

Nutzung der Neuartigkeit von Zuständen in Suchgraphen von AlphaZero

Utilizing Novelty of States in the Searchgraph of AlphaZero

Bachelorarbeit im Studienbereich Computational Engineering von Markus Reuter

Tag der Einreichung: 15. März 2023

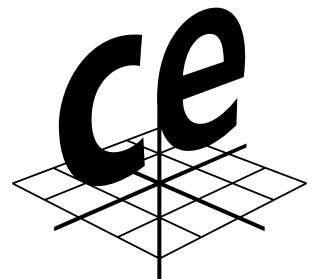
1. Gutachten: Prof. Dr. Kristian Kersting

2. Gutachten: M.Sc. Johannes Czech

Darmstadt



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Studienbereich
Computational Engineering



Artificial Intelligence &
Machine Learning Lab

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

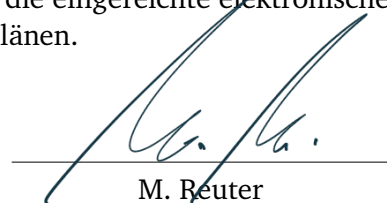
Hiermit versichere ich, Markus Reuter, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 15. März 2023



M. Reuter

Zusammenfassung

Die Monte-Carlo Tree Search ermöglicht Computerprogrammen wie *AlphaZero* komplexe Spiele wie Schach und Go zu erlernen. Damit die Suche nach optimalen Folgeaktionen noch bessere Ergebnisse liefert, wurde die Monte-Carlo Tree Search stetig verbessert und mit weiteren Merkmalen ausgestattet. Eine Methode, die bereits bei dem Einsatz von linearen heuristischen Evaluierungsfunktionen positive Resultate erzielen konnte, ist die Integration der Neuartigkeit von Zuständen in den UCT-Selektionsalgorithmus. Das Ziel der Nutzung von Neuartigkeit ist die gezielte Verbesserung der Exploration von vielversprechenden Folgeaktionen. In dieser Arbeit wird untersucht, ob der Einsatz von Neuartigkeit in der PUCT-Selektionsformel zu Verbesserungen der Suche führt, wenn neuronale Netze zur Zustandsevaluation verwendet werden. Außerdem wird die Nutzung der Neuartigkeit von Zuständen auf die Monte-Carlo Graph Search übertragen, sodass potentiell die Vorteile beider Ansätze kombiniert werden. Die empirische Evaluation anhand der *CrazyAra* Engine zeigt, dass die Nutzung von Neuartigkeit in der Form, wie sie hier verwendet wird, keine signifikanten Verbesserungen bei Schach und Crazyhouse erzielt.

Stichworte: Neuartigkeit, Monte-Carlo Tree Search, PUCT, Schach, Crazyhouse, Monte-Carlo Graph Search

Abstract

The Monte-Carlo Tree Search allows computer programs like *AlphaZero* to learn complex games such as Chess and Go. To achieve better search results for optimal subsequent actions, the Monte-Carlo Tree Search has been improved continuously and enhanced with additional features. One method that has already achieved positive results in the use of linear heuristic evaluation functions is the integration of novelty of states into the UCT selection algorithm. The goal of using novelty is to target the improvement of exploration of promising subsequent actions. This paper investigates whether the use of novelty in the PUCT selection formula leads to improvements in the search when using neural networks for state evaluation. In addition, the use of novelty of states is transferred to the Monte-Carlo Graph Search, potentially combining the advantages of both approaches. The empirical evaluation using the *CrazyAra* engine shows that the use of novelty in the form as used here does not achieve significant improvements in Chess and Crazyhouse.

Keywords: Novelty, Monte-Carlo Tree Search, PUCT, Chess, Crazyhouse, Monte-Carlo Graph Search

Inhaltsverzeichnis

1. Einleitung	7
1.1. Motivation	7
1.2. Formulierung des Problems	8
1.3. Aufbau der Arbeit	9
2. Hintergrund	10
2.1. Monte-Carlo Tree Search	10
2.2. Selektionsalgorithmen	12
2.2.1. Upper Confidence Bounds for Trees - UCT	12
2.2.2. Policy Upper Confidence Bounds for Trees - PUCT	13
3. Verwandte Arbeiten	14
3.1. Prinzip der Neuartigkeit	14
3.1.1. Neuartigkeit durch Zustandsevaluation - <i>Evaluation Novelty</i>	15
3.1.2. Neuartigkeit durch Merkmalszählung - <i>Pseudocount Novelty</i>	15
3.2. Selektion mit Neuartigkeit	17
3.3. Monte-Carlo Graph Search	18
3.4. Die <i>CrazyAra</i> Engine	19
4. Methodik	23
4.1. Repräsentation der Fakten	23
4.2. Neuartigkeit und MCGS	25
5. Empirische Evaluation	28
5.1. Spieleinstellungen	28
5.2. Hyperparameteroptimierung	29
5.2.1. Hyperparameter für die <i>Evaluation Novelty</i>	30
5.2.2. Hyperparameter für die <i>Pseudocount Novelty</i>	32
5.2.3. Hyperparameter für Neuartigkeit bei der MCGS	32



5.3. Elo-Entwicklung	35
5.3.1. Elo-Entwicklung für die <i>Evaluation Novelty</i>	35
5.3.2. Elo-Entwicklung für die <i>Pseudocount Novelty</i>	35
5.3.3. Elo-Entwicklung für die <i>Evaluation Novelty</i> bei MCGS	37
6. Diskussion	40
7. Fazit	44
7.1. Zusammenfassung	44
7.2. Zukünftige Arbeit	45
A. Anhang	49

1. Einleitung

Dieses Kapitel gibt eine Einleitung in das Thema der Arbeit. Zunächst wird das Thema anhand einiger historischer Meilensteine eingeführt, aus denen sich wiederum die Ziele dieser Arbeit und die Motivation für weitere wissenschaftliche Untersuchungen ergeben. Danach folgt die Formulierung des Problems und zum Schluss wird der Aufbau dieser Arbeit erläutert.

1.1. Motivation

Schon seit den 1950er Jahren gibt es immer wieder Versuche, den menschlichen Spieler in Schach mit Hilfe von Computerprogrammen zu schlagen. Einen maßgeblichen Durchbruch hat der von IBM entwickelte Computer *DeepBlue* (Campbell et al., 2002) erzielt, als er 1997 den damaligen Schachgroßmeister Garry Kasparov besiegte. Ein neuer Standard wurde 2018 von der Firma DeepMind mit dem Programm *AlphaZero* (Silver et al., 2018) gesetzt. Das Besondere an *AlphaZero* ist, dass es mithilfe von Reinforcement Learning Schach lernt, indem es gegen sich selbst spielt. Dadurch benötigt *AlphaZero* kein Domänenwissen und keine menschlichen Trainingsdaten, um das Schachspielen zu erlernen, was eine Neuerung zu den bisherigen etablierten Schachengines darstellte. Nach nur vier Stunden Trainingszeit konnte *AlphaZero* das bis dahin beste Schachprogramm Stockfish in 100 Partien schlagen. Der Algorithmus zur Auswahl der Folgezüge beruht bei *AlphaZero* auf der Monte-Carlo Tree Search (MCTS), die seither zum Standard bei Spielen mit großem Suchraum geworden ist.

In den vergangenen Jahren wurde immer wieder versucht, die MCTS durch Hinzunahme weiterer Faktoren zu verbessern. Daran wird diese Arbeit anschließen, indem die Nutzung der Neuartigkeit von Zuständen in die MCTS eingebunden wird. Frühere Arbeiten konnten bereits zeigen, dass die Integration von Neuartigkeit in die Selektion mit UCT

zu positiven Ergebnissen führen kann. In den Spielen Vier-Gewinnt, Othello, Breakthrough sowie Knightthrough konnten Baier und Kaisers, 2021 signifikante Verbesserungen durch die Nutzung der Neuartigkeit von Zuständen gegenüber der vanilla MCTS erzielen. Dort wurden lineare heuristische Evaluierungsfunktionen für die Zustandsbewertung verwendet. In dieser Arbeit wird untersucht, ob die Nutzung von Neuartigkeit auch im Rahmen der PUCT-Selektion in Kombination mit Machine Learning (ML) Modellen zur Zustandsevaluation, Verbesserungen erzielen kann. Die Auswertung wird in dieser Arbeit anhand der noch komplexeren Spiele Schach und Crazyhouse¹ durchgeführt.

1.2. Formulierung des Problems

In der MCTS muss bei der Selektion eine Abwägung zwischen der Ausnutzung von vorhandenem Wissen und der Exploration von neuen Aktionen getroffen werden. Dies ist eine bekannte Herausforderung im Bereich des Reinforcement Learnings (Sutton und Barto, 2018). Ein zu starker Fokus auf die Ausnutzung von Wissen resultiert leicht in der Annäherung an ein lokales Minimum. Die übermäßige Exploration führt wiederum oft zur Wahl von Folgeaktionen, die wenig vielversprechend sind. In dieser Arbeit soll die Exploration durch die Nutzung der Neuartigkeit von Zuständen verbessert werden. Das Ziel besteht darin, zu Beginn vielversprechende Aktionen stärker zu explorieren ohne die langfristige Konvergenz der Suche einzuschränken.

Die Nutzung der Neuartigkeit von Zuständen wurde bisher nur für die UCT-Selektion in Kombination mit linearen heuristischen Evaluierungsfunktionen ausgewertet. In der Praxis hat sich jedoch für komplexe Spiele wie Schach und Go die Selektion mit PUCT etabliert, da diese eine Policy-Approximation in die Wahl der Folgeaktion einbezieht. Diese Art der Selektion wird meistens mit tiefen neuronalen Netzen zur Zustandsevaluation und Policy-Approximation verbunden, da dadurch eine höhere Evaluationsgüte erreicht werden kann. In dieser Arbeit wird daher ausgewertet, ob die Nutzung von Neuartigkeit bei der PUCT-Selektion in Kombination mit ML Modellen zu ähnlich positiven Resultaten führt.

¹Crazyhouse ist eine Schachvariante, bei der geschlagene Figuren nicht endgültig vom Brett verschwinden. Jeder Spieler hat sogenannte Taschen, in denen er die Figuren, die er vom Gegner geschlagen hat, sammelt. Wenn ein Spieler am Zug ist, kann er statt einem Zug, eine Figur aus seiner Tasche auf ein beliebiges freies Feld auf dem Brett stellen. Dadurch ist die Komplexität von Crazyhouse noch größer als die von klassischem Schach.

1.3. Aufbau der Arbeit

Diese Arbeit ist in mehrere Kapitel gegliedert. In Kapitel 2 werden die nötigen Grundlagen für die Arbeit erläutert. Dies umfasst die detaillierte Beschreibung der MCTS und der beiden Selektionsalgorithmen UCT und PUCT.

Im darauffolgenden Kapitel 3 werden verwandte Arbeiten vorgestellt. In besonderem Maße wird auf die Arbeit von Baier und Kaisers, 2021 eingegangen, in der verschiedene Neuartigkeitsmetriken vorgestellt wurden, die hier verwendet werden. Außerdem umfasst die Arbeit von Baier und Kaisers, 2021 die Anpassung des Selektionsalgorithmus der MCTS, um die Nutzung der Neuartigkeit von Zuständen zu ermöglichen. Zum Schluss wird in diesem Kapitel die *CrazyAra* Engine eingeführt, die in dieser Arbeit verwendet wird, um die Ansätze zu implementieren und zu evaluieren.

Kapitel 4 umfasst die Methodik, mit der in dieser Arbeit Neuartigkeit in die MCTS und die Monte-Carlo Graph Search integriert wird.

Danach folgt die empirische Evaluation der Ergebnisse in Kapitel 5. Dort werden zunächst die eingeführten Hyperparameter mit einer Rastersuche optimiert. Anschließend werden für die besten Kombinationen der Hyperparameter ausführliche Tests durchgeführt, um die Elo-Entwicklung relativ zur Anzahl der Evaluationen des neuronalen Netzes zu ermitteln.

In den letzten beiden Kapiteln 6 und 7 werden die Ergebnisse diskutiert und zusammengefasst. Zum Schluss wird ein Ausblick für zukünftige Arbeiten gegeben.

2. Hintergrund

Dieses Kapitel umfasst die konzeptuelle Beschreibung der Monte-Carlo Tree Search (MCTS) mit den beiden Selektionsalgorithmen UCT und PUCT.

2.1. Monte-Carlo Tree Search

Ziel der MCTS ist die Wahl der optimalen Folgeaktion. Dazu wird eine Schleife mit vier Schritten durchlaufen, wie in Abbildung 2.1 dargestellt. Diese Schleife wird so oft wiederholt, bis der Spielbaum vollständig durchsucht wurde oder die Suche durch ein Abbruchkriterium terminiert ist. Der durch die MCTS entstehende Suchbaum ist immer eine Teilmenge des gesamten Spielbaums. Bei Schach entspricht ein Knoten des Baumes einem Zustand bzw. einer Stellung auf dem Schachbrett. Die Pfade repräsentieren die gültigen Aktionen eines Agenten, die von einem zulässigen Zustand in den nächsten führen. Im Folgenden werden die vier Schritte näher erläutert:

1. **Selection:** Bei der Selektion wird ausgewählt, welche Aktion a_t aus dem aktuellen Zustand s_t zum Zeitschritt t als nächstes ausgeführt wird. Dieser Selektionsprozess wird so lange fortgeführt, bis ein neuer unbesuchter Zustand s^* oder ein finaler Zustand s_T ausgewählt wird. Der Selektionsalgorithmus startet immer im Wurzelknoten und wählt rekursiv den besten Kindknoten nach dem Selektionsalgorithmus. Unbesuchte Knoten werden bei der Selektion als Verlierer behandelt und haben den Wert -1 . Durch diesen rekursiven Prozess entsteht eine Suchtrajektorie von der Wurzel bis zu einem unbesuchten- oder finalen Zustand.
2. **Expansion:** In diesem Schritt wird der in der Selektion ausgewählte Knoten an den Suchbaum angehängt. Dieser Schritt wird übersprungen, falls die Selektion einen finalen Zustand s_T auswählt, da dann ein Blattknoten im Spielbaum erreicht ist.

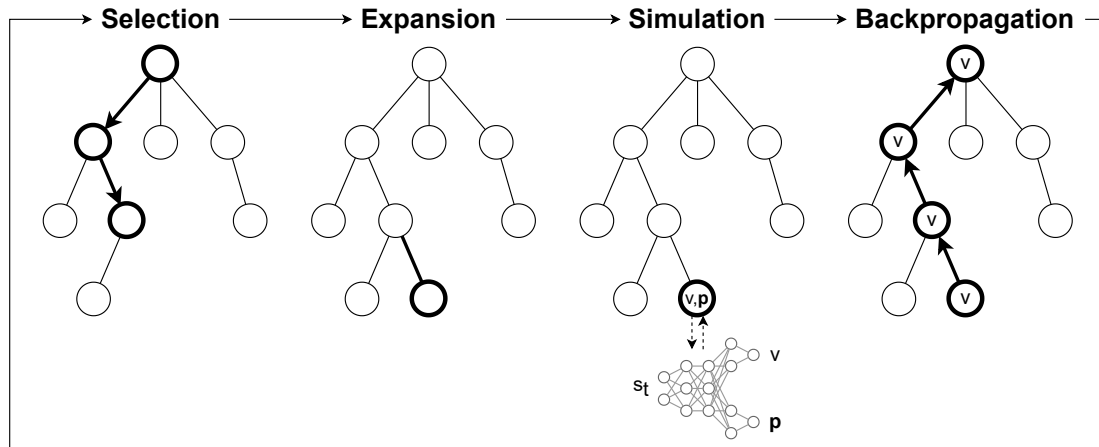


Abbildung 2.1.: Visualisierung der MCTS-Schleife. Nachdem der Agent in einen neuen Zustand der Umgebung gelangt, werden die Schritte **Selection**, **Expansion**, **Simulation** und **Backpropagation** durchlaufen. Dadurch wird iterativ ein Suchbaum aufgebaut. Die Abbildung basiert auf einer Grafik von Browne et al., 2012.

