h_da hochschule darmstadt informatik

member of EUT+ EUROPEAN UNIVERSITY OF TECHNOLOGY

Hochschule Darmstadt

– Fachbereich Informatik –

Robustheit und Generalisierbarkeit in algorithmischen und Reinforcement Learning gestützten Lösungsansätzen: Eine Fallstudie mit Vier Gewinnt

Abschlussarbeit zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt von

Leo Herrmann

Matrikelnummer: 1111455

Referentin: Prof. Dr. Elke Hergenröther
Korreferent: Adriatik Gashi

1 Kurzfassung

Inhaltsverzeichnis

Abbildungsverzeichnis

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Darmstadt, 21.03.2023

Leo Herrmann

2 Einleitung

Fortschreitende Automatisierung durchdringt zahlreiche Bereiche der Gesellschaft, so zum Beispiel die Fertigungsindustrie, das Gesundheitswesen oder den Straßenverkehr. Zwei fundamentale Ansätze sind dabei regelbasierte Algorithmen und Machine Learning. Die Einsatzbedingungen von Automatisierungssystemen unterscheiden sich häufig von den Bedingungen, unter denen sie entwickelt und getestet werden. Häufig müssen Systeme mit fehlerhaften oder veralteten Informationen arbeiten oder es treten Situationen ein, die bei der Konzipierung der Systeme nicht berücksichtigt werden können. Dabei sinkt die Leistungsfähigkeit dieser Systeme.

Im Rahmen dieser Arbeit werden Robustheit und Generalisierbarkeit eines algorithmischen Ansatzes und eines Reinforcement Learning basierten Ansatzes zur Lösung des Brettspiels „Vier Gewinnt“ untersucht. Bei Robustheit und Generalisierbarkeit handelt es sich um Eigenschaften, die beschreiben, wie gut ein Algorithmus oder RL-Modell in der Praxis funktioniert, in der andere Bedingungen herrschen können als während der Entwicklung und Qualitätssicherung. Diese Kriterien sind besonders relevant für den Erfolg von Algorithmen und Modellen in der Praxis.

Spiele eignen sich zur Untersuchung von Algorithmen und Modellen, weil sie reale Probleme auf kontrollierbare Umgebungen abstrahieren und gleichzeitig reproduzierbare und vergleichbare Messungen ermöglichen. Die Untersuchungen dieser Arbeit erfolgen am Beispiel des Brettspiels „Vier Gewinnt“, da aus früheren Untersuchungen ersichtlich wird, dass sich sowohl algorithmische als auch Reinforcement Learning basierte Lösungen eignen.

Es wird Grundlagenforschung zu verbreiteten algorithmischen Ansätzen und Reinforcement Learning basierten Ansätzen betrieben. Anschließend werden die Aspekte Robustheit und Generalisierbarkeit von zwei Ansätzen aus den jeweiligen Bereichen am Beispiel von Vier Gewinnt empirisch untersucht. Dabei werden neue Erkenntnisse über Lösungsansätze von Vier Gewinnt gewonnen, die sich auf vergleichbare Szenarien in der realen Welt übertragen lassen.

Die zentrale Fragestellung lautet: Inwiefern sind bei Vier Gewinnt algorithmische oder Reinforcement Learning basierte Ansätze robuster oder besser generalisierbar? Das Ziel dieser Arbeit besteht darin, ein detailliertes Verständnis über verschiedene Aspekte von Robustheit und Generalisierbarkeit der zu untersuchenden Ansätze zu bekommen.

3 Grundlagen

In diesem Kapitel wird durch Literaturrecherche eine fundierte theoretische Basis geschaffen, auf die im weiteren Verlauf dieser Arbeit Bezug genommen wird. Zunächst wird Vier Gewinnt als zu lösendes Problem untersucht und eingeordnet. Anschließend folgt eine Auswahl von jeweils einem algorithmischen und einem Reinforcement Learning basiertem Lösungsansatz. Die Funktionsweise beider Lösungsmethoden wird erklärt. Außerdem werden bestehende Theorien und Definitionen zum Thema Robustheit und Generalisierbarkeit zusammengetragen. Sie bilden die Grundlage für die Szenarien und Bewertungskriterien in den Experimenten des Hauptteils.

