reinforcement Learning ermöglicht schnelle Anpassung an neue Varianten

Aktuelle Forschung

# Probleme

## KI fehlt menschliche Intuition

**Lösung**

Erklärbare KI (XAI), hybride Systeme

**Beispiel**

AlphaZero imitiert Züge von GMs, psychologische Faktoren berücksichtigt

## Generalisierung & Anwendungsgrenzen

**Lösung**

Transfer Learning für breitere Anwendungen

**Beispiel**

AlphaZero lernte Schach, Go, Shogi gleichzeitig

## Historische Entwicklung

Von Minimax zu selbstlernenden Netzwerken (AlphaZero)

## Technische Fortschritte

neuronale Bewertungsfunktionen, energieeffiziente Algorithmen, Monte Carlo Search Tree

> Schach-KI zeigt, wie Innovation KI vorantreibt – von Spielstrategien bis zur Zukunft der Intelligenz.

## Zukunft der Forschung

- Erklärbare KI
- Generalisierte Systeme
- Mensch & Maschine

## Herausforderungen

Spielbaum-Komplexität, Generalisierung, Erklärbarkeit

# Fazit

> Schach ist die Drosophila der künstlichen Intelligenz
>
> inspiriert von Claude Shannon

## Bedeutung für KI-Forschung

- Schach als Testumgebung
- Kulturelle Relevanz