3. **Simulation:** Bei der Simulation (auch Rollout genannt) wird der Wert des bei der Selektion ausgewählten Knotens bestimmt. Für den Fall, dass ein finaler Zustand s_T erreicht wurde, ist der exakte Wert v_T für diesen Knoten bekannt (+1 für Sieg, 0 für Unentschieden und -1 für Niederlage aus Sicht des Spielers, der am Zug ist). Wenn ein neuer unbesuchter Zustand s^* selektiert wurde, muss ein Rollout durchgeführt werden. Allgemein wird hierbei durch die Rollout-Policy ein Wert für den unbesuchten Zustand s^* bestimmt. Die naive Rollout-Policy besteht darin, aus dem aktuellen unbesuchten Zustand zufällige Folgeaktionen zu wählen, bis ein finaler Zustand erreicht wurde. In modernen Engines wird ein neuronales Netz f_θ verwendet, um den Wert v^* des Zustandes und die Policy $P(s, a_i)$ für die möglichen Aktionen zu ermitteln.
4. **Backpropagation:** Im letzten Schritt wird der durch die Simulation gefundene Wert v^* entlang der gesamten Suchtrajektorie zurückpropagiert. Die Q-Values (Sutton und Barto, 2018) werden nach jedem Schritt mit -1 multipliziert, um den Wert v^* für die gegnerische Seite anzupassen und danach zu einem einfachen gleitenden

Durchschnitt

$$Q'(s_t, a) = Q(s_t, a) + \frac{1}{N(s_t, a)}(v^* - Q(s_t, a)) \quad (2.1)$$

zusammengefasst. Der Wert $N(s_t, a)$ repräsentiert die Anzahl, wie oft Aktion a aus Zustand s_t gewählt wurde.

Schlussendlich wird eine Aktion ausgehend vom Wurzelknoten ausgewählt, der nach wie vor den aktuellen Zustand in der Umgebung darstellt. Es gibt verschiedene Möglichkeiten, nach welchen Kriterien diese Aktion ausgewählt wird. In dieser Arbeit wird die Aktion aus dem Wurzelknoten gewählt werden, die während der Suche am meisten selektiert wurde. Für den weiteren Verlauf dieser Arbeit hat die finale Auswahl der auszuführenden Aktion ausgehend vom Wurzelknoten jedoch keine unmittelbare Bedeutung.

2.2. Selektionsalgorithmen

Um zu verstehen wie sich der Suchbaum aufbaut, muss der Selektionsalgorithmus der MCTS genauer betrachtet werden. In der Vergangenheit hat sich gezeigt, dass eine gute Abwägung zwischen der Ausnutzung von vorhandenem Wissen und der Exploration von neuen Aktionen eine Herausforderung darstellt. Es haben sich die beiden Selektionsalgorithmen UCT und PUCT etabliert, die in den nächsten zwei Abschnitten definiert werden.

2.2.1. Upper Confidence Bounds for Trees - UCT

Die Upper Confidence Bounds for Trees (UCT) ist die für Monte-Carlo Suchbäume angepasste UCB1-Formel (Auer et al., 2002), die erstmals in *AlphaGo* (Silver, Schrittwieser et al., 2017) mit der MCTS verknüpft wurde. Die Selektion der Folgeaktion a_t , die zu dem nächsten Zustand führt, ist durch

$$a_t = \operatorname{argmax}_a (Q(s_t, a) + U(s_t, a)) \quad \text{mit} \quad U(s, a) = c \sqrt{\frac{\ln N(s_t)}{N(s_t, a)}} \quad (2.2)$$

beschrieben, wobei $N(s_t, a)$ die Häufigkeit darstellt, wie oft die Aktion a des Zustandes s zum Zeitpunkt t schon gewählt wurde. Der Gewichtungsfaktor c bestimmt den Grad zwischen Exploration und Ausnutzung und $N(s_t)$ die Anzahl der Knotenbesuche während der Suche.

2.2.2. Policy Upper Confidence Bounds for Trees - PUCT

Die Policy Upper Confidence Bounds for Trees (PUCT) ist eine erweiterte Version der UCT, die erstmals von Rosin, 2011 veröffentlicht und später von Silver, Schrittwieser et al., 2017 angepasst wurde. Im Vergleich zur UCT wird bei der PUCT der U-Value $U(s, a)$ anders berechnet. Die durch das neuronale Netz f_θ ermittelte Policy $P(s, a)$, die eine a-priori Verteilung über alle zulässigen Folgeaktionen a darstellt, ermöglicht es bei der Selektion vielversprechende Aktionen stärker zu gewichten. Daraus folgt, je besser die Policy-Bewertung des neuronalen Netzes ist, desto besser ist auch die Selektion. Gleichzeitig wird ein Explorationsanteil beibehalten, um potentiellen Unsicherheiten der Policy $P(s, a)$ entgegenzuwirken. Der Selektionsalgorithmus für PUCT ist durch

$$a_t = \operatorname{argmax}_a (Q(s_t, a) + U(s_t, a)) \quad \text{mit} \quad U(s, a) = c_{puct}(s) P(s, a) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)} \quad (2.3)$$

beschrieben. Der Gewichtungsfaktor $c_{puct}(s)$ bestimmt analog zum UCT-Algorithmus die Gewichtung von Exploration und Ausnutzung in einem Zustand. Die Berechnung von $c_{puct}(s)$ erfolgt über

$$c_{puct}(s) = \log \frac{\sum_a N(s, a) + c_{puct-base} + 1}{c_{puct-base}} + c_{puct-init}, \quad (2.4)$$

wobei die Parameter $c_{puct-base}$ und $c_{puct-init}$ basierend auf Silver, Schrittwieser et al., 2017 als $c_{puct-base} = 19\,652$ und $c_{puct-init} = 2,5$ gewählt werden.

3. Verwandte Arbeiten

In diesem Kapitel wird zunächst auf das Prinzip der Neuartigkeit von Zuständen und dessen Integration in die Selektionsformel eingegangen. Danach wird das Konzept der Monte-Carlo Graph Search (MCGS) beschrieben. Zuletzt wird die *CrazyAra* Engine vorgestellt.

3.1. Prinzip der Neuartigkeit

Der Ansatz, Neuartigkeit zum Finden einer optimalen Folgeaktion zu verwenden, wurde bereits für einige Probleme angewendet (Tot et al., 2022; Lehman und Stanley, 2011). Das Ziel besteht unter anderem darin, Sackgassen und lokale Minima bei der Suche zu vermeiden, sowie die Exploration zu verbessern (Baier und Kaisers, 2021). Auch im Bereich von (Brett-)Spielen konnten bereits positive Resultate durch die Verwendung von Neuartigkeit erreicht werden. In der Arbeit von Baier und Kaisers, 2021 wurden drei verschiedene Neuartigkeitsmetriken in den UCT-Selektionsalgorithmus der Monte-Carlo Tree Search (MCTS) integriert und an den Spielen Vier-Gewinnt, Othello, Breakthrough sowie Knightthrough evaluiert. Indem Baier und Kaisers, 2021 den Neuartigkeitsfaktor in Form einer Linearkombination in die UCT-Selektionsformel eingebunden haben, konnte mit einem Verfallsparemeter eine anfängliche Kontrolle der Exploration gewährleistet werden, ohne die Konvergenzgarantien zu reduzieren.

Um zu verstehen, wie die Neuartigkeit berechnet wird, muss zunächst erläutert werden, wie ein Zustand aufgebaut ist. Eine konkrete Anordnung der Figuren auf dem Schachfeld entspricht einem Zustand s aus der Menge aller möglichen Zustände S . Dieser ist aus verschiedenen unabhängigen Fakten f aus der Menge aller möglichen Fakten F aufgebaut. Ein Fakt setzt sich aus einem Feld und dem ihm zugewiesenen Wert zusammen, beispielsweise ist die Kombination „weißer Bauer auf d4“ ein Fakt. Im Folgenden werden zwei Metriken aus der Arbeit von Baier und Kaisers, 2021 vorgestellt, um Neuartigkeit von Zuständen zu bewerten.

3.1.1. Neuartigkeit durch Zustandsevaluation - *Evaluation Novelty*

Die sogenannte *Evaluation Novelty* beruht darauf, den durch die Evaluation des neuronalen Netzes f_θ ermittelten Wert v^* eines Zustandes in die Neuartigkeitsbewertung einfließen zu lassen. Der Neuartigkeitswert eines Faktes $N_t(f)$ zu einem Zeitpunkt t berechnet sich durch

$$N_t(f) = \begin{cases} \max_{s \in S_t, f \in s} v(s) & \text{wenn } f \in s \text{ für ein } s \in S_t \\ -1 & \text{sonst.} \end{cases} \quad (3.1)$$

Dabei ist S_t als die Menge aller bisher gefundenen Zustände definiert. Der Wert $N_t(f)$ entspricht also immer dem maximalen Wert eines Zustandes, der bereits evaluiert wurde und Fakt f beinhaltet. Die obige Formel weicht leicht von der aus Baier und Kaisers, 2021 ab, da der Initialwert für $N_t(f)$ dort $-\infty$ beträgt. Der Zustandswert $V(s_t)$ kann jedoch nur im Bereich von $[-1, 1]$ liegen, weshalb der Initialwert hier auf -1 gesetzt wird.

Der Neuartigkeitswert eines Zustandes $M_E(s)$ ergibt sich dann zu

$$M_E(s) = \begin{cases} \alpha & \text{wenn } V(s) > N_t(f) \text{ für ein } f \in s \\ 0 & \text{sonst.} \end{cases} \quad (3.2)$$

Der Wert α stellt einen Hyperparameter dar, der die Belohnung für die Neuartigkeit eines Zustandes steuert. Ein Zustand wird nach dieser Logik als neuartig bewertet, wenn der Wert des betrachteten Zustandes $V(s)$ größer ist als der Neuartigkeitswert $N_t(f)$ eines beliebigen Faktes, der im betrachteten Zustand s vorkommt. Außerdem ist zu sehen, dass die Bewertung der Neuartigkeit eines Zustandes binär ist. Statt der atomaren Fakten können auch andere Merkmale für die *Evaluation Novelty* verwendet werden.

3.1.2. Neuartigkeit durch Merkmalszählung - *Pseudocount Novelty*

Die *Pseudocount Novelty* beruht auf dem ϕ -*Exploration-Bonus*-Algorithmus, der von Martin et al., 2017 eingeführt wurde. Der Neuartigkeitswert wird dabei nicht mithilfe einer heuristischen Evaluierungsfunktion berechnet, sondern mit der Wahrscheinlichkeitsverteilung von Merkmalen über einen Zustand. Die Merkmalsfunktion $\phi : S \rightarrow T$ stellt die Abbildung aus dem Zustandsraum S in den M -dimensionalen Merkmalsraum T dar. Außerdem bezeichnet $\phi_t = \phi(s_t)$ den M -dimensionalen Merkmalsvektor des Zustandes s zum Zeitpunkt t .

Daraus lässt sich die Merkmalsdichte durch

$$\rho_t(\phi) = \prod_{i=1}^M \rho_t^i(\phi_i) \quad (3.3)$$

berechnen. Das Produkt setzt sich aus unabhängigen Faktorverteilungen $\rho_t^i(\phi_i)$ der individuellen Merkmale ϕ_i zusammen. Diese werden hier, wie in der Arbeit von Martin et al., 2017, durch den Krichevsky-Trofimov (KT) Schätzer

$$\rho_t^i(\phi) = \frac{C_t(\phi_i) + \frac{1}{2}}{t + 1} \quad (3.4)$$

ermittelt. Dabei stellt $C_t(\phi_i)$ die Häufigkeit dar, wie oft Merkmal ϕ_i bis zum Zeitpunkt t bereits beobachtet wurde.

Die Merkmalsdichte $\rho_t(\phi)$ ermöglicht einen einfachen Vergleich, ob neu beobachtete Zustände mehr oder weniger neuartig gegenüber den zuvor beobachteten Zuständen sind. Wenn ein Zustand mehr neuartige Merkmale enthält, folgt daraus ein niedrigeres $\rho_t(\phi)$.

Die Wahl der Merkmalsfunktion ϕ hat signifikanten Einfluss auf die Neuartigkeitsbewertung. In dieser Arbeit werden die Fakten F als Merkmalsraum verwendet, sodass $F = T$ gilt. Wenn zum Zeitpunkt $t = 1$ die erste Beobachtung durchgeführt wird, gilt $C_1(\phi_i) = 1$ für alle 32 aktiven Merkmale $\phi_i = f_i \subset F$. Der KT-Schätzer berechnet sich dann für alle Merkmale zu $\rho_t^i(\phi) = 0,75$, was zu einer Merkmalsdichte von $\rho_t(\phi) = (\frac{3}{4})^{32} \approx 1,0045 \cdot 10^{-4}$ führt.

Mithilfe der Merkmalsdichte $\rho_t(\phi)$ kann als nächstes der ϕ -pseudocount als

$$\hat{C}_t^\phi(s) = \frac{\rho_t(\phi(s))(1 - \rho'_t(\phi(s)))}{\rho'_t(\phi(s)) - \rho_t(\phi(s))} \quad (3.5)$$

definiert werden. Hier stellt $\rho'_t(\phi)$ die Merkmalsdichte dar, wenn die selben Merkmale zum Zeitpunkt $t + 1$ erneut beobachtet werden. Im obigen Beispiel ergibt sich $\rho'_t(\phi) = (\frac{5}{6})^{32} \approx 2,9255 \cdot 10^{-3}$, was zu einem ϕ -pseudocount von $\hat{C}_t^\phi(s) \approx 0,0355$ führt.

Der Neuartigkeitswert eines Zustandes s ist dann als

$$M_C(s) = \frac{\alpha}{\sqrt{\hat{C}_t^\phi(s)}} \quad (3.6)$$

definiert. Ein niedriger ϕ -pseudocount Wert impliziert folglich einen hohen Neuartigkeitswert.

3.2. Selektion mit Neuartigkeit

Neben der durch das neuronale Netz f_θ gefundenen Zustandsevaluierung v^* muss zusätzlich der ermittelte Neuartigkeitswert m^* entlang der Suchtrajektorie zurückpropagiert werden. Um die Backpropagation analog zu den Q-Values durchführen zu können, müssen die Neuartigkeitswerte an den Kanten des Suchbaumes gespeichert werden. Im Folgenden definiert $M(s, a)$ den Neuartigkeitswert des Zustands-Aktionspaares (s, a) . Die Backpropagation des ermittelten Neuartigkeitswertes m^* erfolgt äquivalent zu den Q-Values durch die Bildung eines einfachen gleitenden Durchschnitts. Für jeden Knoten s der Suchtrajektorie muss demnach die Aktualisierung

$$M'(s_t, a) = M(s_t, a) + \frac{m^* - M(s_t, a)}{N(s_t, a)} \quad (3.7)$$

durchgeführt werden. Dabei stellt $N(s_t, a)$ die Anzahl dar, wie oft Aktion a aus Zustand s bis zum Zeitpunkt t bereits gewählt wurde und $M(s_t, a)$ den Neuartigkeitswert für eine Aktion a aus dem Zustand s_t .

Die entscheidende Änderung der Suche entsteht in der Selektionsformel für die Wahl der Folgeknoten. Es gibt mehrere Möglichkeiten den Selektionsalgorithmus anzupassen und Neuartigkeit in die Auswahl einfließen zu lassen. Die Ausführung von Baier und Kaisers, 2021 lässt die Neuartigkeit durch

$$a_t = \operatorname{argmax}_a (bM(s_t, a) + (1 - b)Q(s_t, a) + U(s_t, a)) \quad (3.8)$$

in die Selektion einfließen. Der Gewichtungskoeffizient b ergibt sich aus

$$b = \sqrt{\frac{\beta}{3N(s_t, a) + \beta}}, \quad (3.9)$$

mit β als Hyperparameter, der den Abfall von b über die Zeit steuert. In anderen Worten bestimmt β wie schnell die Selektion die Neuartigkeit von gefundenen Zuständen vernachlässigen soll, um sich mehr auf den Q-Value zu fokussieren und somit die ursprüngliche Formel anzustreben. Dadurch bleibt das Konvergenzverhalten der Selektion unverändert. Die Formel wurde ursprünglich von Gelly und Silver, 2007 zur Anwendung für *Rapid Action Value Estimates* entwickelt.

Anders als bei Baier und Kaisers, 2021, die den UCT-Algorithmus modifiziert haben, wird hier die PUCT-Selektionsformel angepasst. Gleichung 3.8 gilt weiterhin, jedoch ist der U-Value nicht wie bei UCT durch Gleichung 2.2 definiert, sondern durch Gleichung 2.3, wie in Abschnitt 2.2 dargelegt.

