Monte Carlo Tree Search

und deren Anwendung in Echtzeitspielen

Dr. Johannes Riesterer

Motivation

Künstliche Intelligenz in Spielen

Monte Carlo Tree Search (MCTS) ist ein Verfahren um künstliche Intelligenz in Spielen zu implementieren.

Was ist ein Spiel?

Definition

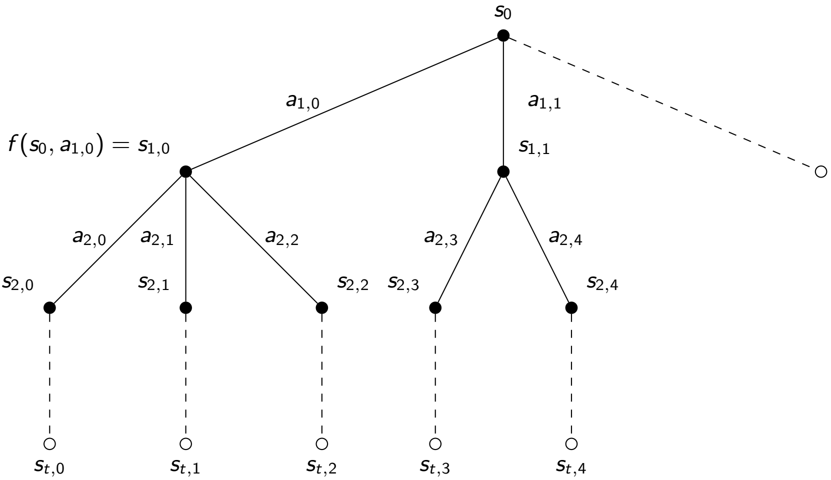
Ein Spiel besteht aus den folgenden Daten:

- Menge S von Zuständen.
- Startzustand $s_0 \in S$.
- Teilmenge $S_T \subset S$ von Terminalzuständen.
- Anzahl der Spieler $n \in \mathbb{N}$.
- Auswahlfunktion des Spielers $\rho : S \to \{1, \cdots, n\}$
- Menge A von Aktionen.
- Zustandsübergangsfunktuion $f : S \times A \to S$.
- Bewertungsfunktion $R : S^k \to \mathbb{R}$, die von den Spielern optimiert werden muss.

Definition

Eine Abbildung $P : S \to A$ heißt Strategie.

Gametree



s_0
$a_{1,0}$
$a_{1,1}$
$f(s_0, a_{1,0}) = s_{1,0}$
$s_{1,1}$
$a_{2,0}$
$a_{2,1}$
$a_{2,2}$
$a_{2,3}$
$a_{2,4}$
$s_{2,0}$
$s_{2,1}$
$s_{2,2}$
$s_{2,3}$
$s_{2,4}$
$s_{t,0}$
$s_{t,1}$
$s_{t,2}$
$s_{t,3}$
$s_{t,4}$

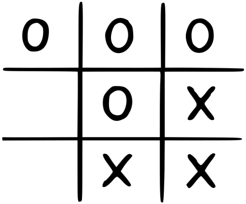
Optimale Strategie

Baumsuche

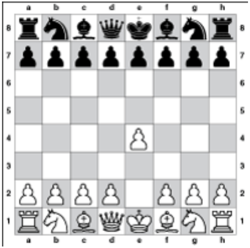
Durchsuche den Baum um Strategie auszuwählen.

Vergleiche hinreichend viele Möglichkeiten

Oft (nahezu) unmöglich und nur für triviale Spiele echtzeitfähig!

