Meilenstein 2 - Basis-KI

Gruppe AC: [AC/DC!]

Mitglieder:

- Amar Salcinovic
- Carlos Andres Sulbaran Fandino
- Peter Wladislaw Urbansky

Benchmark Tests

Hardware Angabe:
CPU: 13th Gen Intel(R) Core(TM) i7-1355U 1.70 GHz
RAM: 32,0 GB (31,6 GB verwendbar)
Systemtyp: 64-Bit-Betriebssystem, x64-basierter Prozessor
Programmiersprache: Python
Zustandsrepräsentation: Bitboard (64 Bits)

Fen-String: b0b0b0b0b0b0/1b0b0b0b0b0b01/8/8/8/8/1r0r0r0r0r0r01/r0r0r0r0r0r0

Tiefe	Anzahl Zustände	Benötigte Zeit (ms)	Bester Zug (Blau)	Algorithmus
0	1	-	-	-
1	34	16.190	B2-B3	Alpha-Beta
2	420	182.287	B2-B3	Alpha-Beta
3	7058	2.691.641	B1-B2	Alpha-Beta
4	42062	25.466.070	B1-B2	Alpha-Beta
Gesamt	49575	28.360.147	B1-B2	Alpha-Beta

Tiefe	Anzahl Zustände	Benötigte Zeit (ms)	Bester Zug (Blau)	Algorithmus
0	1	-	-	-
1	34	14.165	B2-B3	Minimax
2	1190	434.156	B2-B3	Minimax

3	37434	13.795.259	B1-B2	Minimax
4	1173790	509.619.080	B1-B2	Minimax
Gesamt	1212449	523.867.099	B1-B2	Minimax

Fen-String: b01bbb01b0/1b02b03/3bbr01b01/8/3rr1b0b01/8/2r01r01rr1/r0r0r01r01

Tiefe	Anzahl Zustände	Benötigte Zeit (ms)	Bester Zug (Blau)	Algorithmus
0	1	-	-	-
1	15	8.492	F3-E5	Alpha-Beta
2	159	83.061	F3-E5	Alpha-Beta
3	1479	645.617	F3-E5	Alpha-Beta
4	13266	6.539.096	F3-E5	Alpha-Beta
Gesamt	14920	7.279.903	F3-E5	Alpha-Beta

Tiefe	Anzahl Zustände	Benötigte Zeit (ms)	Bester Zug	Algorithmus
0	1	-	-	Minimax
1	28	11.449	D1-E3	Minimax
2	732	291.404	D1-E3	Minimax
3	20452	7.942.662	F5-G5	Minimax
4	518688	201.569.849	F5-G5	Minimax
Gesamt	539901	209.818.990	F5-G5	Minimax

FEF-String: 2b02bb/1bb2b03/5bb2/8/1r03r02/6r01/8/r01r01rrr0

Tiefe	Anzahl Zustände	Benötigte Zeit (ms)	Bester Zug (Blau)	Algorithmus
0	1	-	-	-
1	15	8.492	F3-E5	Alpha-Beta

2	159	83.061	F3-E5	Alpha-Beta
3	1479	645.617	F3-E5	Alpha-Beta
4	13266	6.539.096	F3-E5	Alpha-Beta
Gesamt	14920	7.279.903	F3-E5	Alpha-Beta

Tiefe	Anzahl Zustände	Benötigte Zeit (ms)	Bester Zug (Blau)	Algorithmus
0	1	-	-	-
1	15	6.991	F3-E5	Minimax
2	299	124.736	F3-E5	Minimax
3	4915	1.888.264	F3-E5	Minimax
4	94046	37.010.718	F3-E5	Minimax
Gesamt	99276	39.036.471	F3-E5	Minimax

Anmerkungen:

Die Tabellen zeigen deutlich die Überlegenheit von Alpha-Beta gegenüber Minimax in Bezug auf die Anzahl der untersuchten Zustände und die benötigte Zeit.