Initialisierung des Neuartigkeitswertes Damit unbesuchte Aktionen bei der Selektion weiterhin mit der gleichen Wahrscheinlichkeit ausgewählt werden, muss eine Besonderheit bei der Initialisierung beachtet werden. Für unbesuchte Aktionen a eines Zustandes s_t zum Zeitpunkt t gilt $N(s_t, a) = 0$. Daher besitzt b aus Gleichung 3.9 in diesem Fall immer den Wert 1. Das führt dazu, dass der Q-Value nicht in die Selektion der Folgeaktion a aus Zustand s_t einfließt, sondern die Gewichtung vollständig auf dem Neuartigkeitswert $M(s, a)$ liegt. Um die Selektion von unbesuchten Aktionen unverändert zu lassen, muss den Neuartigkeitswerten $M(s, a)$ zu Beginn der Initialwert der Q-Values (hier -1) zugewiesen werden.

3.3. Monte-Carlo Graph Search

Die MCGS ist eine modifizierte Version der MCTS, bei der sogenannte Transpositionen nur einmal gespeichert und evaluiert werden. Als Transposition wird ein Zustand bezeichnet, der durch verschiedene Zugfolgen erreicht wurde, also mehrere Elternknoten besitzt. Dadurch wird der Suchbaum zu einem gerichteten azyklischen Graphen (Saffidine et al., 2012), wie in Abbildung 3.1 illustriert. Wenn bei einer Suche ein neuer Zustand gefunden wird, der bereits im Rahmen einer anderen Zugfolge evaluiert wurde, ist es nicht nötig, das neuronale Netz für eine erneute Evaluation aufzurufen, da ein signifikanter Informationsgewinn unwahrscheinlich ist. Somit kann Speicherplatz und wertvolle Zeit für die restliche Suche gespart werden.

Für den Algorithmus der MCGS gibt es mehrere Möglichkeiten. In diesem Abschnitt wird sich auf die von Czech et al., 2021 vorgestellte Variante beschränkt, da diese auch in der hier verwendeten *CrazyAra* Engine (siehe Abschnitt 3.4) implementiert ist. Bei dieser Variante wird die Backpropagation weiterhin nur entlang der traversierten Trajektorie durchgeführt. Das kann unter Umständen zu einem Informationsverlust führen, wenn die traversierte Trajektorie eine Transposition enthält. Um das zwischenzeitliche Informationsdefizit ausgleichen zu können, wird es nötig, neben den Q-Values $Q(s_t, a)$ für alle zulässigen Aktionen a eines Zustandes s zum Zeitpunkt t , den aktuellen Zustandswert $V(s_t)$ für alle Zustände zu speichern. Die Zustandswerte $V(s_t)$ werden analog zu den Q-Values bei der Backpropagation aktualisiert und enthalten immer den Wert des Zustandes ohne Informationsdefizit. Für alle Knoten, die keine Transposition abbilden, gilt

$$Q(s_t, a) = -V(s_{t+1}), \quad (3.10)$$

wobei a die Aktion repräsentiert, die von dem Elternknoten in den entsprechenden Kindknoten führt. Die Negierung des Zustandswertes von Knoten s_{t+1} muss vorgenommen werden, da der Folgeknoten bei der Suche immer dem Gegner zugeordnet ist. Ein zwischenzeitliches Informationsdefizit tritt auf, wenn eine Suchtrajektorie eine bereits expandierte Transposition enthält. Dann werden bei der Backpropagation nur die Zustandswerte und Q-Values der traversierten Knoten und Kanten aktualisiert. Dadurch werden die Q-Values $Q(s_t, a)$ der Aktionen a^* , die in eine Transposition führen, jedoch selber nicht traversiert wurden, nicht aktualisiert. Eine Veranschaulichung eines solchen Szenarios ist in Abbildung 3.2 zu finden.

Aus diesem Grund muss bei der Selektion einer bereits evaluierten Transposition immer geprüft werden, ob ein Informationsausgleich stattfinden muss. Um festzustellen, ob ein Informationsdefizit vorliegt, wird zunächst der Q-Value $Q(s_t, a)$ der Aktion a , die in die Transposition führt mit dem Zielwert

$$V^*(s_t) = -V(s_t) \quad (3.11)$$

der Transposition verglichen. Wenn die Differenz

$$Q_\delta(s_t, a) = V^*(s_{t+1}) - Q(s_t, a) \quad (3.12)$$

kleiner als ein vorgegebener Schwellwert $Q_\epsilon(s_t, a)$ ist, wird von keinem Informationsverlust ausgegangen. In diesem Fall folgt die nächste Selektion anhand des PUCT-Algorithmus. Falls jedoch $Q_\delta(s_t, a) > Q_\epsilon(s_t, a)$ wird die Suchtrajektorie an der Transposition beendet und der Korrekturwert

$$Q_\phi(s_t, a) = N(s_t, a)Q_\delta(s_t, a) + V^*(s_{t+1}) \quad (3.13)$$

wird zurückpropagiert. Dadurch wird das Informationsdefizit ausgeglichen und ein erneutes Aufrufen des neuronalen Netzes kann vermieden werden. Der Hyperparameter $Q_\epsilon(s_t, a)$ wird wie von Czech et al., 2021 empfohlen als 0,01 gewählt.

Bei diesem MCGS Algorithmus werden alle optimalen Garantien der MCTS weiterhin erfüllt, wie schon von Czech et al., 2021 diskutiert.

3.4. Die CrazyAra Engine

Die in dieser Arbeit untersuchten Ansätze werden in die Schachengine *CrazyAra*¹ eingebettet und durch Spiele gegen die Engine evaluiert. Daher erscheint es als sinnvoll einen

¹<https://github.com/QueensGambit/CrazyAra>, besucht am 29.12.2022

kurzen Überblick über *CrazyAra* zu geben (siehe auch bei Czech et al., 2020).

CrazyAra ist eine frei zugängliche Multi-Varianten Schachengine, die auf einem tiefen neuronalen Netz $f_\theta(s)$ basiert. Letzteres wurde zunächst auf menschlichen Spieldaten trainiert und kann optional durch Reinforcement Learning verfeinert werden. Das neuronale Netz f_θ dient zur Vorhersage des Zustandswertes v^* eines Zustandes s und der Ermittlung der zugehörigen stochastischen Policy-Verteilung $P(s, a)$ über alle zulässigen Aktionen a des Zustandes s . Die Auswahl der Folgeaktion wird in *CrazyAra*, analog zu *AlphaZero*, durch die MCTS ermittelt, die das neuronale Netz f_θ integriert.

Ursprünglich wurde *CrazyAra* im Rahmen eines Semesterprojekts von Johannes Czech, Moritz Willig und Alena Beyer mit dem Ziel entwickelt, ein neuronales Netz zu trainieren, mit dem die Schachvariante Crazyhouse gespielt werden kann. Das Projekt wurde hauptsächlich inspiriert durch die Techniken aus den Arbeiten von Silver et al., 2018 und Silver, Hubert et al., 2017. Im Jahr 2018 hat *CrazyAra* ein Spiel aus fünf Partien gegen den damaligen Crazyhouse Weltmeister JannLee mit 4:1 gewonnen². Seitdem wurde das Projekt von Johannes Czech weiterentwickelt. Der Name *CrazyAra* entstand durch eine Wortkombination aus der Schachvariante Crazyhouse und der südamerikanischen Papageiengattung Ara. Die Papageien dieser Gattung haben die Fähigkeit, aus dem Verhalten der Menschen lernen zu können. Dies kann mit einem neuronalen Netz verglichen werden, das mit menschlichen Trainingsdaten trainiert wird.

Seit der Entwicklung von *CrazyAra* wurden zahlreiche Anpassungen und Optimierungen vorgenommen. Dazu zählen unter anderem die Integrationen der ϵ -greedy Exploration, des *Terminal Solvers* und der MCGS. Nähere Informationen dazu können unter anderem aus der Arbeit von Czech et al., 2021 entnommen werden. Außerdem ist *CrazyAra* kompatibel mit dem Universal Chess Interface (UCI) (Kahlen und Muller, 2004).

Mittlerweile bietet *CrazyAra* die Möglichkeit, durch ein abstraktes Interface weitere Spiele außerhalb der Schachwelt einzubetten.

²<https://www.chessprogramming.org/CrazyAra>, besucht am 28.02.2023

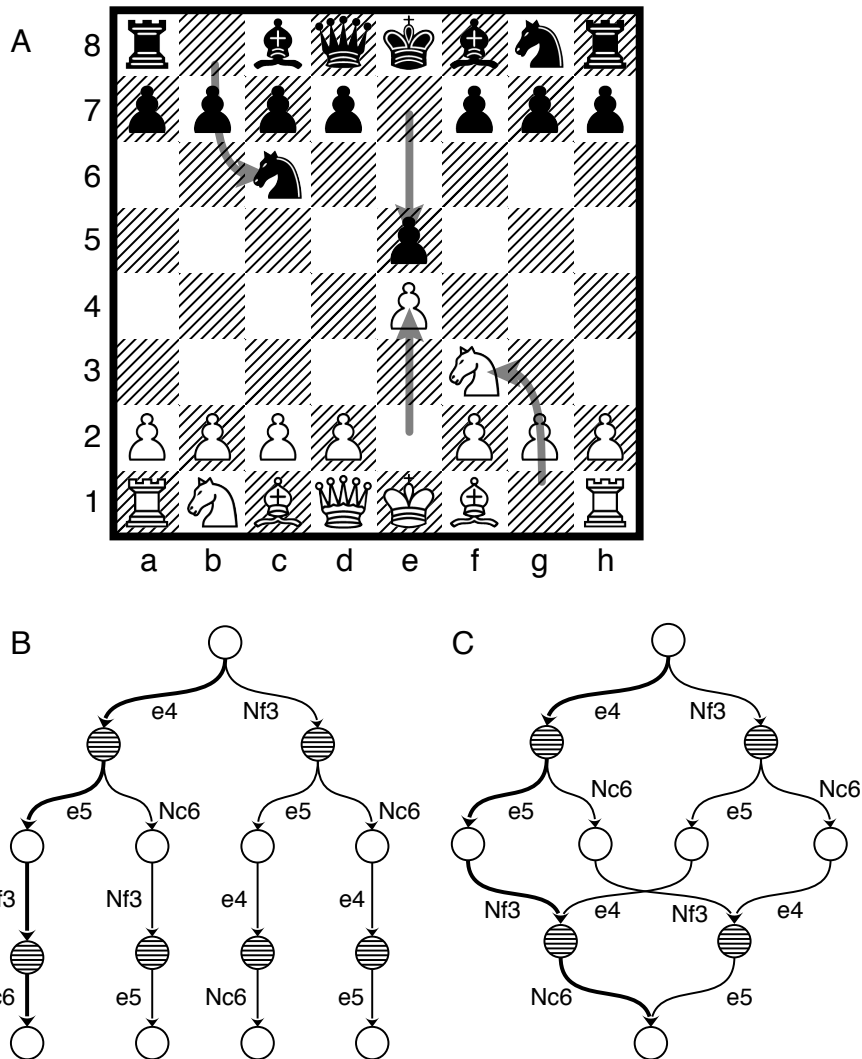


Abbildung 3.1.: Ein Vergleich der MCTS und der MCGS anhand des King's Knight Opening (A). Es ist zu erkennen, dass mehrere Zugfolgen in den dargestellten Zustand führen können. Durch die Graphenrepräsentation (C) ist es möglich, die Anzahl der Knoten im gezeigten Beispiel auf zehn zu reduzieren, statt der 15, die bei der Baumrepräsentation (B) nötig sind. Ein Auszug aus Czech et al., 2021.

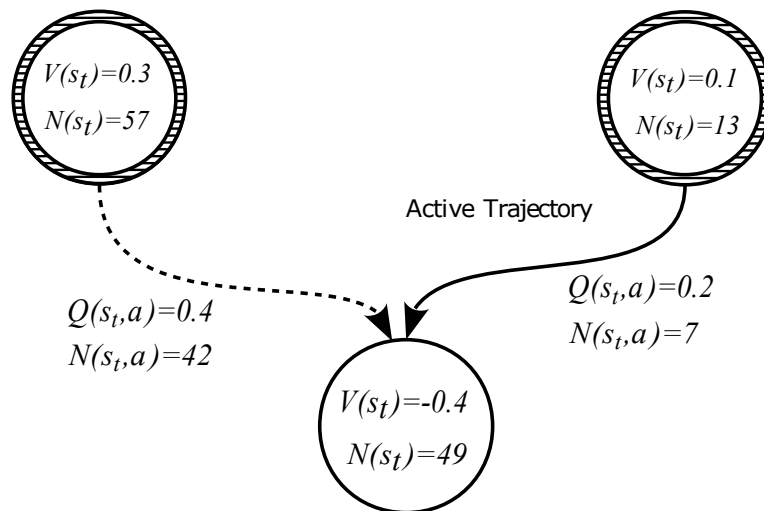


Abbildung 3.2.: Darstellung eines Szenarios, bei dem entlang einer Trajektorie ein Informationsdefizit vorliegt. In einem vorherigen Suchschritt wurde der Zustandswert $V(s_t)$ und der Q-Value $Q(s_t, a)$ des linken Elternknotens, der in die Transposition führt, aktualisiert. Danach hat der Q-Value $Q(s_t, a)$ des rechten Elternknotens, der in die Transposition führt, ein Informationsdefizit, das bei der erneuten Wahl dieser Aktion ausgeglichen werden muss. Ein Auszug aus Czech et al., 2021.

4. Methodik

In diesem Kapitel wird zunächst die Repräsentation der Fakten erläutert. Anschließend wird ein Ansatz eingeführt, durch den die Nutzung der Neuartigkeit von Zuständen mit der Monte-Carlo Graph Search (MCGS) verknüpft werden können.

4.1. Repräsentation der Fakten

Um einen Wert für die Neuartigkeit zu berechnen, werden sowohl für die *Evaluation Novelty* (siehe Abschnitt 3.1.1) als auch für die *Pseudocount Novelty* (siehe Abschnitt 3.1.2) die einzelnen Fakten benötigt. Um diese möglichst effizient traversieren zu können, ist eine geeignete Repräsentation notwendig. Da sich die Fakten aus der aktuellen Brettanordnung inklusive der Taschen (für Crazyhouse) ergeben, liegt es nahe, die Repräsentation so zu wählen, dass eine einfache Indexierung der Fakten aus der Eingaberepräsentation des neuronalen Netzes in *CrazyAra* möglich ist. Dort sind alle relevanten Informationen des aktuellen Zustandes durch mehrere aneinandergereihte Ebenen mit dem Format 8×8 abgebildet. Eine Beschreibung der Eingaberepräsentation in *CrazyAra* für Crazyhouse¹ ist in Tabelle 4.1 zu finden. Pro Figur und pro Spieler gibt es eine eigene Ebene, um alle Figuren dieses Typs auf dem Spielbrett darzustellen. Abbildung 4.1 zeigt eine zusammengefasste Darstellung eines Zustandes. Die Felder, auf denen im betrachteten Zustand eine Figur steht, haben den Wert 1. Im Beispiel aus Abbildung 4.1 steht die weiße Dame auf Feld d1, sodass in der fünften Ebene (dort werden die Position der weißen Dame(n) gespeichert) dem Feld (0,3) der Wert 1 zugewiesen ist (die Zählung der Indizes beginnt immer bei 0). Die Ebenen sind dabei jedoch nicht in Form einer 8×8 -Matrix indiziert, sondern abgeflacht in ein Array der Länge $64 \cdot \text{Anzahl der Ebenen}$. Um darzustellen, dass sich die weiße Dame auf Feld d1 befindet, wird der Wert des Arrays an Index $64 \cdot 4 + 3 = 259$ auf

¹<https://github.com/QueensGambit/CrazyAra/wiki/Input-representation>, besucht am 03.01.2023

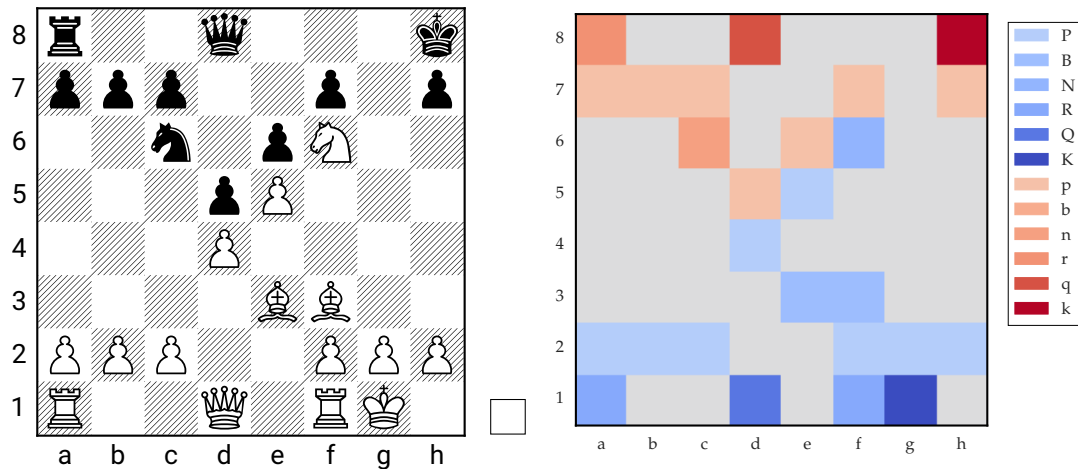


Abbildung 4.1.: Farblich markierte Eingaberepräsentation einer exemplarischen Brettanordnung (FEN: r2q3k/ppp2p1p/2n1pN2/3pP3/3P4/4BB2/PPP2PPP/R2Q1RK1 w - - 4 22). Links befindet sich die klassische Repräsentation des Zustandes und rechts eine farbliche Visualisierung. Ein Auszug aus der Arbeit von Czech et al., 2020.