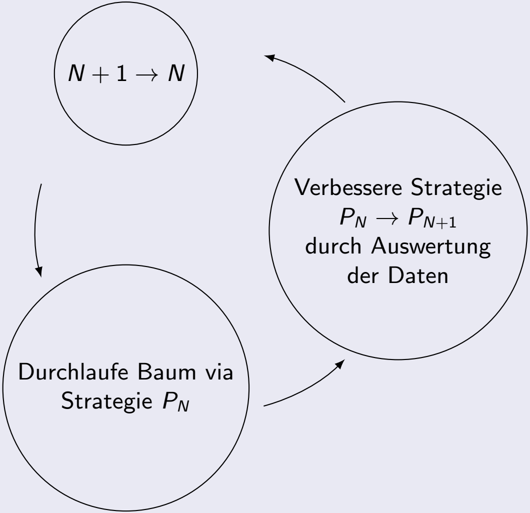


(a) 255.169 verschiedene Spielverläufe



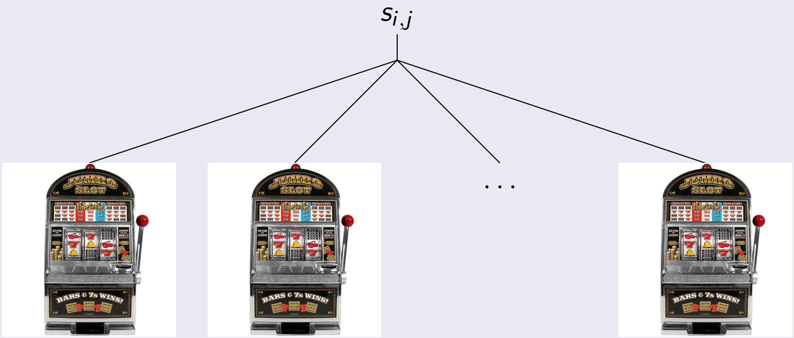
(b) Schachmatt in 42 Zügen?

Monte Carlo tree search



Monte Carlo tree search

Abstraktion und Unterteilung des Problems



K-Bandit Problem

An jedem Knoten s_{ij} das Problem:

- Gegeben sind k binäre Zufallsvariablen $X_1, \cdots, X_k \in \{0, 1\}$ mit unbekannten Wahrscheinlichkeitsfunktion $p_1, \cdots, p_k$ (Banditen).
- Eine der Zufallsvariablen X_i wird ausgewählt und eine Stichprobe durchgeführt.
- Verbessere Auswahlstrategie P_N, so dass nach N Durchgängen möglichst oft die Zufallsvariable mit dem höchsten Erwartungswert (höchste Gewinnwahrscheinlichkeit) ausgewählt wurde.

Gesetz der großen Zahlen

- Seien $(X_m)_1, \cdots, (X_m)_N$ Stichproben der m-ten Zufallsvariable und $(\overline{X_m})_N := \frac{1}{N} \sum_{i=1}^{N} (X_m)_i$. Dann gilt

$$(\overline{X_m})_N \xrightarrow{N \to \infty} \mathbb{E}(X_m) \text{ (in Wahrscheinlichkeit).}$$

- Je öfter man einen Banditen spielt, desto verlässlicher können wir den durchschnittlichen Gewinn durch den momentan beobachteten durchschnittlichen Gewinn annähern.

Ausbeutung VS Erkundung

Möglichst oft den Banditen mit momentan besten beobachteten durchschnittlichen Gewinn spielen und trotzdem alle Banditen häufig genug ausprobieren.

ϵ-greedy

P_N :

- Sei j der Index mit $(\overline{X}_j)_N = \max\limits_{1 \le i \le k} (\overline{X}_i)_N$.
- wähle mit Wahrscheinlichkeit ϵ den Banditen X_j und mit Wahrscheinlichkeit $1 - \epsilon$ zufällig einen der Übrigen.

$P_N \to P_{N+1}$:

- Update $(\overline{X}_j)_{N+1}$

UCB1

P_N :

- Wähle Bandit X_j, so dass $(\overline{X}_j)_N + \sqrt{\frac{2\ln(N)}{T_j(N)}}$ maximal ist.

$P_N \to P_{N+1}$:

- Update $(\overline{X}_j)_{N+1}$, $T_j(N+1)$

Durchschnittlicher Fehlgriff

$$T_i(N) := \text{Anzahl der Wahlen von } X_i \text{ nach } N \text{ Durchgängen}$$
$$\mu_j := \mathbb{E}(X_j)$$
$$\mu := \max_{1 \le i \le k} \mu_i$$

Der Regret ist definiert durch $R(N) := \mu \cdot N - \sum_{i=1}^{k} \mu_i \cdot \mathbb{E}(T_i(N))$.

ϵ-greedy

$R_N < O(N)$ ist linear begrenzt.

UCB1

$R_N < O(Log(N))$ ist logarithmisch begrenz

Zusammenfassung

- Wende an jedem Knoten s_{ij} des Gamtree eine fest gewählte Monte-Carlo-Strategie an um nächsten Knoten auszuwählen (z.B. ϵ-greedy, UCB1).
- Der Baum wird dann entsprechend dieser Regel durchlaufen.
- Gelangt man an einen Terminalzustand, so werden die gewonnen Information zum update der Strategie jedes Knotens angewendet.

MCTS mit UCB1

Mittels Monte Carlo Tree search, bei der UCB1 als Standardstrategie verwendet wird, wurden erstmals nennenswerte Erfolge in Go erzielt!