In allen Fällen findet Alpha-Beta den gleichen besten Zug wie Minimax, aber mit deutlich weniger Aufwand.

Planung des 3. Meilensteins: Ausbau der KI

Ziel: Entwicklung einer verbesserten KI, die strategisch anspruchsvollere Entscheidungen treffen kann.

Mögliche KI-Techniken und Vorhaben sortiert nach Priorität:

1. **Zugsortierung:** Die Implementierung einer Zugsortierung kann uns dabei helfen, enorm an Rechenleistung zu sparen, da starke Züge früh durchlaufen werden und somit viele Cut-offs geschehen können.
2. **Serverkommunikation:** Bisher haben wir noch keine Methoden für die Kommunikation mit dem Spielserver geschrieben. Dafür wollen wir voraussichtlich eine "Middleman"-Klasse schreiben, die genau dafür zuständig sein wird.
3. **Zeitmanagement:** Unser Algorithmus soll die Zeit, die ihm noch bleibt und die bereits vergangen ist, im Blick behalten und so dynamisch seine maximale Suchtiefe anpassen.
4. **Verschärfte Heuristiken durch evolutionäre Optimierung:** Wir haben bereits Heuristiken erstellt, indem wir durch Ausprobieren und gegeneinander Spielen erste Überlegungen gemacht haben, welche Positionen stark sind und welche wir vermeiden wollen. Wir hatten Anfangs viele Ideen, welche Strategien wir verfolgen wollen und haben uns zuerst dafür entschieden, die KI so simpel wie möglich zu halten, da wir uns nicht in zu hoher Komplexität verhaspeln wollen. Komplexer machen im Nachhinein, können wir immer.
   Im weiteren Projektverlauf, wollen wir die Heuristiken untersuchen und verbessern, indem wir evolutionäre Optimierung durchführen. Dabei sollen zwei KIs mit mutierten Heuristiken einige Male gegeneinander spielen. Die stärkere KI kommt in die nächste “Runde” und tritt danach gegen eine wieder mutierte Version von sich selbst an. Damit erhoffen wir uns, dass unsere KI lernt, ohne ein neuronales Netz zu nutzen.
5. **Zobrist-Hashing und Transpositionstabelle:** Sowohl die Implementierung von Zobrist-Hashing, als auch ihre Erweiterung zu einer Transpositionstabelle, wollen wir bald schaffen. Das würde uns ermöglichen, bereits untersuchte Stellungen zu speichern und im Laufe der Alpha-Beta-Suche diese bei wiederholtem Auftreten zu ignorieren.

6. **Ruhe-Suche (Quiescence-Search):** Implementierung einer Ruhe-Suche, um in unruhigen Stellungen (z.B. nach Schlagzügen) eine tiefere Suche durchzuführen und somit bessere Entscheidungen zu treffen.

Zusätzliche Vorhaben und Ideen:

- **Spielanalyse:** Wenn die Zeit da ist und wir wissen wie, ist eine Überlegung, Spiele nach immer wiederkehrenden Positionen zu analysieren und diese vielleicht statistisch zu modellieren. So können wir vorhersagen, welche Spielsituationen wichtig sind und anhand dieser testen, wie stark unsere KI spielt.
- **Fortgeschrittene Strategien:** Auch weiterhin bleibt die Idee einer Schwarmintelligenz oder der Deklaration von “König” Steinen im Ideen-Pool. Bislang haben wir uns eher darauf konzentriert eine funktionierende und relativ simple KI zu schreiben, in der Hoffnung, dass im späteren Verlauf des Projektes unser Code genug Modularität aufweist, um solche fortgeschrittenen Strategien einzuführen.

Deadline für den 3. Meilenstein: 13.06.2024
Für die Implementierung unserer Vorhaben brauchen wir voraussichtlich drei Wochen. Dabei wollen wir besonders darauf achten, dass die Implementierung sauber ist, da die Code-Komplexität immer höher wird und wir vermeiden wollen, dass es in Spaggethi-Code ausartet.