1 gesetzt. Die Eingaberepräsentation für Schach² (und andere Varianten) hat noch mehr Ebenen, aber die für die Indexierung der Fakten notwendigen ersten 14 Ebenen bleiben gleich.

Wenn für die Repräsentation der Fakten ebenfalls eine Ebenendarstellung verwendet wird, können die Fakten somit direkt aus der Eingaberepräsentation abgelesen werden. Das obige Beispiel hat gezeigt, dass der Fakt „weiße Dame auf d1“ an Index 259 im Array lokalisiert ist. Welcher Wert dem Fakt zugewiesen wird, hängt von der gewählten Metrik für die Neuartigkeit ab. Bei der *Evaluation Novelty* enthält jeder Fakt den besten Wert eines Zustandes, in dem der Fakt bisher aufgetreten ist. Bei der *Pseudocount Novelty* wird für alle Fakten gespeichert, wie oft diese bis zum Zeitschritt t bereits aufgetreten sind.

Bei Crazyhouse ergeben sich durch die Taschenelemente zusätzliche Fakten. Die Fakten der Taschenelemente sind durch den Figurentyp, die Farbe des Spielers und die Anzahl der Figuren dieses Typs in der Tasche bestimmt. „Zwei Bauern in der Tasche von weiß“ stellt beispielsweise einen Fakt dar. Die Faktenrepräsentation der Taschen beider Spieler kann

²<https://github.com/QueensGambit/CrazyAra/pull/134>, besucht am 03.01.2023

daher nicht analog zu *CrazyAra* durch Einzelwertebenen erfolgen (siehe Tabelle 4.1). Dort ist nur die Anzahl der Figuren eines Typs in der Tasche eines Spielers als normalisierter Wert zwischen 0 und 1 gespeichert und auf alle Felder der Ebene kopiert. In dieser Arbeit werden die Ebenen der Fakten so definiert, dass pro Fakt nur ein Eintrag in der zugehörigen Ebene existiert. Dadurch bildet das erste Feld einer solchen Ebene, den Fakt „eine Figur vom Typ p in der Tasche von weiß/schwarz“ ab. Der Wert, der für jeden Fakt f gespeichert wird, hängt erneut von der gewählten Metrik ab. Da bis auf umgewandelte Bauern alle Fakten implizit durch den Zustand auf dem Spielfeld gegeben sind, könnte man möglicherweise auf die Fakten der Taschenfiguren verzichten. In dieser Arbeit wird jedoch der oben beschriebene Ansatz verwendet.

4.2. Neuartigkeit und MCGS

Um gegebenenfalls noch weitere Verbesserungen der Suche zu erzielen, wird in dieser Arbeit der Ansatz verfolgt, Neuartigkeit mit der MCGS zu verknüpfen. Wie von Czech et al., 2021 gezeigt, erzielt MCGS gegenüber Monte-Carlo Tree Search eine bessere Performance, sowohl für Schach als auch für Crazyhouse. Es gibt mehrere Möglichkeiten, die Nutzung der Neuartigkeit von Zuständen in die MCGS zu integrieren. Die in dieser Arbeit verwendete Variante orientiert sich an der MCGS-Erweiterung von *CrazyAra*. Da die Abarbeitung der Suche in Trajektorien erfolgt, werden Informationen nur entlang dieser Trajektorien zurückpropagiert. Um ein Informationsdefizit wie in Abbildung 4.2 dargestellt zu vermeiden, müssen die Neuartigkeitswerte, analog zu den Zustandswerten, auch innerhalb der Knoten gespeichert werden. Für Knoten ohne Informationsdefizit gilt nach der Backpropagation immer

$$M(s_{t+1}) = M(s_t, a) \quad (4.1)$$

für die Aktion a , die von Zustand s_t in den Zustand s_{t+1} führt. Wenn von Zustand s_t über die Aktion a eine Transposition zum ersten Mal gefunden wird, kommt es zu einer Backpropagation des Zustandswertes $V(s_{t+1})$ und des Neuartigkeitswertes $M(s_{t+1})$ der Transposition. Somit kann auch nach der Integration der Neuartigkeitsbewertung der lauffzeitintensive Aufruf des neuronalen Netzes vermieden werden.

Wenn die Selektion auf eine Transposition trifft, hat der Q-Value $Q(s_t, a)$ und der Neuartigkeitswert $M(s_t, a)$ des Zustands-Aktionspaares (s_t, a) , dass in die Transposition s_{t+1} führt, möglicherweise ein Informationsdefizit (wie in Abschnitt 3.3 beschrieben). Um das Informationsdefizit eines Q-Values festzustellen, wird überprüft, ob die Differenz zum

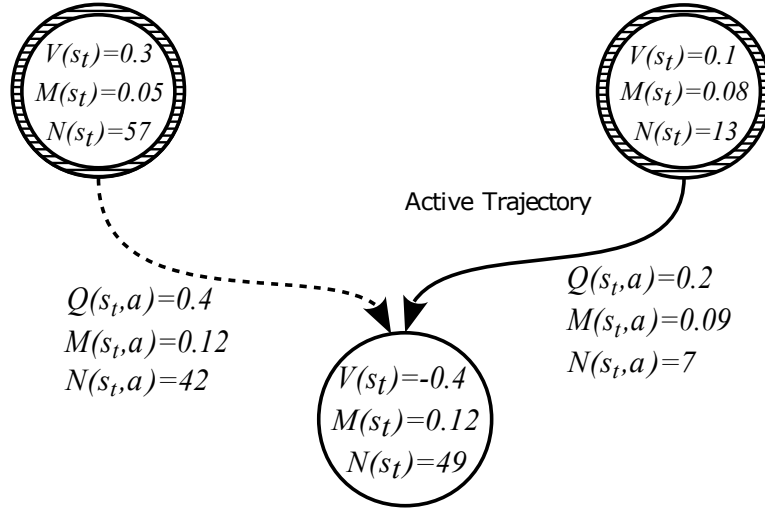


Abbildung 4.2.: Darstellung eines Szenarios, bei dem entlang einer Trajektorie ein Informationsdefizit vorliegt. In einem vorherigen Suchschritt wurde der Zustandswert $V(s_t)$ und der Q-Value $Q(s_t, a)$ des linken Elternknoten, der in die Transposition führt aktualisiert. Danach hat der Q-Value $Q(s_t, a)$ des rechten Elternknoten, der in die Transposition führt, ein Informationsdefizit, das bei der erneuten Wahl dieser Aktion ausgeglichen werden muss. Basierend auf einer Grafik von Czech et al., 2021.

Zielwert eine Schwelle $Q_\epsilon(s, a)$ übersteigt. Der Schwellwert $Q_\epsilon(s, a)$ ist hier ein Hyperparameter. Um die Einführung eines weiteren Hyperparameters zu vermeiden, wird die Entscheidung, ob ein Informationsdefizit vorliegt, allein anhand des Q-Values getroffen. In einem solchen Fall wird neben dem Zustandswert $V(s_t)$ auch der Neuartigkeitswert $M(s_t)$ der Transposition s_t zurückpropagiert. Für die Aktualisierung der Neuartigkeit $M(s_t, a)$ der Aktion a ist die Differenz

$$M_\delta(s_t, a) = M(s_{t+1}) - M(s_t, a) \quad (4.2)$$

zum Zielwert $M(s_{t+1})$ der Transposition s_{t+1} erforderlich. Der zurückpropagierte Korrekturwert $M_\phi(s_t, a)$ ergibt sich aus

$$M_\phi(s_t, a) = N(s_t, a)M_\delta(s_t, a) + M(s_{t+1}). \quad (4.3)$$

Feature	Planes	Comment
P1 piece	6	Pieces are ordered: PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING
P2 piece	6	Pieces are ordered: PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING
Repetitions*	2	Two planes (full zeros/ones) indicating how often the board position has occurred
P1 prisoner count*	5	Pieces are ordered: PAWN, KNIGHT, BISHOP, ROOK, QUEEN (excluding the KING)
P2 prisoner count*	5	Pieces are ordered: PAWN, KNIGHT, BISHOP, ROOK, QUEEN (excluding the KING)
P1 Promoted Pawns Mask	1	Binary map indicating the pieces which have been promoted
P2 Promoted Pawns Mask	1	Binary map indicating the pieces which have been promoted
En-passant square	1	Binary map indicating the square where en-passant capture is possible
Colour*	1	All zeros for black and all ones for white
Total move count*	1	Integer value setting the move count (UCI notation)
P1 castling*	2	One if castling is possible, else zero
P2 castling*	2	One if castling is possible, else zero
No-progress count*	1	Setting the no progress counter as integer values, (described by UCI halfmoves format)
Total	34	

Tabelle 4.1.: Crazyhouse Eingaberepräsentation für das neuronale Netz in *CrazyAra*. P_i steht für den Spieler $i \in \{1,2\}$. Die mit * markierten Felder sind Einzelwerte, die über die ganze Ebene dargestellt werden und normalisiert sind auf Werte im Intervall $[0,1]$. Die Tabelle stammt von <https://github.com/QueensGambit/CrazyAra/wiki/Input-representation>, besucht am 03.12.2022.

5. Empirische Evaluation

Die Evaluation der Spielstärke von Engines, die Neuartigkeit von Zuständen nutzen, erfolgt in zwei Schritten. Zunächst wird eine Hyperparameteroptimierung durchgeführt, um vielversprechende Kombinationen der beiden Parameter α und β zu ermitteln. Anschließend wird die Entwicklung der Elo-Differenz von der besten Kombination über verschiedene Simulationsdauern analysiert. Die Referenz für die Bestimmung der Elo-Differenz bildet die *CrazyAra* Engine mit der Parameterbelegung $\alpha = 0$ und $\beta = 0$. Das entspricht der *CrazyAra* Engine ohne den Einfluss von Neuartigkeit. Alle Ergebnisse sind im Anhang A in tabellarischer Form zu finden.

5.1. Spieleinstellungen

Um die Engines zu vergleichen, wird das frei verfügbare Programm *Cutechess*¹ verwendet. Damit ist es möglich Partien und Turniere zwischen zwei oder mehreren Engines zu spielen und die Elo-Differenz zu bestimmen. Die durch *Cutechess* erzeugten Ergebnisse sind abhängig von den gewählten Eingaben, die im Folgenden kurz erläutert werden. Außerdem werden weitere Evaluationsbedingungen definiert.

Anzahl an Spielen Um eine sinnvolle Abwägung zwischen Aussagekraft der Ergebnisse und Dauer des Tests zu treffen, hat jede Engine bei der Hyperparameteroptimierung mindestens 100 Partien gegen die Referenzengine gespielt. Für den Test der Elo-Entwicklung sollten die Unsicherheiten noch weiter reduziert werden. Aus diesem Grund hat jede Engine im Rundenturnier² zwischen 540 und 900 Spiele gegen die anderen Engines des Turniers gespielt.

¹<https://github.com/cutechess/cutechess>, besucht am 12.01.2023

²<https://www.chess.com/de/terms/rundenturnier>, besucht am 27.02.2023

Eröffnungen Die Partien sind aus verschiedenen Eröffnungen gestartet. Die Wahl von verschiedenen Spieleröffnungen ist wichtig, um sicherzustellen, dass die Engine mit verschiedenen Spielverläufen umgehen kann. Dies steigert einerseits die Aussagekraft der Ergebnisse und hilft andererseits dabei den Vorteil von Weiß zu verringern. Um den Einfluss von letzterem noch weiter zu senken, wird jede Eröffnung pro Aufeinandertreffen zweimal gespielt, jeweils mit getauschten Farben. Für diese Evaluierung wurden die Eröffnungen von Fabian Fichter^{3 4} verwendet.

Anzahl der Knoten und Simulationen Für die Hyperparameteroptimierung wurden pro Zug 800 Knoten ausgewertet und 1600 Simulationen durchgeführt. Für den Test der Elo-Entwicklung sind Engines mit 100, 200, 400, 800 bzw. 1600 Knoten gegeneinander angetreten. Die Anzahl der durchgeführten Simulationen war immer doppelt so hoch wie die Zahl der Evaluationen durch das neuronale Netz. Die Zeitkontrolle wurde auf den hohen Wert von 300 Sekunden gesetzt, damit die Ergebnisse von der Anzahl der durchsuchten Knoten und nicht von der Zeit abhängen. Dadurch können die Resultate besser verglichen werden.

Hardware Die Evaluierung wurde auf einem DGX-Server durchgeführt, der 16 Nvidia V100 SXM3 GPUs und 96 Intel(R) Xeon(R) Platinum 8174 CPUs umfasst.

5.2. Hyperparameteroptimierung

Bevor eine Aussage zur Spielstärke der Engine getroffen werden kann, die Neuartigkeit in den Selektionsalgorithmus integriert, müssen die Hyperparameter α und β optimiert werden. Die Engine spielt mit verschiedenen Kombinationen dieser Variablen gegen die *CrazyAra* Referenzengine. Die Kombinationen von α und β werden durch eine Rastersuche mit exponentieller Punkteverteilung getestet, um eine Abschätzung von den Gebieten zu erhalten, die sinnvolle Ergebnisse erzielen. Der Vorteil der Rastersuche besteht darin, dass der Agent nicht in lokale Maxima hineinflüht und die Spiele parallel ablaufen können. Das abgesuchte Gebiet wird relativ groß gewählt, da die Analyse von Baier und Kaisers, 2021

³Für Schach: <https://github.com/ianfab/books/blob/master/chess.epd>, besucht am 15.01.2023

⁴Für Crazyhouse: <https://github.com/ianfab/books/blob/master/crazyhouse.epd>, besucht am 15.01.2023

zeigt, dass Kombinationen der Parameter mit unterschiedlichen Größenordnungen durchaus die besten Ergebnisse erzielen können. Die einzelnen Kombinationen der Rastersuche sind durch

$$\alpha = 10^p \quad \text{mit} \quad -3 \leq p \leq 2, p \in \mathbb{Z} \quad (5.1)$$

$$\beta = 10^q \quad \text{mit} \quad -3 \leq q \leq 2, q \in \mathbb{Z} \quad (5.2)$$

beschrieben.

5.2.1. Hyperparameter für die *Evaluation Novelty*

Die Ergebnisse der Hyperparameteroptimierung von den Engines, die zur Neuartigkeitsbewertung die *Evaluation Novelty* (siehe Abschnitt 3.1.1) verwendet haben, sind in Abbildung 5.1 zu finden.

Sowohl für Schach als auch für Crazyhouse kann beobachtet werden, dass es einige Parameterkombinationen gibt, mit denen die Referenzengine geschlagen werden konnte. Jedoch erzielen die positiven Parameterkombinationen nur eine Elo-Differenz von maximal 37 für Schach, bzw. maximal 28 für Crazyhouse. Damit liegen diese Ergebnisse noch im von *Cutechess* angegebenen Unsicherheitsintervall, das der zweiten Standardabweichung entspricht. Bei Schach liegt der Mittelwert⁵ der Unsicherheiten bei 56 mit einer Standardabweichung von 6,8 und bei Crazyhouse liegt der Mittelwert bei 48 mit einer Standardabweichung von 7,1. Die Kombinationen der besten Ergebnisse sind $(\alpha|\beta) = (0,001|0,01)$ bei Schach und $(\alpha|\beta) = (0,1|0,001)$ bei Crazyhouse. Das gleiche Resultat erzielt bei Crazyhouse die Kombination $(\alpha|\beta) = (10|0,01)$.