3.1 Vier Gewinnt

Vier Gewinnt ist ein Brettspiel, das aus einem 7 x 6 Spielfeld besteht. Die beiden Spieler werfen abwechselnd einen Spielstein in eine Spalte hinein, der in dieser Spalte bis zur untersten freien Position fällt. Es gewinnt der Spieler, der als erstes vier Spielsteine in einer horizontalen, vertikalen oder diagonalen Linie nebeneinander stehen hat[6].

Bei Vier Gewinnt handelt es sich um ein kombinatorisches Nullsummenspiel für zwei Spieler. Kombinatorische Spiele weisen „perfekte Information“ auf. Das bedeutet, dass alle Spieler zu jeder Zeit den gesamten Zustand des Spiels kennen. So ist es bei vielen Brettspielen der Fall. Kartenspiele hingegen besitzen diese Eigenschaft meistens nicht, weil jedem Spieler die Handkarten ihrer Gegenspieler unbekannt sind. Bei kombinatorischen Spielen sind außerdem keine Zufallselemente enthalten. Die einzige Herausforderung beim Spielen kombinatorischer Spiele besteht darin, unter einer Vielzahl von Entscheidungsoptionen diejenige auszuwählen, die den besten weiteren Spielverlauf verspricht([5], S. 96-100)([7], Kapitel 4.1).

Bei Zwei-Spieler-Nullsummenspielen, verursacht der Gewinn eines Spielers zwangsläufig einen Verlust des anderen Spielers. Die beiden Spieler haben also entgegengesetzte Interessen([5], S.100)([4], S. 100). Das bedeutet, dass sich der Erfolg von verschiedenen Lösungsansätzen durch die durchschnittliche Gewinnrate im Spiel gegeneinander bewerten lässt. Bei Nullsummenspielen mit mehr als zwei Personen, kann es passieren, dass wenn ein Spieler (bewusst oder versehentlich) nicht optimal spielt, ein zweiter Spieler davon profitiert, während ein dritter Spieler dadurch benachteiligt wird. Solche Wechselwirkungen sind bei Zwei-Personen-Nullsummenspielen ausgeschlossen([5], S. 113 ff.). Das macht die Messergebnisse im Hauptteil besser vergleichbar.

Nach Victor Allis lässt sich die Komplexität eines Spiels von strategie-basierten Zwei-

Spieler-Nullsummenspielen durch ihre Zustandsraum- und Spielbaumkomplexität beschreiben. Die Zustandsraumkomplexität entspricht der Anzahl der verschiedenen möglichen Spielfeldkonfigurationen ab dem Start. Für ein Spiel kann dieser Wert oder zumindest dessen obere Schranke bestimmt werden, indem zunächst alle Konfigurationen des Spielfelds gezählt, dann Einschränkungen wie Regeln und Symmetrie berücksichtigt werden, und die Anzahl der illegalen und redundanten Zustände von der Anzahl aller möglichen Konfigurationen abgezogen wird[4].

Die Spielbaumkomplexität beschreibt die Anzahl der Blattknoten des Lösungsbaums. Der Spielbaum ist ein Baum, der die Zustände eines Spiels als Knoten und die Züge als Kanten darstellt([5], S. 102). Der Lösungsbaum beschreibt die Teilmenge des Spielbaums, der benötigt wird, um die Gewinnaussichten bei optimaler Spielweise beider Spieler zu berechnen. Die Spielbaumkomplexität lässt sich durch die durchschnittliche Spiellänge und der Anzahl der Entscheidungsmöglichkeiten pro Zug (entweder konstant oder abhängig vom Spielfortschritt) approximieren. Da in den meisten Spielen ein Zustand über mehrere Wege erreicht werden kann, fällt die Spielbaumkomplexität meist wesentlich größer aus als die Zustandsraumkomplexität[4].

Die Spielbaumkomplexität ist maßgeblich für die praktische Berechenbarkeit einer starken Lösung. Für Tic Tac Toe wurde durch Allis eine obere Grenze für die Spielbaumkomplexität von 362880 ermittelt und eine starke Lösung lässt sich innerhalb von Sekundenbruchteilen berechnen[11]. Für Schach wird die Spielbaumkomplexität auf 10^{31} geschätzt und die Aussichten auf eine starke Lösung liegen noch in weiter Ferne[13].

Für Vier Gewinnt wurde eine durchschnittliche Spiellänge von 36 Zügen und eine durchschnittliche Anzahl von Entscheidungsmöglichkeiten (freie Spalten) von 4 ermittelt. Damit wurde die Spielbaumkomplexität auf $4^{36} \approx 10^{21}$ geschätzt[4].

Verschiedene Lösungsverfahren von Vier Gewinnt sind bereits ausgiebig untersucht. Das Spiel wurde 1988 von James Dow Allen und Victor Allis unabhängig voneinander mit wissensbasierten Methoden schwach gelöst, was bedeutet, dass für die Anfangsposition eine optimale Strategie ermittelt wurde. Im Fall von Vier Gewinnt kann der Spieler, der den ersten Zug macht, bei optimaler Spielweise immer gewinnen[2][3].

1993 wurde das Spiel von John Tromp auch durch einen Brute-Force Ansatz stark gelöst. Bei dieser Lösung kam Alpha-Beta-Pruning zum Einsatz, um für alle 4.531.985.219.092 legalen Zustände des Spiels den optimalen Zug zu berechnen. Das hat damals etwa 40.000 CPU-Stunden gedauert[17].

Lösungen, die alle Möglichkeiten durchrechnen, um die optimale Entscheidung zu treffen, sind für den Einsatz in der Praxis aufgrund des hohen Rechenaufwands bei

komplexeren Anwendungen auch heute noch selten praktikabel. Aus diesem Grund wird bevorzugt auf gute Heuristiken zurückgegriffen, die den Rechenaufwand minimieren, aber dennoch gute Ergebnisse liefern([8], Kapitel 7.6).

Untersuchungen haben gezeigt, dass sich sowohl regelbasierte Tree-Search-Algorithmen als auch verschiedene Reinforcement Learning Ansätze eignen, um sogenannte Agents zu entwickeln, die das Spiel selbstständig spielen.[1][16][18][15][14][12]. Wissensbasierte Methoden werden in dieser Arbeit nicht näher betrachtet, da ihre Ergebnisse stark an die jeweiligen Spielregeln gebunden und schwer zu verallgemeinern sind.

3.2 Symbolische Algorithmen

3.2.1 Minimax

Minimax ist ein Algorithmus, der aus Sicht eines Spielers ausgehend von einem beliebigen Ursprungsknoten im Spielbaum die darauf folgenden Knoten bewertet und den Kindknoten des Ursprungsknotens mit der besten Bewertung zurückgibt. Bei der Bewertung wird davon ausgegangen, dass der Gegner ebenfalls den Zug wählt, der für ihn am günstigsten ist. Der zu untersuchende Spieler versucht, die Bewertung zu maximieren, während der Gegenspieler versucht, sie zu minimieren.

Zunächst werden die Blattknoten des Spielbaums bewertet. Je günstiger ein Spielfeldzustand für den zu untersuchenden Spieler ist, desto größer ist die Zahl, die diesem Zustand zugeordnet wird. In Abhängigkeit der zuvor bewerteten Knoten, werden nun deren Elternknoten bewertet. Ist im betrachteten Zustand der zu untersuchende Spieler am Zug, übernimmt dieser Zustand die Bewertung des Kindknotens mit der höchsten Bewertung. Umgekehrt ist es, wenn der Gegenspieler Spieler am Zug ist. Dann bekommt der zu untersuchte Knoten die Bewertung des Kindknotens mit der niedrigsten Bewertung. Dieser Vorgang wird wiederholt, bis der Ursprungsknoten erreicht ist. Zurückgegeben wird der Kindknoten des Ursprungsknotens, dem die größte Bewertung zugeordnet wurde.