Bei der Betrachtung der Ergebnisse kann ein Muster eindeutig identifiziert werden. Für Werte von $\beta \geq 1$ nimmt die Spielstärke der Engines, die Neuartigkeit von Zuständen nutzen, stark ab. Ab $\beta \geq 10$ spielt die Referenzengine eindeutig besser. Bei Crazyhouse wirken sich diese Ausschläge noch deutlicher aus als bei Schach, was den Erwartungen entspricht. Da Unentschieden bei Schach deutlich häufiger auftreten als bei Crazyhouse, können tendentiell schwächere Engines seltener eine Punkteteilung in Form eines Unentschieden erzielen. Ansonsten kann in den beiden hier betrachteten Evaluationen kein klares Muster erkannt werden.

⁵Für den Mittelwert werden nur Ergebnisse im Bereich $[-55, 55]$ betrachtet, da die Unsicherheit hauptsächlich für geringe Elo-Differenzen von Relevanz ist. Dies gilt auch für die folgenden Abschnitte.

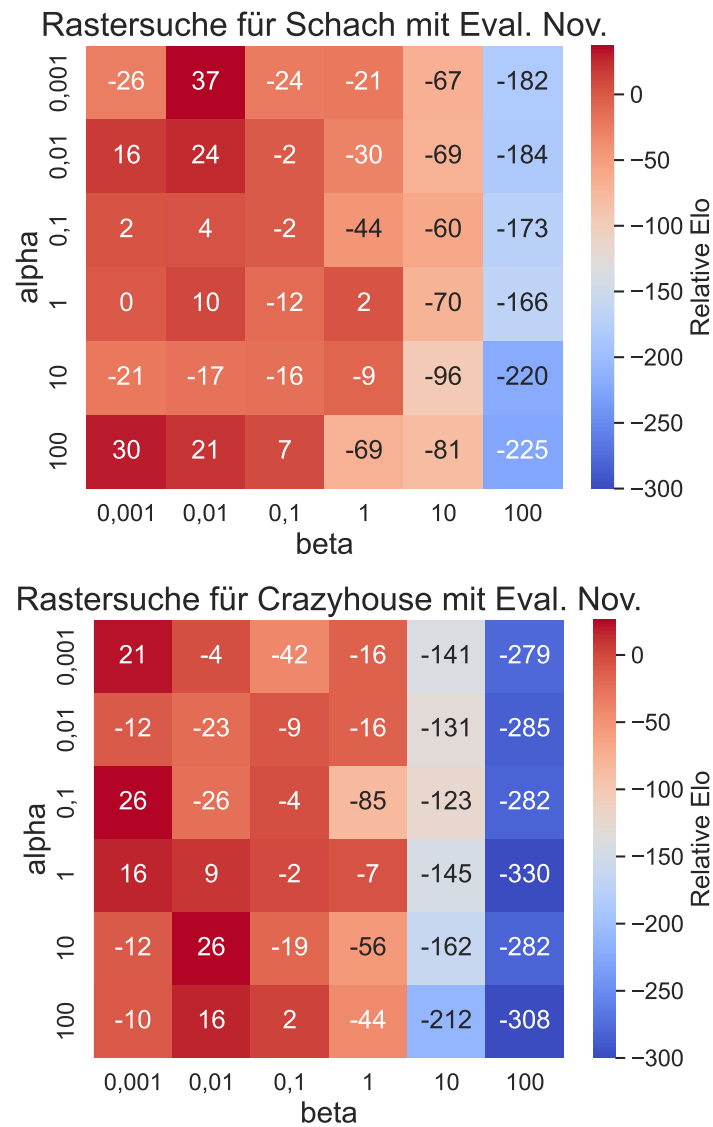


Abbildung 5.1.: Ergebnisse der Rastersuche für verschiedene Hyperparameterkombinationen von α und β bei Schach (oben) und Crazyhouse (unten). Bei dieser Rastersuche wurde die *Evaluation Novelty* verwendet. Die Unsicherheiten der Ergebnisse haben bei Schach einen Mittelwert von 36 mit einer Standardabweichung von 2,7 und bei Crazyhouse einen Mittelwert von 48 mit einer Standardabweichung von 7,1. Die Referenzengine hatte die Parameterkombination $(\alpha|\beta) = (0|0)$.

5.2.2. Hyperparameter für die *Pseudocount Novelty*

Die Ergebnisse der Hyperparameteroptimierung von den Engines, die zur Neuartigkeitsbewertung die *Pseudocount Novelty* (siehe Abschnitt 3.1.2) verwendet haben, sind in Abbildung 5.2 zu finden.

Im Allgemeinen kann beobachtet werden, dass sich die Ergebnisse der Engines mit dieser Neuartigkeitsmetrik nur leicht von denen aus dem letzten Abschnitt unterscheiden. Das beste Ergebnis bei Schach erzielt, mit einer Elo-Differenz von 28, die Engine mit der Parameterkombination $(\alpha|\beta) = (0,001|0,01)$. Die Unsicherheiten liegen hier bei einem Mittelwert von 51 mit einer Standardabweichung von 7,3. Das beste Resultat bei Crazyhouse erzielen die beiden Kombinationen $(\alpha|\beta) = (1|0,001)$ sowie $(\alpha|\beta) = (0,1|0,1)$, ebenfalls mit einer Elo-Differenz von 28.

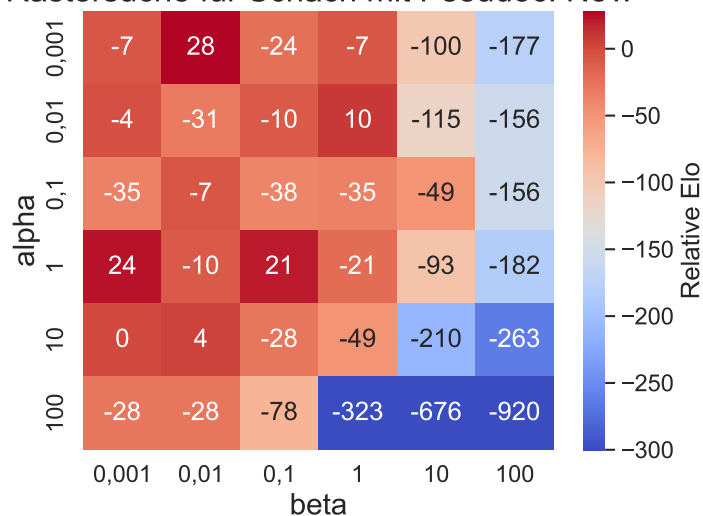
Es kann analog zum vorherigen Abschnitt ein signifikanter Zusammenhang der Ergebnisse mit Werten von $\beta \geq 10$ festgestellt werden. Ab diesem Wert spielen die Engines, die Neuartigkeit von Zuständen nutzen, eindeutig schlechter als die Referenzengine. Bei dieser Neuartigkeitsmetrik kann zusätzlich eine Tendenz für die unterschiedlichen Werte von α identifiziert werden. Ab $\alpha \geq 10$ ist eine Abnahme der Spielstärke zu erkennen, die sehr stark für Kombinationen mit großen α - und β -Werten auftritt. Bei der Kombination $(\alpha|\beta) = (100|100)$ kann die Engine bei Schach und Crazyhouse kein Spiel gegen die Referenzengine gewinnen.

5.2.3. Hyperparameter für Neuartigkeit bei der MCGS

Die Ergebnisse der Hyperparameteroptimierung von den Engines, die Monte-Carlo Graph Search (MCGS) mit der Neuartigkeitsbewertung kombinieren (siehe Abschnitt 4.2), sind in Abbildung 5.3 zu finden. Als Neuartigkeitsmetrik wurde hier die *Evaluation Novelty* verwendet.

Die Ergebnisse zeigen ein ähnliches Bild wie in den vorherigen Abschnitten. Einige Engines, die Neuartigkeit in die MCGS integriert haben, schneiden etwas besser ab als die Referenzengine. Das beste Resultat bei Schach erzielt die Engine mit der Parameterkombination $(\alpha|\beta) = (1|0,01)$ mit einer Elo-Differenz von 35. Die Unsicherheiten liegen dafür im Mittel bei 51 mit einer Standardabweichung von 7,5. Bei Crazyhouse schneidet die beste Engine mit einer Elo-Differenz von 53 ein wenig besser ab. Die Parameterkombination dabei ist $(\alpha|\beta) = (0,001|0,001)$ und die Unsicherheiten haben einen Mittelwert von 69 mit einer Standardabweichung von 13,4.

Rastersuche für Schach mit Pseudoc. Nov.



Rastersuche für Crazyhouse mit Pseudoc. Nov.

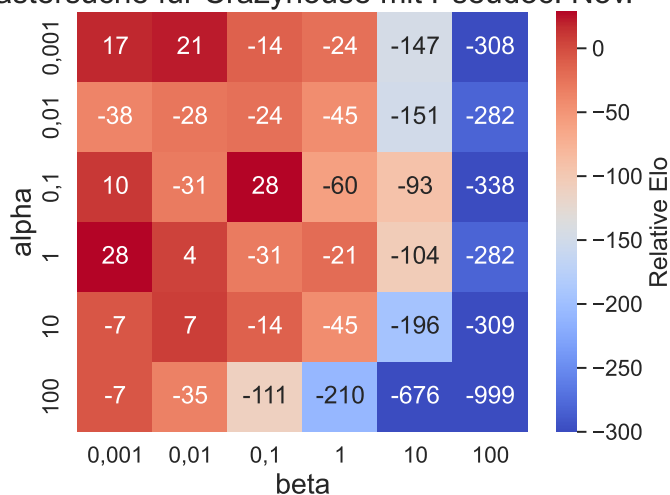


Abbildung 5.2.: Ergebnisse der Rastersuche für verschiedene Hyperparameterkombinationen von α und β bei Schach (oben) und Crazyhouse (unten). Bei dieser Rastersuche wurde die *Pseudocount Novelty* verwendet. Die Unsicherheiten der Ergebnisse haben bei Schach einen Mittelwert von 51 mit einer Standardabweichung von 7,3 und bei Crazyhouse einen Mittelwert von 68 mit einer Standardabweichung von 11,8. Die Referenzengine hatte die Parameterkombination $(\alpha|\beta) = (0|0)$.

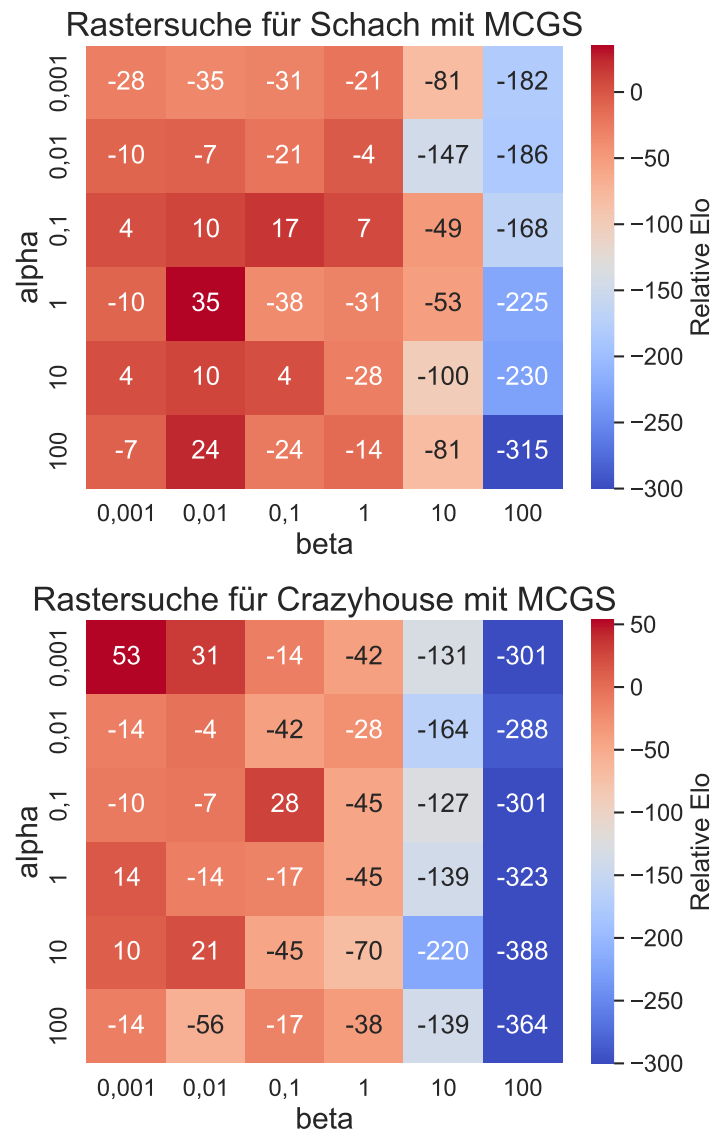


Abbildung 5.3.: Ergebnisse der Rastersuche für verschiedene Hyperparameterkombinationen von α und β bei Schach (oben) und Crazyhouse (unten). Bei dieser Rastersuche wurde die MCGS verwendet und die *Evaluation Novelty* als Neuartigkeitsmetrik. Die Unsicherheiten der Ergebnisse haben bei Schach einen Mittelwert von 51 mit einer Standardabweichung von 7,5 und bei Crazyhouse einen Mittelwert von 69 mit einer Standardabweichung von 13,4. Die Referenzengine hatte die Parameterkombination $(\alpha|\beta) = (0|0)$.

Analog zu den Ergebnissen aus den vorherigen Abschnitten spielen die Engines ab einem Wert von $\beta \geq 10$ deutlich schlechter als die Referenzengine. Die Ergebnisse sind erwartungsgemäß bei Crazyhouse noch signifikanter als bei Schach. Es kann außerdem beobachtet werden, dass für Werte von $\alpha \leq 0,01$ die Engines bei Schach stets negative Elo-Differenzen aufweisen.

5.3. Elo-Entwicklung

In diesem Abschnitt soll die Spielstärke der Engines relativ zu der Anzahl an Evaluationen des neuronalen Netzes ausgewertet werden. Die Evaluation erfolgt durch ein Rundenturnier mit fünf Engines, die Neuartigkeit von Zuständen nutzen und fünf Referenzengines. Die Anzahl der Evaluationen des neuronalen Netzes pro Zug ist mit 100, 200, 400, 800, 1600 auf die fünf Engines jeder Klasse verteilt. Dadurch ist es möglich abzuschätzen, wie gut der Einfluss der Neuartigkeitsbewertung sich über die verschiedenen Auswertungsintervalle entwickelt. Für die Hyperparameter α und β wurde jeweils das beste Ergebnis aus der Hyperparameteroptimierung verwendet. Falls zwei Kombinationen in der Rastersuche das gleiche Ergebnis erzielt haben, wurde zufällig eine Kombination ausgewählt.

5.3.1. Elo-Entwicklung für die *Evaluation Novelty*

Die Ergebnisse des Rundenturniers von den Engines, die zur Neuartigkeitsbewertung die *Evaluation Novelty* (siehe Abschnitt 3.1.1) verwendet haben, sind in Abbildung 5.4 zu finden. Die Kombinationen der Hyperparameter sind bei Schach $(\alpha|\beta) = (0,01|0,001)$ und bei Crazyhouse $(\alpha|\beta) = (0,1|0,001)$.

Für Schach und Crazyhouse kann beobachtet werden, dass kein signifikanter Unterschied im Verlauf der Kurven vorhanden ist. Für einige Evaluationsanzahlen schneidet die modifizierte Engine leicht besser ab, jedoch ist kein Muster zu erkennen. Insgesamt kann keine Verbesserung der Spielstärke durch die Hinzunahme von *Evaluation Novelty* in den Selektionsalgorithmus festgestellt werden.

5.3.2. Elo-Entwicklung für die *Pseudocount Novelty*

Die Ergebnisse des Rundenturniers von den Engines, die zur Neuartigkeitsbewertung die *Pseudocount Novelty* (siehe Abschnitt 3.1.2) verwendet haben, sind in Abbildung 5.5 zu

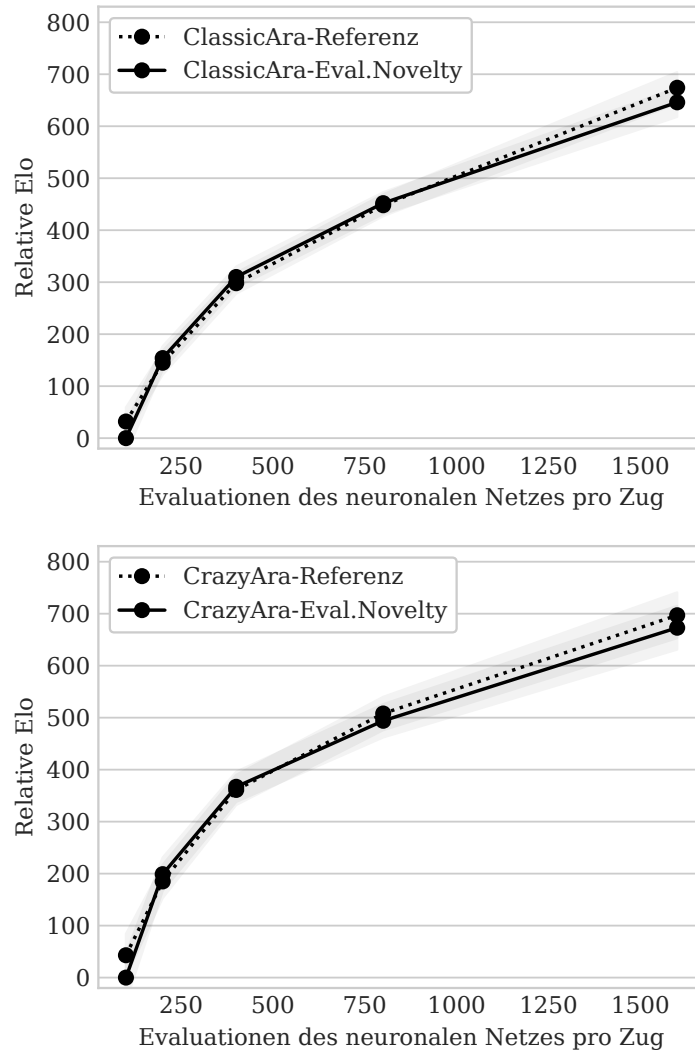


Abbildung 5.4.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Schach (oben) und Crazyhouse (unten). Als Neuartigkeitsmetrik wurde die *Evaluation Novelty* verwendet. Die Hyperparameterkombinationen sind $(\alpha|\beta) = (0,01|0,001)$ für Schach und $(\alpha|\beta) = (0,1|0,001)$ für Crazyhouse. Die Unsicherheit ist durch die schattierten Bereiche repräsentiert.

finden. Die Kombinationen der Hyperparameter sind bei Schach $(\alpha|\beta) = (0,001|0,01)$ und bei Crazyhouse $(\alpha|\beta) = (1|0,001)$.

Ähnlich zu den Ergebnissen aus dem vorherigen Abschnitt kann kein deutliches Muster erkannt werden. Somit kann auch für die *Pseudocount Novelty* im Vergleich zur Referenz keine Verbesserung der Spielstärke festgestellt werden.

5.3.3. Elo-Entwicklung für die *Evaluation Novelty* bei MCGS

Die Ergebnisse des Rundenturniers von den Engines, die MCGS mit der Neuartigkeitsbewertung kombinieren (siehe Abschnitt 4.2), sind in Abbildung 5.6 zu finden. Als Neuartigkeitsmetrik wurde hier die *Evaluation Novelty* verwendet. Die Kombinationen der Hyperparameter sind bei Schach $(\alpha|\beta) = (1|0,01)$ und bei Crazyhouse $(\alpha|\beta) = (0,001|0,001)$.

Die Kurve der Elo-Differenzen verlaufen für die Referenzengine und die modifizierte Engine sehr ähnlich. Somit entsprechen die Resultate dieser Auswertung denen aus den vorherigen Abschnitten. Es können keine signifikanten Muster wahrgenommen werden.

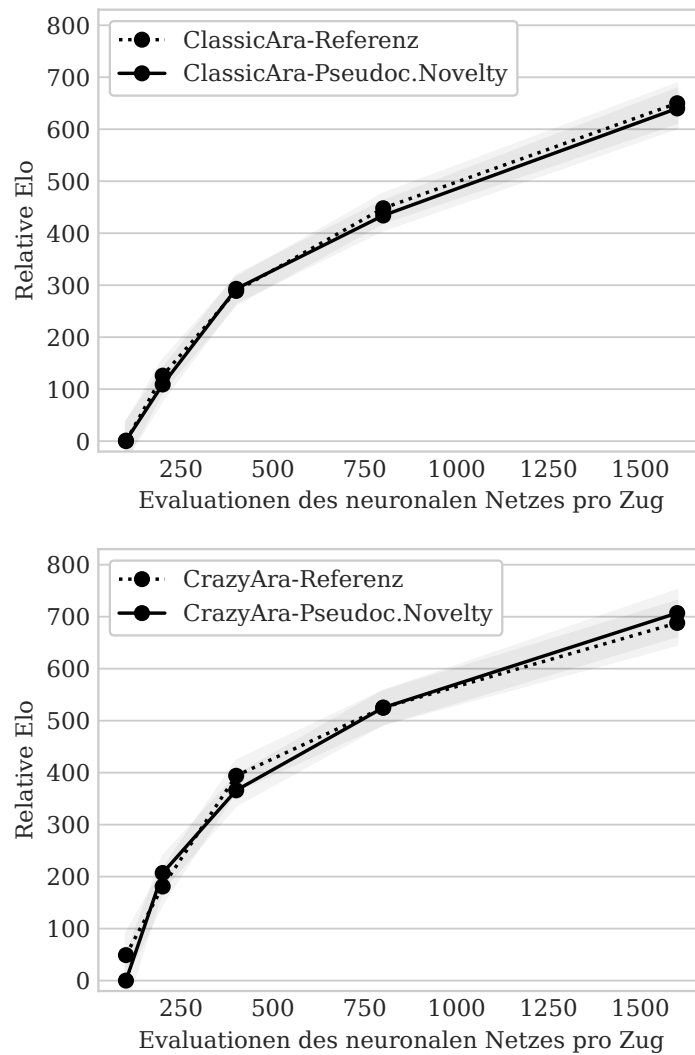


Abbildung 5.5.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Schach (oben) und Crazyhouse (unten). Als Neuartigkeitsmetrik wurde die *Pseudocount Novelty* verwendet. Die Hyperparameterkombinationen sind $(\alpha|\beta) = (0,001|0,01)$ für Schach und $(\alpha|\beta) = (1|0,001)$ für Crazyhouse. Die Unsicherheit ist durch die schattierten Bereiche repräsentiert.

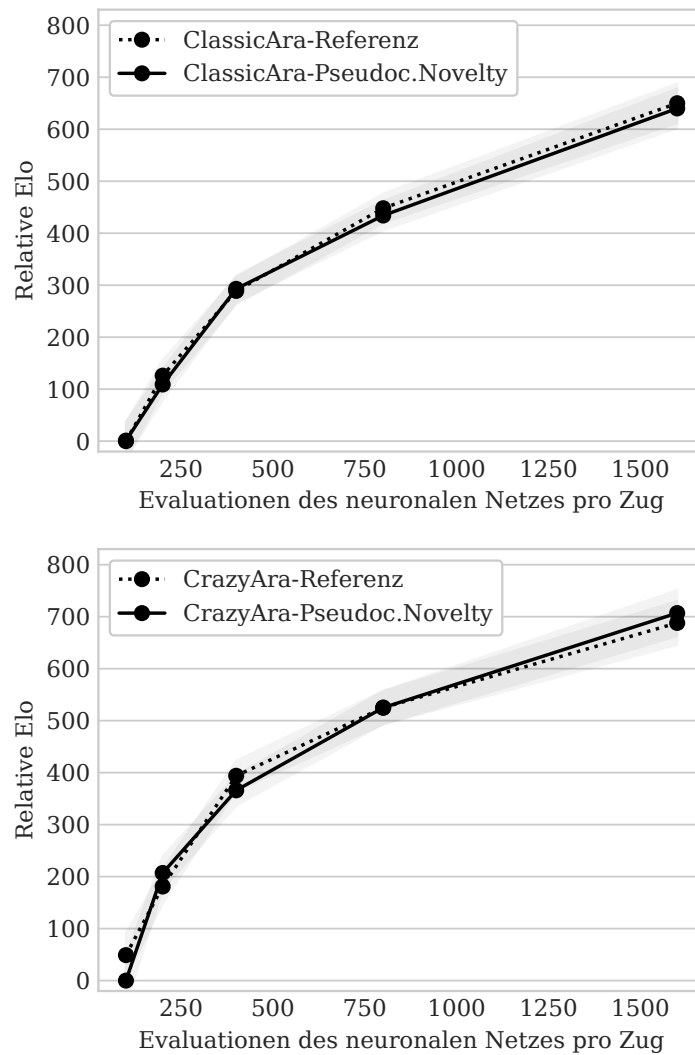


Abbildung 5.6.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Schach (oben) und Crazyhouse (unten). Für die Suche wurde die MCGS verwendet und als Neuartigkeitsmetrik die *Evaluation Novelty*. Die Hyperparameterkombinationen sind $(\alpha|\beta) = (1|0,01)$ für Schach und $(\alpha|\beta) = (0,001|0,001)$ für Crazyhouse. Die Unsicherheit ist durch die schattierten Bereiche repräsentiert.

6. Diskussion

In diesem Kapitel werden die Resultate der Evaluation diskutiert. Dabei wird untersucht, welche Einflussfaktoren es auf die Ergebnisse geben kann. Außerdem erfolgt der Vergleich der Resultate mit denen von Baier und Kaisers, 2021. Zuletzt werden mögliche Fehlerquellen bei der Ergebniserzeugung dargelegt.

Insgesamt kann festgehalten werden, dass die Nutzung der Neuartigkeit von Zuständen bei Schach und Crazyhouse in dieser Arbeit keine signifikant positiven Ergebnisse erzielen konnte. In den sechs durchgeführten Rundenturnieren konnte keine der modifizierten Engines die Referenzengine eindeutig schlagen. Andererseits führte der Einsatz von Neuartigkeit mit optimierten Hyperparametern nicht zu einer deutlichen Abnahme der relativen Elo gegenüber der Referenzengine. Es kann beobachtet werden, dass die Spielstärke sensibler auf die Wahl des Verfallsparameters β reagiert, als auf den Belohnungsfaktor α . Für Werte von $\beta \geq 1$ spielen die meisten Engines signifikant schlechter als die Referenzengine. Wenn $\beta \leq 0,1$ ist, zeigt sich ein relativ ausgeglichenes Ergebnis. Bei der Wahl von α weist die relative Elo nur bei der *Pseudocount Novelty* eine gewisse Sensitivität auf. Große Werte von α führen hier ebenfalls zu einer Abnahme der relativen Elo.

Diese Beobachtungen stehen im Gegensatz zu den Ergebnissen von Baier und Kaisers, 2021, bei denen die Nutzung der Neuartigkeit von Zuständen bei 14 von 24 Auswertungen signifikant positive Resultate erzielte. Dort wurden die Spiele Vier-Gewinnt, Othello, Breakthrough sowie Knightthrough untersucht. Die möglichen Gründe dafür sind vielseitig und werden im Folgenden näher beleuchtet.

Zunächst kann es sein, dass die Nutzung von Neuartigkeit nicht für alle Domänen Vorteile bringt. So kann man bei Baier und Kaisers, 2021 sehen, dass bei Othello nur in einem von sechs Testfällen signifikant positive Ergebnisse erzielt werden. Im Gegensatz dazu erzielen fünf von sechs Tests bei Vier-Gewinnt und Knightthrough eindeutig positive Resultate. Es ist also möglich, dass in den Domänen Schach und Crazyhouse die Nutzung von Neuartigkeit keinen positiven Beitrag leistet.

Des Weiteren hängt die Neuartigkeitsbewertung von der Wahl der Merkmale ab, die für die Berechnung der Neuartigkeit verwendet werden. In dieser Arbeit wurden die atomaren Fakten F als Merkmale gewählt. Möglicherweise haben diese zu wenig inhaltliche Aussagekraft, sodass die Neuartigkeit nicht für eine verstärkte Gewichtung von wirklich vielversprechenden Aktionen sorgt. Für Schach könnten bedeutsamere Merkmale zum Beispiel „der schwarze König kann sich bewegen“ oder „weiß kontrolliert die H-Linie“ sein. Durch den Einsatz von aussagekräftigeren Merkmalen, könnten Aktionen stärker gewichtet werden, die in der Zukunft potentiell mehr Belohnung erzielen. Hier stellt sich jedoch die Frage, warum bei den von Baier und Kaisers, 2021 evaluierten Spielen durch die Verwendung von atomaren Fakten zur Neuartigkeitsbewertung positive Resultate erzielt werden konnten. Eine potentielle Erklärung dafür könnte in der bereits angesprochenen Domänenspezifität liegen. Die dort evaluierten Spiele sind weniger komplex als Schach und Crazyhouse. Dadurch liefert die Verwendung von atomaren Fakten bei diesen Spielen eventuell ausreichend Informationen um wirklich vielversprechende Aktionen zu belohnen.

Eine weitere Möglichkeit, warum die Ergebnisse dieser Arbeit, im Gegensatz zu denen von Baier und Kaisers, 2021, keine positiven Effekte durch die Nutzung von Neuartigkeit zeigen, resultiert aus der Wahl des Selektionsalgorithmus. Hier wurde der PUCT-Selektionsalgorithmus um einen Neuartigkeitsfaktor erweitert. Bei Baier und Kaisers, 2021 wurde die UCT-Formel angepasst. Im Unterschied zu UCT lässt PUCT die Policy-Verteilung in den Explorationsfaktor einfließen, sodass potentiell vielversprechende Aktionen bevorzugt werden. Es ist jedoch schwierig zu sagen, ob die verschiedenen Explorationsfaktoren einen signifikanten Einfluss auf den Unterschied der Ergebnisse dieser Arbeit gegenüber denen von Baier und Kaisers, 2021 haben. Wenn die durch das neuronale Netz ermittelte Policy-Verteilung bereits einen treffenden Prior für vielversprechende Folgeaktionen darstellt, führt die Nutzung von Neuartigkeit wie sie hier verwendet wurde, eventuell nicht zu einer Verbesserung der Exploration.

In der Arbeit von Baier und Kaisers, 2021 wurde eine lineare heuristische Evaluierungsfunktion zur Ermittlung der Zustandswerte verwendet. Im Gegensatz dazu, verwendet die *CrazyAra* Engine ein tiefes neuronales Netz, was bei gutem Training zu besseren Zustandsevaluationen als bei linearen Heuristiken führen sollte. Das liegt daran, dass Spiele wie Schach oder Crazyhouse zu komplex sind, um sie durch einen linearen Zusammenhang zu beschreiben. Wie auch von Baier und Kaisers, 2021 diskutiert, müsste dadurch die Güte der Neuartigkeitsbewertung verbessert werden, da die Neuartigkeitsmetriken direkt oder indirekt von der Zustandsevaluation abhängen. Trotzdem besteht die Möglichkeit, dass die Qualität der Selektion abnimmt. Das liegt an der Art, wie die Neuartigkeit in den Selektionsalgorithmus integriert ist. Durch den Faktor b in der Gleichung 3.8 wird zwischen der Neuartigkeit und dem Q-Value eines Zustands-Aktionspaares gewichtet.

Umso größer der Wert von b , desto mehr wird der Q-Value vernachlässigt und der Neuartigkeitswert gewichtet. Das führt möglicherweise zu einer Verschlechterung der Selektion, da die Gewichtung zwischen der Neuartigkeit und dem Q-Value ursprünglich für die Evaluierung mit der *Rapid Action Value Estimation* (RAVE) eingeführt wurde (Gelly und Silver, 2007). Bei dem RAVE-Algorithmus werden Evaluierungen von Aktionen zwischen allen Subbäumen geteilt. Somit ist eine schnelle und grobe Evaluierung möglich, die jedoch ungenauer als die Evaluierung bei klassischer Monte-Carlo Tree Search ist (Gelly und Silver, 2011). Durch die relativ hohe Güte der Evaluierung durch das neuronale Netz ist die anfänglich schwächere Gewichtung des Q-Values bei der Selektion eventuell nicht zielführend. Dieser Erklärungsansatz liefert eine mögliche Begründung, warum die relative Elo sensitiver auf große Werte von β als von α reagiert. Ein hohes β führt zu einer langsameren Abnahme von b und somit ist der Einfluss des Q-Values zu Beginn stärker gedämpft. Stattdessen könnte eine andere Art der Integration der Neuartigkeit in die PUCT-Formel zu besseren Ergebnissen führen. Die Neuartigkeit könnte beispielsweise in den U-Value integriert oder als ganz neuer Faktor hinzugefügt werden.