Erfolgt die Bewertung anhand der Gewinnchancen, führt das dazu, dass die Wahl des Knotens mit der besten Bewertung auch die Gewinnchancen maximiert. Um die Gewinnchancen zu ermitteln, müssen jedoch alle Knoten des Spielbaums untersucht werden. Die Laufzeit des Algorithmus steigt linear zur Anzahl der zu untersuchenden Knoten und damit bei konstanter Anzahl von Möglichkeiten pro Zug exponentiell zur Suchtiefe. Den gesamten Spielbaum zu durchsuchen, ist daher nur für wenig komplexe Spiele praktikabel. Damit die Bewertung in akzeptabler Zeit erfolgen kann, muss für

komplexere Spiele die Suchtiefe oder -breite begrenzt werden und die Bewertunt der Knoten muss auf Grundlage von Heuristiken erfolgen ([7], Kapitel 4)([8], Kapitel 7.6).

3.2.2 Alpha-Beta-Pruning

Beim Alpha-Beta-Pruning handelt es sich um eine Optimierung des Minimax-Algorithmus. Dabei werden die Teilbäume übersprungen, die das Ergebnis nicht beeinflussen können, weil bereits absehbar ist, dass diese Teilbäume bei optimaler Spielweise beider Spieler nicht erreicht werden. Alpha-Beta-Pruning liefert dieselben Ergebnisse wie der Minimax-Algorithmus, aber untersucht dabei wesentlich weniger Knoten im Spielbaum.

Dazu werden während der Suche die Werte Alpha und Beta aufgezeichnet. Alpha entspricht der Mindestbewertung, die der zu untersuchende Spieler garantieren kann, wenn beide Spieler optimal spielen. Beta entspricht der Bewertung, die der Gegenspieler bei optimaler Spielweise maximal zulassen wird. Zu Beginn der Suche wird Alpha auf minus unendlich und Beta auf plus unendlich initialisiert.

Alpha wird aktualisiert, wenn für einen Knoten, bei dem der zu untersuchende Spieler am Zug ist, ein Kindknoten gefunden wurde, dessen Bewertung größer ist als das bisherige Alpha. Beta hingegen wird aktualisiert, wenn für einen Knoten, bei dem der Gegenspieler am Zug ist, ein Kindknoten gefunden wurde, dessen Bewertung kleiner ist als Beta.

Sobald bei einem Knoten Alpha größer oder gleich Beta ist, kann die Untersuchung dessen Kindknoten aus folgenden Gründen abgebrochen werden:

- Ist bei diesem Knoten der zu untersuchende Spieler am Zug, hatte der Gegenspieler in einem zuvor untersuchten Teilbaum bessere Chancen, und wird den aktuellen untersuchten Teilbaum nicht auswählen.
- Ist bei diesem Knoten der zu Gegenspieler am Zug, hatte der zu untersuchende Spieler in eine zuvor untersuchten Teilbaum bessere Chancen, und wird den aktuell untersuchten Teilbaum nicht auswählen([8], Kapitel 7.8; [7], Kapitel 4.5).

So kann im Vergleich zum Minimax-Algorithmus die Untersuchung von 80% bis 95% der Knoten übersprungen werden. Der Anteil der Knoten, die bei der Untersuchung übersprungen werden können, ist abhängig davon, wie schnell das Fenster zwischen Alpha und Beta verkleinert wird. Wenn die Reihenfolge, in der die Züge untersucht werden, geschickt gewählt wird, kann dies sogar zu einer Reduktion von über 99% führen([8], Kapitel 7.8). In Schach ist dies beispielsweise möglich, indem Züge früher bewertet werden, je höherwertiger eine im Zug geschmissene Figur ist.