Im Folgenden wird untersucht, welche Ungenauigkeiten bei der Ergebnisgenerierung vorliegen. Eine mögliche Problematik der Ergebnisse liegt in den Unsicherheiten. Mehrere Elo-Differenzen der Hyperparameterevaluationen haben ein Unsicherheitsintervall, dass größer ist als die Elo-Differenz, sodass die realen Spielstärken anders sein können. Durch eine höhere Anzahl an Partien können die Unsicherheiten verringert werden. Außerdem ergeben sich etwas schlechtere Resultate bei den Rundenturnieren im Vergleich zu den Partien des Hyperparametertests für 800 Evaluationen des neuronalen Netzes. In keinem der Rundenturniere erreicht die modifizierte Engine die Elo-Differenz aus dem Hyperparametertest. Es besteht zwar die Möglichkeit, dass dies aufgrund der angesprochenen Unsicherheiten passiert, jedoch ist es unwahrscheinlich, dass dies bei allen sechs Rundenturnieren vorkommt. Ein naheliegenderer Grund sind die unterschiedlichen Spielarten. Bei den Rundenturnieren wird die Elo-Differenz aus den Spielen gegen mehrere Engines ermittelt, während bei den Hyperparametertests nur ein Gegner vorhanden ist. Daher ist es wahrscheinlich, dass die modifizierte Engine bei einigen Parameterkombinationen von α und β bei 800 Evaluationen des neuronalen Netzes bessere Ergebnisse in Einzelpartien gegen die Referenzengine erzielt, als bei Rundenturnieren gegen verschiedene Engines.

Ein weiterer Faktor, der die Ergebnisse verzerren kann, resultiert aus der Verarbeitung der Trajektorien in Mini-Batches bei *CrazyAra*. Durch die Batch-Abarbeitung können einige Aspekte der Suche parallelisiert werden, da nicht nach jeder Trajektorienerstellung das neuronale Netz aufgerufen werden muss. Um die Wahrscheinlichkeit zu reduzieren, dass ein Knoten, der innerhalb eines Mini-Batches bereits ausgewählt wurde, noch einmal selektiert wird, werden alle selektierten Aktionen mit einem virtuellen Verlust der Q-Values

belegt. Dieser Verlust wird bei der Backpropagation der Trajektorien des Mini-Batches wieder herausgerechnet. In dieser Arbeit wurde kein virtueller Verlust für die Neuartigkeit innerhalb einer Mini-Batch-Verarbeitung implementiert. Folglich wird die Selektion im Verlaufe einer Mini-Batch-Erstellung verzerrt. Durch den Verfall der Neuartigkeitsgewichtung nimmt auch der Einfluss dieses Phänomens über die Zeit ab.

Zuletzt muss festgehalten werden, dass die Wahl von mehr Hyperparameterkombinationen bei der Rastersuche eventuell zu besseren Ergebnissen führen kann. Aufgrund der ähnlichen Resultate bei allen sechs Hyperparameter-tests, ist es jedoch unwahrscheinlich, dass eine Kombination innerhalb des betrachteten Gebietes zu signifikant anderen Ergebnissen führt. Zudem ist die Wahl von Kombinationen außerhalb des Gebietes nicht vielversprechend, da noch kleinere Werte für die Parameter keine deutliche Abweichung zur Referenzengine mehr erzeugen können. Größere Werte für die Parameter bieten ebenfalls keine Evidenz für bessere Ergebnisse.

7. Fazit

In diesem Kapitel werden die Ziele und Ergebnisse dieser Arbeit zusammengefasst und es wird ein Ausblick für zukünftige Arbeiten gegeben.

7.1. Zusammenfassung

In dieser Ausarbeitung wurde die Nutzung der Neuartigkeit von Zuständen bei den Spielen Schach und Crazyhouse getestet. Es wurde untersucht, ob die Hinzunahme einer Neuartigkeitsbewertung in den Selektionsalgorithmus der Monte-Carlo Tree Search zu besseren Ergebnissen führt. Dazu wurden zwei Neuartigkeitsmetriken aus der Arbeit von Baier und Kaisers, 2021 verwendet, die über eine Linearkombination in die PUCT-Formel integriert wurden. Zur Steuerung zwischen der Neuartigkeit und dem Q-Value wurde ein Gewichtungsfaktor benutzt, der mit zunehmender Simulationsdauer die Neuartigkeit vernachlässigt. Dadurch werden die Konvergenzeigenschaften der Suche erhalten. Außerdem wurde untersucht, ob die Nutzung der Neuartigkeit von Zuständen bei der Monte-Carlo Graph Search zu positiven Resultaten führt. Der Ansatz zur Nutzung der Neuartigkeit von Zuständen wurde zur Evaluation in die *CrazyAra* Engine integriert.

Um die Ergebnisse auszuwerten, wurde zunächst eine Optimierung der beiden eingeführten Hyperparameter α und β durchgeführt. Der Parameter α beeinflusst die Belohnung für einen neuartigen Zustand und der Parameter β steuert die Abnahme der Neuartigkeitsgewichtung bei der Selektion. Die Hyperparameteroptimierung wurde in Form einer Rastersuche für verschiedene Größenordnungen von α und β durchgeführt. Anschließend wurden für die besten Kombinationen aus den Hyperparametertests die Elo-Entwicklung für verschieden viele Evaluationen des neuronalen Netzes pro Zug ausgewertet.

Es konnte insgesamt keine signifikante Verbesserung der Spielstärke durch die Hinzunahme der Neuartigkeit in die Suche bei Schach und Crazyhouse festgestellt werden. Die

Entwicklung der relativen Elo über verschiedene Simulationsdauern führte zu ähnlichen Ergebnissen, wie bei der Referenzengine. Es kann festgehalten werden, dass die langsame Abnahme der Neuartigkeitsgewichtung durch einen hohen Wert von β in allen Evaluationen zu deutlich schlechteren Ergebnissen geführt hat. Anschließend wurde diskutiert, was potentielle Gründe für die gegensätzlichen Resultate dieser Arbeit gegenüber denen von Baier und Kaisers, 2021 waren.

7.2. Zukünftige Arbeit

Um herauszufinden, ob die Nutzung von Neuartigkeit bei Schach und Crazyhouse prinzipiell keine Verbesserung bringen kann, oder ob bestimmte Einflussfaktoren verändert werden müssen, ist zukünftige Arbeit nötig.

Eine Möglichkeit zur Verbesserung der Ergebnisse besteht darin, andere Merkmale zu wählen. Um strategisch bedeutungsvolle Merkmale für Schach und Crazyhouse festzulegen, kann Domänenwissen genutzt werden. Solche Merkmale könnten beispielsweise „der schwarze König kann sich bewegen“ oder „weiß kontrolliert die H-Linie“ sein. Andererseits können die Merkmale durch ein neuronales Netz bestimmt werden, indem zum Beispiel ein Autoencoder trainiert wird, der aus Zuständen latente Merkmalsvektoren ermittelt. Durch diese beiden Ansätze kann die Aussagekraft der Merkmale gesteigert werden, was eventuell zu besseren Neuartigkeitsbewertungen führt. Das Prinzip, einen Autoencoder zur Generierung des Merkmalsvektors zu nutzen, könnte im Erfolgsfall auf andere Domänen übertragen werden.

Außerdem könnten Verbesserungen des Einsatzes von Neuartigkeit bei Schach und Crazyhouse durch neue Metriken zur Berechnung der Neuartigkeit erreicht werden. Zudem wären weitere Varianten, die Neuartigkeit in die PUCT-Selektion einbinden, interessant. Wenn die Neuartigkeit in den U-Value integriert oder als ganz neuer Faktor hinzugefügt wird, kann die anfänglich schwächere Gewichtung der Q-Values vermieden werden. Dies ist eventuell für Evaluierungsfunktionen mit hoher Güte von Vorteil.

Danksagung Zunächst dankt der Author Johannes Czech für seine geduldige und ziel-führende Betreuung dieser Arbeit. Er dankt außerdem Moritz Willig, Alena Beyer und in besonderem Maße Johannes Czech für die Entwicklung, Veröffentlichung und Wartung der *CrazyAra* Engine. Des Weiteren dankt der Author Ilari Pihlajisto, Arto Jonsson und den weiteren Entwicklern von *Cutechess* für die Bereitstellung einer stabilen Testumgebung für Multivarianten Schachengines. Die Arbeit profitiert außerdem von der Zusammenstellung von bekannten Eröffnungen für Schach und Crazyhouse von Fabian Fichter. Sein Dank richtet sich zudem an das *Artificial Intelligence and Machine Learning Lab* der TU Darmstadt für die Unterstützung dieser Arbeit und den Zugang zu ihrem DGX-Server. Zuletzt dankt der Author allen Korrekturlesern dieser Arbeit für ihr wertvolles Feedback.

Literatur

- Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47, 235–256. <https://doi.org/https://doi.org/10.1023/A:1013689704352>
- Baier, H., & Kaisers, M. (2021). Novelty and MCTS. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 1483–1487. <https://doi.org/10.1145/3449726.3463217>
- Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., & Colton, S. (2012). A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1), 1–43. <https://doi.org/10.1109/TCIAIG.2012.2186810>
- Campbell, M., Hoane, A., & Hsu, F.-h. (2002). Deep Blue. *Artificial Intelligence*, 134(1), 57–83. [https://doi.org/https://doi.org/10.1016/S0004-3702\(01\)00129-1](https://doi.org/https://doi.org/10.1016/S0004-3702(01)00129-1)
- Czech, J., Korus, P., & Kersting, K. (2021). Improving AlphaZero Using Monte-Carlo Graph Search. *Proceedings of the International Conference on Automated Planning and Scheduling*, 31(1), 103–111. <https://ojs.aaai.org/index.php/ICAPS/article/view/15952>
- Czech, J., Willig, M., Beyer, A., Kersting, K., & Fürnkranz, J. (2020). Learning to Play the Chess Variant Crazyhouse Above World Champion Level With Deep Neural Networks and Human Data. *Frontiers in Artificial Intelligence*, 3. <https://doi.org/10.3389/frai.2020.00024>
- Gelly, S., & Silver, D. (2007). Combining Online and Offline Knowledge in UCT. *Proceedings of the 24th International Conference on Machine Learning*, 273–280. <https://doi.org/10.1145/1273496.1273531>
- Gelly, S., & Silver, D. (2011). Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence*, 175(11), 1856–1875. <https://doi.org/https://doi.org/10.1016/j.artint.2011.03.007>
- Kahlen, S.-M., & Muller, G. H. (2004). UCI Protocol [Accessed: 2023-02-16].

-
- Lehman, J., & Stanley, K. O. (2011). Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2), 189–223. https://doi.org/10.1162/EVCO_a_00025
- Martin, J., Sasikumar, S. N., Everitt, T., & Hutter, M. (2017). Count-based exploration in feature space for reinforcement learning. *arXiv preprint arXiv:1706.08090*.
- Rosin, C. D. (2011). Multi-armed bandits with episode context. *Ann Math Artif Intell*, 61(1), 203–230. <https://doi.org/https://doi.org/10.1007/s10472-011-9258-6>
- Saffidine, A., Cazenave, T., & Méhat, J. (2012). UCD : Upper confidence bound for rooted directed acyclic graphs [A Special Issue on Artificial Intelligence in Computer Games: AICG]. *Knowledge-Based Systems*, 34, 26–33. <https://doi.org/https://doi.org/10.1016/j.knosys.2011.11.014>
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. <https://doi.org/10.48550/ARXIV.1712.01815>
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419), 1140–1144. <https://doi.org/10.1126/science.aar6404>
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550, 354–359. <https://doi.org/https://doi.org/10.1038/nature24270>
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (Second). The MIT Press. <http://incompleteideas.net/book/the-book-2nd.html>
- Tot, M., Conserva, M., Devlin, S., & Liebana, D. P. (2022). Making Something Out of Nothing: Monte Carlo Graph Search in Sparse Reward Environments.

A. Anhang

Ergebnisse der Rastersuche

Variant	Alpha	Beta	Elo-Diff.	Uncertainty	Games	LOS %	Draws %
Chess	0.001	0.001	-26	36	200	8%	44.5%
Chess	0.001	0.01	37	36	200	98%	44.5%
Chess	0.001	0.1	-24	37	200	10%	40.0%
Chess	0.001	1	-21	37	200	13%	43.0%
Chess	0.001	10	-67	37	200	0%	43.0%
Chess	0.001	100	-182	43	200	0%	30.0%
Chess	0.01	0.001	16	38	200	79%	39.5%
Chess	0.01	0.01	24	35	200	91%	47.0%
Chess	0.01	0.1	-2	35	200	46%	48.5%
Chess	0.01	1	-30	35	200	5%	47.5%
Chess	0.01	10	-69	36	200	0%	44.5%
Chess	0.01	100	-184	41	200	0%	34.5%
Chess	0.1	0.001	2	35	200	54%	47.5%
Chess	0.1	0.01	4	37	200	57%	42.0%
Chess	0.1	0.1	-2	37	200	46%	41.5%
Chess	0.1	1	-44	36	200	1%	44.5%
Chess	0.1	10	-60	37	200	0%	41.0%
Chess	0.1	100	-173	42	200	0%	33.0%
Chess	1	0.001	0	36	200	50%	45.0%
Chess	1	0.01	10	37	200	71%	42.0%
Chess	1	0.1	-12	35	200	25%	48.5%
Chess	1	1	2	36	200	54%	45.5%
Chess	1	10	-70	38	200	0%	40.0%
Chess	1	100	-166	41	200	0%	33.5%

Chess	10	0.001	-21	36	200	13%	44.0%
Chess	10	0.01	-17	36	200	17%	46.0%
Chess	10	0.1	-16	37	200	20%	43.5%
Chess	10	1	-9	37	200	32%	42.5%
Chess	10	10	-96	38	200	0%	41.0%
Chess	10	100	-220	45	200	0%	29.0%
Chess	100	0.001	30	35	200	95%	48.5%
Chess	100	0.01	21	35	200	88%	48.0%
Chess	100	0.1	7	37	200	65%	42.0%
Chess	100	1	-69	37	200	0%	41.5%
Chess	100	10	-81	38	200	0%	38.0%
Chess	100	100	-225	45	200	0%	28.0%

Tabelle A.1.: Ergebnisse der Rastersuche für Schach mit der *Evaluation Novelty*.

Variant	Alpha	Beta	Elo-Diff.	Uncertainty	Games	LOS %	Draws %
Crazyhouse	0.001	0.001	21	48	200	81%	3.0%
Crazyhouse	0.001	0.01	-4	48	200	44%	3.0%
Crazyhouse	0.001	0.1	-42	48	200	4%	3.0%
Crazyhouse	0.001	1	-16	48	200	26%	0.5%
Crazyhouse	0.001	10	-141	52	200	0%	1.5%
Crazyhouse	0.001	100	-279	65	200	0%	2.5%
Crazyhouse	0.01	0.001	-12	48	200	31%	0.5%
Crazyhouse	0.01	0.01	-23	47	200	17%	4.5%
Crazyhouse	0.01	0.1	-9	48	200	36%	2.5%
Crazyhouse	0.01	1	-16	48	200	26%	2.5%
Crazyhouse	0.01	10	-131	51	200	0%	4.0%
Crazyhouse	0.01	100	-285	65	200	0%	3.5%
Crazyhouse	0.1	0.001	26	48	200	86%	2.5%
Crazyhouse	0.1	0.01	-26	48	200	14%	3.5%
Crazyhouse	0.1	0.1	-4	48	200	44%	0.0%
Crazyhouse	0.1	1	-85	49	200	0%	5.0%
Crazyhouse	0.1	10	-123	51	200	0%	3.0%
Crazyhouse	0.1	100	-282	65	200	0%	2.0%
Crazyhouse	1	0.001	16	48	200	74%	2.5%
Crazyhouse	1	0.01	9	48	200	64%	1.5%
Crazyhouse	1	0.1	-2	48	200	47%	1.5%

Crazyhouse	1	1	-7	48	200	39%	1.0%
Crazyhouse	1	10	-145	52	200	0%	2.5%
Crazyhouse	1	100	-330	73	200	0%	2.0%
Crazyhouse	10	0.001	-12	48	200	31%	2.5%
Crazyhouse	10	0.01	26	48	200	86%	2.5%
Crazyhouse	10	0.1	-19	48	200	22%	2.5%
Crazyhouse	10	1	-56	48	200	1%	4.0%
Crazyhouse	10	10	-162	53	200	0%	3.5%
Crazyhouse	10	100	-282	65	200	0%	2.0%
Crazyhouse	100	0.001	-10	48	200	33%	3.0%
Crazyhouse	100	0.01	16	48	200	74%	1.5%
Crazyhouse	100	0.1	2	48	200	53%	0.5%
Crazyhouse	100	1	-44	48	200	4%	2.5%
Crazyhouse	100	10	-212	57	200	0%	2.5%
Crazyhouse	100	100	-308	69	200	0%	2.0%

Tabelle A.2.: Ergebnisse der Rastersuche für Crazyhouse mit der *Evaluation Novelty*.