Durch Alpha-Beta-Pruning kann der Spielbaum bei gleichbleibender Zeit wesentlich tiefer durchsucht werden, was beim Einsatz von Heuristiken als Bewertungsfunktion zu präziseren Ergebnissen führt. Die Laufzeit ist allerdings weiterhin exponentiell abhängig zur Suchtiefe. Den gesamten Spielbaum zu durchsuchen, um die Bewertung auf tatsächlichen Gewinnaussichten durchzuführen, bleibt bei komplexen Spielen weiterhin unpraktikabel([8], Kapitel 7.8).

Heuristische Bewertungsfunktionen sind in der Hinsicht problematisch, als dass sie spezifisch für die Regeln eines Spiels zugeschnitten sein müssen, bzw. dass es für bestimmte Anwendungsfälle keine guten Heuristiken gibt([7], Kapitel 4.5). Das führt dazu, dass die Eigenschaften von Alpha-Beta-Pruning schwer auf verschiedene Anwendungsfälle übertragbar sind.

3.2.3 MCTS

Bei MCTS ist ein regelbasierter Ansatz, der ohne Heuristiken auskommt...

3.3 Reinforcement Learning

3.4 Robustheit und Generalisierbarkeit

Verschiedene Reinforcement Learning Ansätze sind auf ihre Robustheit untersucht und es gibt verschiedene Verfahren, um RL-Modelle auf Robustheit zu optimieren[10].

4 Konzept

In diesem Kapitel wird erklärt, wie eine Messumgebung aufgesetzt wurde, um die Eigenschaften der Lösungsansätze Robustheit und Generalisierbarkeit empirisch zu bewerten. In dieser Messumgebung spielen zwei Agents, die die zu untersuchenden Ansätze implementieren, das Spiel wiederholt gegeneinander. Dabei werden deren Gewinnraten und die Spieldauer gemessen.

Die Messungen werden unter verschiedenen Szenarien durchgeführt, in denen die Spielumgebung verschiedene Eigenschaften besitzt. Diese Szenarien enthalten unter anderem gestörte Daten, stochastische Elemente oder veränderte Spielregeln:

- Neutrale Umgebung als Grundlage für die folgenden Messungen.
- Rauschen: Agents erhalten fehlerhafte Informationen über das Spielfeld.

- Stochastik: Unter einer bestimmten Wahrscheinlichkeit landet ein Spielstein nicht in der vorgesehenen Spalte sondern in eine benachbarte Spalte.
- Stochastik: Unter einer bestimmten Wahrscheinlichkeit führt ein Spieler nicht den Zug aus, den er für am besten hält, sondern einen zufälligen Zug.
- Stochastik: Unter einer bestimmten Wahrscheinlichkeit führt ein Spieler mehrere Züge hintereinander durch.
- Generalisierbarkeit: Zum Gewinnen werden nicht vier Spielsteine in einer Reihe benötigt, sondern fünf.

Als Grundlage für die Messumgebung dient das PettingZoo Toolkit. Es abstrahiert Probleme in Umgebungen und stellt eine Schnittstelle für Agents bereit, die mit verschiedene Lösungsstrategien mit den Umgebungen interagieren. Eine Umgebung, die das Spiel Vier Gewinnt abstrahiert, ist Teil des PettingZoo Toolkits. Es kommen Reinforcement Learning Modelle zum Einsatz, die aus Machine Learning Bibliotheken wie CleanRL oder Stable-Baselines bereitgestellt werden. Falls vorhanden, wird auf fertig implementierte Algorithmen zurückgegriffen.

5 Realisierung

6 Ergebnisdiskussion

7 Zusammenfassung und Ausblick

8 Literaturverzeichnis

[1] E. Alderton, E. Wopat, J. Koffman. *Reinforcement Learning for Connect Four.* Techn. Ber. Stanford University, Stanford, California 94305, USA, 2019.

[2] James Dow Allen. *The complete book of Connect 4: history, strategy, puzzles.* New York, NY : Puzzle Wright Press, 2010.

[3] Victor Allis. „A Knowledge-Based Approach of Connect-Four". In: *J. Int. Comput. Games Assoc.* 11 (1988), S. 165. URL: https://api.semanticscholar.org/CorpusID:24540039.

[4] Victor Allis. „Searching for solutions in games and artificial intelligence". In: 1994. URL: https://api.semanticscholar.org/CorpusID:60886521.

[5] Jörg Bewersdorff. *Glück, Logik und Bluff: Mathematik im Spiel - Methoden, Ergebnisse und Grenzen.* 7. Aufl. Springer Spektrum Wiesbaden, 8. Mai 2018. ISBN: 978-3-658-21764-8. DOI: 10.1007/978-3-658-21765-5.

[6] Milton Bradley Company. *Connect Four.* https://www.unco.edu/hewit/pdf/giant-map/connect-4-instructions.pdf. [Letzer Zugriff am 17. December-2024]. 1990.

[7] Kevin Ferguson, Max Pumperla. *Deep Learning and the Game of Go.* Manning Publications, January 2019.

[8] George T. Heineman, Gary Pollice, Stanley Selkow. *Algorithms in a Nutshell.* O'Reilly Media, Inc., October 2008.

[9] „IEEE Standard Glossary of Software Engineering Terminology". In: *IEEE Std 610.12-1990* (1990), S. 1–84. DOI: 10.1109/IEEESTD.1990.101064.

[10] Janosch Moos u. a. „Robust Reinforcement Learning: A Review of Foundations and Recent Advances“. In: *Machine Learning and Knowledge Extraction* 4.1 (2022), S. 276–315. ISSN: 2504-4990. DOI: 10.3390/make4010013. URL: https://www.mdpi.com/2504-4990/4/1/13.

[11] Aditya Jyoti Paul. „Randomized fast no-loss expert system to play tic tac toe like a human“. In: *CoRR* abs/2009.11225 (2020). arXiv: 2009.11225. URL: https://arxiv.org/abs/2009.11225.

[12] Yiran Qiu, Zihong Wang, Duo Xu. „Comparison of Four AI Algorithms in Connect Four“. In: *MEMAT 2022; 2nd International Conference on Mechanical Engineering, Intelligent Manufacturing and Automation Technology*. 2022, S. 1–5.

[13] Jonathan Schaeffer u. a. „Checkers Is Solved“. In: *Science* 317 (Okt. 2007), S. 1518–1522. DOI: 10.1126/science.1144079.

[14] Kavita Sheoran u. a. „Solving Connect 4 Using Optimized Minimax and Monte Carlo Tree Search“. In: *Advances and Applications in Mathematical Sciences* 21.6 (2022), S. 3303–3313.

[15] Henry Taylor, Leonardo Stella. *An Evolutionary Framework for Connect-4 as Test-Bed for Comparison of Advanced Minimax, Q-Learning and MCTS*. 2024. arXiv: 2405.16595 [cs.AI]. URL: https://arxiv.org/abs/2405.16595.

[16] Markus Thill, Patrick Koch, Wolfgang Konen. *Reinforcement Learning with N-tuples on the Game Connect-4*. Techn. Ber. Department of Computer Science, Cologne University of Applied Sciences, 51643 Gummersbach, Germany, 2012.

[17] John Tromp. *John's Connect Four Playground*. https://en.wikipedia.org/w/index.php?title=Wine&oldid=1262619132. [Letzer Zugriff am 13. December-2024].

[18] Stephan Wäldchen, Felix Huber, Sebastian Pokutta. *Training Characteristic Functions with Reinforcement Learning: XAI-methods play Connect Four*. 2022. arXiv: 2202.11797 [cs.LG]. URL: https://arxiv.org/abs/2202.11797.