Variant	Alpha	Beta	Elo-Diff.	Uncertainty	Games	LOS %	Draws %
Chess	0.001	0.001	-7	48	100	39%	50.0%
Chess	0.001	0.01	28	47	100	88%	52.0%
Chess	0.001	0.1	-24	50	100	17%	47.0%
Chess	0.001	1	-7	52	100	40%	42.0%
Chess	0.001	10	-100	57	100	0%	34.0%
Chess	0.001	100	-177	62	100	0%	29.0%
Chess	0.01	0.001	-4	49	100	44%	49.0%
Chess	0.01	0.01	-31	49	100	10%	49.0%
Chess	0.01	0.1	-10	54	100	35%	37.0%
Chess	0.01	1	10	54	100	65%	39.0%
Chess	0.01	10	-115	53	100	0%	42.0%
Chess	0.01	100	-156	57	100	0%	36.0%
Chess	0.1	0.001	-35	50	100	9%	46.0%
Chess	0.1	0.01	-7	47	100	39%	52.0%
Chess	0.1	0.1	-38	53	100	8%	41.0%
Chess	0.1	1	-35	54	100	10%	38.0%
Chess	0.1	10	-49	52	100	3%	42.0%
Chess	0.1	100	-156	64	100	0%	24.0%

Chess	1	0.001	24	51	100	83%	45.0%
Chess	1	0.01	-10	53	100	35%	41.0%
Chess	1	0.1	21	53	100	78%	40.0%
Chess	1	1	-21	52	100	22%	42.0%
Chess	1	10	-93	52	100	0%	44.0%
Chess	1	100	-182	63	100	0%	28.0%
Chess	10	0.001	0	53	100	50%	40.0%
Chess	10	0.01	4	51	100	55%	45.0%
Chess	10	0.1	-28	54	100	16%	38.0%
Chess	10	1	-49	57	100	5%	32.0%
Chess	10	10	-210	66	100	0%	26.0%
Chess	10	100	-263	78	100	0%	16.0%
Chess	100	0.001	-28	49	100	13%	48.0%
Chess	100	0.01	-28	47	100	12%	52.0%
Chess	100	0.1	-78	56	100	0%	36.0%
Chess	100	1	-323	77	100	0%	21.0%
Chess	100	10	-676	nan	100	0%	0.0%
Chess	100	100	-920	nan	100	0%	1.0%

Tabelle A.3.: Ergebnisse der Rastersuche für Schach mit der *Pseudocount Novelty*.

Variant	Alpha	Beta	Elo-Diff.	Uncertainty	Games	LOS %	Draws %
Crazyhouse	0.001	0.001	17	69	100	69%	1.0%
Crazyhouse	0.001	0.01	21	68	100	73%	2.0%
Crazyhouse	0.001	0.1	-14	68	100	34%	2.0%
Crazyhouse	0.001	1	-24	69	100	24%	1.0%
Crazyhouse	0.001	10	-147	74	100	0%	4.0%
Crazyhouse	0.001	100	-308	103	100	0%	1.0%
Crazyhouse	0.01	0.001	-38	69	100	13%	1.0%
Crazyhouse	0.01	0.01	-28	69	100	21%	0.0%
Crazyhouse	0.01	0.1	-24	69	100	24%	1.0%
Crazyhouse	0.01	1	-45	68	100	9%	3.0%
Crazyhouse	0.01	10	-151	75	100	0%	3.0%
Crazyhouse	0.01	100	-282	96	100	0%	1.0%
Crazyhouse	0.1	0.001	10	69	100	62%	1.0%
Crazyhouse	0.1	0.01	-31	68	100	18%	3.0%
Crazyhouse	0.1	0.1	28	68	100	79%	2.0%

Crazyhouse	0.1	1	-60	70	100	4%	1.0%
Crazyhouse	0.1	10	-93	69	100	0%	6.0%
Crazyhouse	0.1	100	-338	111	100	0%	1.0%
Crazyhouse	1	0.001	28	68	100	79%	2.0%
Crazyhouse	1	0.01	4	69	100	54%	1.0%
Crazyhouse	1	0.1	-31	68	100	18%	3.0%
Crazyhouse	1	1	-21	68	100	27%	4.0%
Crazyhouse	1	10	-104	71	100	0%	3.0%
Crazyhouse	1	100	-282	96	100	0%	1.0%
Crazyhouse	10	0.001	-7	68	100	42%	2.0%
Crazyhouse	10	0.01	7	68	100	58%	2.0%
Crazyhouse	10	0.1	-14	69	100	35%	0.0%
Crazyhouse	10	1	-45	69	100	10%	1.0%
Crazyhouse	10	10	-196	80	100	0%	3.0%
Crazyhouse	10	100	-309	103	100	0%	1.0%
Crazyhouse	100	0.001	-7	68	100	42%	2.0%
Crazyhouse	100	0.01	-35	69	100	16%	2.0%
Crazyhouse	100	0.1	-111	73	100	0%	1.0%
Crazyhouse	100	1	-210	83	100	0%	2.0%
Crazyhouse	100	10	-676	nan	100	0%	0.0%
Crazyhouse	100	100	$-\infty$	nan	100	0%	0.0%

Tabelle A.4.: Ergebnisse der Rastersuche für Crazyhouse mit der *Pseudocount Novelty*.

Variant	Alpha	Beta	Elo-Diff.	Uncertainty	Games	LOS %	Draws %
Chess	0.001	0.001	-28	48	100	13%	50.0%
Chess	0.001	0.01	-35	46	100	7%	54.0%
Chess	0.001	0.1	-31	55	100	13%	37.0%
Chess	0.001	1	-21	50	100	21%	46.0%
Chess	0.001	10	-81	50	100	0%	47.0%
Chess	0.001	100	-182	56	100	0%	38.0%
Chess	0.01	0.001	-10	51	100	34%	45.0%
Chess	0.01	0.01	-7	47	100	39%	52.0%
Chess	0.01	0.1	-21	50	100	21%	46.0%
Chess	0.01	1	-4	48	100	44%	51.0%
Chess	0.01	10	-147	52	100	0%	44.0%
Chess	0.01	100	-186	61	100	0%	31.0%

Chess	0.1	0.001	4	49	100	56%	49.0%
Chess	0.1	0.01	10	49	100	66%	49.0%
Chess	0.1	0.1	17	53	100	74%	41.0%
Chess	0.1	1	7	50	100	61%	46.0%
Chess	0.1	10	-49	50	100	3%	46.0%
Chess	0.1	100	-168	58	100	0%	35.0%
Chess	1	0.001	-10	54	100	35%	37.0%
Chess	1	0.01	35	50	100	91%	46.0%
Chess	1	0.1	-38	53	100	8%	41.0%
Chess	1	1	-31	53	100	12%	41.0%
Chess	1	10	-53	51	100	2%	45.0%
Chess	1	100	-225	71	100	0%	21.0%
Chess	10	0.001	4	54	100	55%	37.0%
Chess	10	0.01	10	49	100	66%	49.0%
Chess	10	0.1	4	50	100	56%	47.0%
Chess	10	1	-28	49	100	13%	48.0%
Chess	10	10	-100	53	100	0%	42.0%
Chess	10	100	-230	65	100	0%	28.0%
Chess	100	0.001	-7	48	100	39%	50.0%
Chess	100	0.01	24	53	100	82%	41.0%
Chess	100	0.1	-24	53	100	18%	41.0%
Chess	100	1	-14	50	100	29%	46.0%
Chess	100	10	-81	58	100	0%	31.0%
Chess	100	100	-315	87	100	0%	14.0%

Tabelle A.5.: Ergebnisse der Rastersuche für Schach mit der Monte-Carlo Graph Search und der *Evaluation Novelty*.

Variant	Alpha	Beta	Elo-Diff.	Uncertainty	Games	LOS %	Draws %
Crazyhouse	0.001	0.001	53	69	100	93.4%	1.0%
Crazyhouse	0.001	0.01	31	69	100	81.7%	1.0%
Crazyhouse	0.001	0.1	-14	69	100	34.5%	0.0%
Crazyhouse	0.001	1	-42	69	100	11.3%	2.0%
Crazyhouse	0.001	10	-131	74	100	0.0%	2.0%
Crazyhouse	0.001	100	-301	102	100	0.0%	0.0%
Crazyhouse	0.01	0.001	-14	69	100	34.5%	0.0%
Crazyhouse	0.01	0.01	-4	69	100	46.0%	1.0%

Crazyhouse	0.01	0.1	-42	69	100	11.5%	0.0%
Crazyhouse	0.01	1	-28	69	100	21.2%	0.0%
Crazyhouse	0.01	10	-164	76	100	0.0%	4.0%
Crazyhouse	0.01	100	-288	97	100	0.0%	2.0%
Crazyhouse	0.1	0.001	-10	69	100	38.2%	1.0%
Crazyhouse	0.1	0.01	-7	69	100	42.1%	0.0%
Crazyhouse	0.1	0.1	28	68	100	79.0%	2.0%
Crazyhouse	0.1	1	-45	69	100	9.6%	1.0%
Crazyhouse	0.1	10	-127	72	100	0.0%	5.0%
Crazyhouse	0.1	100	-301	100	100	0.0%	2.0%
Crazyhouse	1	0.001	14	69	100	65.5%	0.0%
Crazyhouse	1	0.01	-14	68	100	34.3%	2.0%
Crazyhouse	1	0.1	-17	69	100	30.8%	1.0%
Crazyhouse	1	1	-45	69	100	9.6%	1.0%
Crazyhouse	1	10	-139	73	100	0.0%	4.0%
Crazyhouse	1	100	-323	104	100	0.0%	3.0%
Crazyhouse	10	0.001	10	69	100	61.8%	1.0%
Crazyhouse	10	0.01	21	69	100	72.6%	0.0%
Crazyhouse	10	0.1	-45	69	100	9.6%	1.0%
Crazyhouse	10	1	-70	71	100	2.3%	0.0%
Crazyhouse	10	10	-220	84	100	0.0%	2.0%
Crazyhouse	10	100	-388	111	100	0.0%	1.0%
Crazyhouse	100	0.001	-14	69	100	34.5%	0.0%
Crazyhouse	100	0.01	-56	69	100	5.3%	2.0%
Crazyhouse	100	0.1	-17	69	100	30.8%	1.0%
Crazyhouse	100	1	-38	69	100	13.4%	1.0%
Crazyhouse	100	10	-139	74	100	0.0%	2.0%
Crazyhouse	100	100	-364	115	100	0.0%	0.0%

Tabelle A.6.: Ergebnisse der Rastersuche für Crazyhouse mit der Monte-Carlo Graph Search und der *Evaluation Novelty*.

Ergebnisse der Tests für die Elo-Entwicklung

Variant	Alpha	Beta	Nodes	Elo-Diff	Unc.	Games	Score %	Draw %
Chess	0	0	1600	363	30	900	89.0%	12.9%
Chess	0.01	0.001	1600	335	28	900	87.3%	14.3%
Chess	0.01	0.001	800	141	21	900	69.3%	22.3%
Chess	0	0	800	137	21	900	68.8%	23.1%
Chess	0.01	0.001	400	-1	20	900	49.9%	22.4%
Chess	0	0	400	-13	20	900	48.2%	21.2%
Chess	0.01	0.001	200	-157	23	900	28.8%	14.3%
Chess	0	0	200	-166	23	900	27.8%	12.7%
Chess	0	0	100	-279	27	900	16.7%	10.8%
Chess	0.01	0.001	100	-311	29	900	14.3%	9.2%

Tabelle A.7.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Schach. Für die Neuartigkeitsbewertung wurde die *Evaluation Novelty* verwendet.

Variant	Alpha	Beta	Nodes	Elo-Diff	Unc.	Games	Score %	Draw %
Crazyhouse	0	0	1600	344	44	540	87.9%	2.8%
Crazyhouse	0.1	0.001	1600	320	42	540	86.3%	1.5%
Crazyhouse	0	0	800	155	32	540	70.9%	3.0%
Crazyhouse	0.1	0.001	800	141	32	540	69.3%	1.5%
Crazyhouse	0.1	0.001	400	14	29	540	51.9%	3.5%
Crazyhouse	0	0	400	8	29	540	51.1%	2.6%
Crazyhouse	0.1	0.001	200	-154	32	540	29.2%	2.8%
Crazyhouse	0	0	200	-168	32	540	27.5%	4.3%
Crazyhouse	0	0	100	-310	41	540	14.4%	3.1%
Crazyhouse	0.1	0.001	100	-353	46	540	11.6%	1.7%

Tabelle A.8.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Crazyhouse. Für die Neuartigkeitsbewertung wurde die *Evaluation Novelty* verwendet.

Variant	Alpha	Beta	Nodes	Elo-Diff	Unc.	Games	Score %	Draw %
Chess	0	0	1600	356	38	540	88.6%	13.5%
Chess	0.001	0.01	1600	346	38	540	88.0%	12.2%
Chess	0	0	800	154	28	540	70.8%	21.7%
Chess	0.001	0.01	800	140	28	540	69.2%	21.3%
Chess	0.001	0.01	400	-1	26	540	49.8%	19.6%
Chess	0	0	400	-5	26	540	49.3%	21.5%
Chess	0	0	200	-168	30	540	27.5%	12.4%
Chess	0.001	0.01	200	-185	30	540	25.6%	15.4%
Chess	0.001	0.01	100	-293	36	540	15.6%	10.9%
Chess	0	0	100	-294	36	540	15.6%	11.1%

Tabelle A.9.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Schach. Für die Neuartigkeitsbewertung wurde die *Pseudocount Novelty* verwendet.

Variant	Alpha	Beta	Nodes	Elo-Diff	Unc.	Games	Score %	Draw %
Crazyhouse	1	0.001	1600	341	44	540	87.7%	2.8%
Crazyhouse	0	0	1600	322	42	540	86.5%	2.6%
Crazyhouse	0	0	800	159	32	540	71.4%	2.4%
Crazyhouse	1	0.001	800	159	32	540	71.4%	1.3%
Crazyhouse	0	0	400	28	29	540	54.1%	3.0%
Crazyhouse	1	0.001	400	0	29	540	50.0%	1.9%
Crazyhouse	1	0.001	200	-159	32	540	28.6%	3.1%
Crazyhouse	0	0	200	-185	33	540	25.6%	2.8%
Crazyhouse	0	0	100	-317	42	540	13.9%	1.9%
Crazyhouse	1	0.001	100	-366	47	540	10.8%	1.7%

Tabelle A.10.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Crazyhouse. Für die Neuartigkeitsbewertung wurde die *Pseudocount Novelty* verwendet.

Variant	Alpha	Beta	Nodes	Elo-Diff	Unc.	Games	Score %	Draw %
Chess	0	0	1600	356	38	540	88.6%	13.5%
Chess	1	0.01	1600	352	38	540	88.3%	13.3%
Chess	1	0.01	800	159	29	540	71.4%	18.7%
Chess	0	0	800	136	28	540	68.6%	20.9%
Chess	0	0	400	-18	27	540	47.4%	17.0%
Chess	1	0.01	400	-24	27	540	46.5%	17.0%
Chess	0	0	200	-165	31	540	27.9%	10.6%
Chess	1	0.01	200	-191	32	540	25.0%	10.4%
Chess	1	0.01	100	-247	34	540	19.4%	10.0%
Chess	0	0	100	-277	36	540	16.9%	8.1%

Tabelle A.11.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Schach. Für die Suche wurde die MCGS verwendet und als Neuartigkeitsmetrik die *Evaluation Novelty*.

Variant	Alpha	Beta	Nodes	Elo-Diff	Unc.	Games	Score %	Draw %
Crazyhouse	0.001	0.001	1600	335	44	540	87.3%	2.0%
Crazyhouse	0	0	1600	334	44	540	87.2%	1.9%
Crazyhouse	0.001	0.001	800	152	31	540	70.6%	4.1%
Crazyhouse	0	0	800	146	31	540	69.8%	4.1%
Crazyhouse	0	0	400	15	29	540	52.1%	3.9%
Crazyhouse	0.001	0.001	400	-3	29	540	49.6%	3.3%
Crazyhouse	0.001	0.001	200	-137	31	540	31.2%	3.5%
Crazyhouse	0	0	200	-160	32	540	28.5%	4.4%
Crazyhouse	0.001	0.001	100	-349	44	540	11.9%	2.6%
Crazyhouse	0	0	100	-350	45	540	11.8%	2.8%

Tabelle A.12.: Elo-Entwicklung relativ zu der Anzahl der Evaluationen durch das neuronale Netz bei Crazyhouse. Für die Suche wurde die MCGS verwendet und als Neuartigkeitsmetrik die *Evaluation Novelty*.