Hauptseminar: Skat-KI auf Basis von Monte Carlo Tree Search

Erwin Lang

Haw Hamburg

1 Einleitung

In dieser Ausarbeitung wird die bisherige Arbeit bezüglich einer künstlichen Intelligenz für das Kartenspiel Skat vorgestellt. Zusätzlich soll die geplante Masterarbeit im Rahmen einer Thesis-Outline beschrieben werden.

Diese Arbeit ist in folgende Abschnitte gegliedert. Zunächst wird die Motivation der Arbeit beschrieben. In Abschnitt 2 werden die Grundlagen für das Verständnis gegeben. Der aktuelle Stand der Arbeit wird in Abschnitt 3 näher beschrieben. Weitere Analysen, die vor der Masterarbeit noch durchgeführt werden sollen werden in Abschnitt 4 vorgestellt. Anschließend wird die geplante Masterarbeit skizziert (5), sowie die Chancen und Risiken analysiert (6).

1.1 Motivation

Künstliche Intelligenz im allgemeinen kann als lernende Agenten in fest definierten Umgebungen verstanden werden. Spiele bieten durch klare Abläufe und Regeln von vornherein eine Abstraktionsebene, während Szenarien aus der realen Welt zunächst analysiert und abstrahiert werden müssen. Aus diesem Grund eignen sich Spiele besonders gut für die Forschung im Bereich der künstlichen Intelligenz. Skat ist eines der am weitesten verbreiteten Kartenspiele, ist jedoch mit Blick auf die KI bisher nur wenig betrachtet worden. Besondere Eigenschaften von Skat sind dabei unter Anderem die Asymmetrie zwischen den Spielern, die Unterteilung in eine Biet- sowie Spielphase und die unvollständigen Informationen durch verdeckte Karten.

Monte Carlo Tree Search wird als Methodik besonders hervorgehoben, da MCTS in vielen aktuellen Umsetzungen, wie zum Beispiel Google AlphaGo [8] verwendet wird. Besonders ist MCTS Zusammen mit anderen Ansätzen für eine Hybride KI interessant [5] [7]. Diese Kombination aus einem unterrepräsentierten Spiel und einem interessanten neuartigen Ansatz bildet die Hauptmotivation für die Arbeit.

2 Grundlagen

In diesem Teil der Ausarbeitung soll zunächst ein grober Überblick zu Skat gegeben werden um ausreichend Kontext für den Rest der Arbeit herzustellen. Anschließend wird Monte Carlo Tree Search kurz vorgestellt.

2.1 Skat

Skat wird mit einem 32 Kartenblatt und genau 3 Spielern gespielt. Dabei wird immer zwei gegen einen gespielt. Das Spiel wird in zwei Phasen unterteilt, das Reizen, und das Spiel. Reizen ist hierbei der Begriff für eine Bietphase bei der auf Basis der Hände geboten und ein Spieltyp festgelegt wird. Das eigentliche Spiel ähnelt anderen Stichspielen. Es werden nacheinander Karten gespielt und die stärkste Karte gewinnt. Wenn alle Karten gespielt sind findet eine Punkteauswertung für den gewählten Spieltyp statt. Der Ablauf von Bridge hat große Ähnlichkeit mit Skat, daher sind Arbeiten zu Bridge mit in die Recherche für diese Arbeit eingeflossen.

2.2 Monte Carlo Tree Search

MCTS ist ein aktueller Ansatz der Monte Carlo Methoden mit Baumsuchen kombiniert. Abbildung 1 zeigt den Ablauf des Verfahrens. Das Prinzip ähnelt anderen Suchverfahren in dem Zustände mit einer Bewertung versehen werden und auf Basis einer Explorationsrate ausgewählt werden. Der entscheidende Unterschied gegenüber typischen Verfahren wie MinMax oder Alpha-Beta Pruning ist jedoch, dass keine Heuristik benötigt wird, sondern Monte Carlo Simulationen für die Bewertung verwendet werden [1] [3].

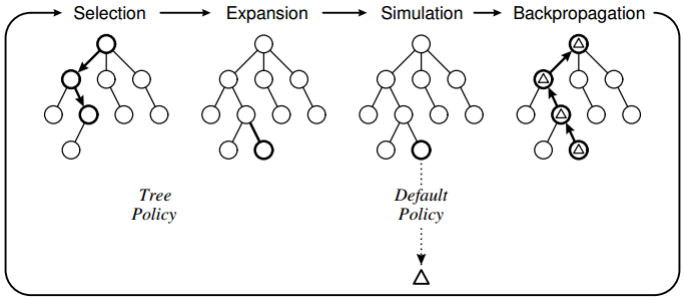

Fig. 1. MCTS Ablauf aus [1]

3 Bisherige Arbeit

In diesem Teil der Arbeit sollen die Ergebnisse aus dem Grundprojekt vorgestellt werden. Im Grundprojekt ging es vor Allem um eine Einarbeitung in die Thematik, sowie um einen Vergleich zwischen diversen Monte Carlo Verfahren.

3.1 Infrastruktur

Im ersten Schritt ging es darum die Infrastruktur für die Implementierung zu schaffen. Dazu musste zunächst eine Skatimplementierung ausgewählt werden. Die Wahl ist hierbei auf JSkat gefallen. JSkat bot bereits eine eingebaute Schnittstelle für die Entwicklung von Spielern. Für die Umsetzung von Monte Carlo Methoden werden jedoch simulierte Spiele benötigt. Hier musste der Code noch angepasst werden. Als Programmiersprache wurde Java verwendet.

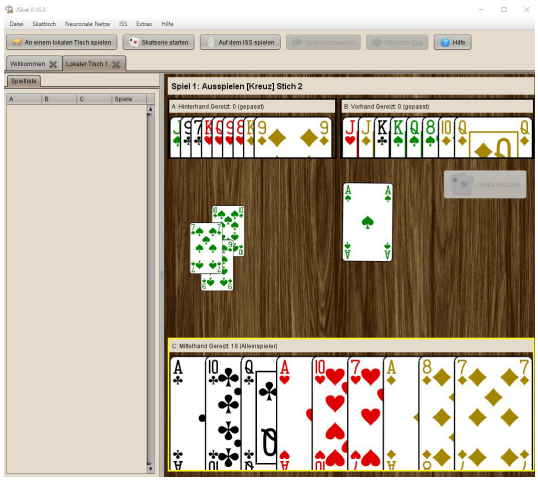

Fig. 2. JSkat Oberfläche

3.2 Umsetzung

Nachdem die Umgebung geschaffen wurde ist das Szenario für die Implementierung definiert worden. Es wurde zunächst nur die Spielphase betrachtet. Außerdem ist das Szenario eingeschränkt worden indem nur der Alleinspieler betrachtet wurde. Die Gegenspieler sind als vollkommen zufällige Spieler umgesetzt worden.

In dieser Umgebung sind dann drei Monte Carlo Varianten implementiert worden. Die erste Variante beschreibt das grundlegende Verfahren ohne "Gedächtnis". Es wurden vom aktuellen Zustand aus Simulationen durchgeführt und die Karte mit dem besten Simulationsergebnis wurde gespielt. Dies entspricht eher dem stochastischen Monte Carlo als irgendeiner tatsächlichen Intelligenz. Dieses Verfahren wurde als einfaches Monte Carlo bezeichnet.

Die zweite Umsetzung entspricht dem On Policy Monte Carlo [10] und unterscheidet sich vor Allem dadurch, dass Q-Werte für besuchte Zustände im Speicher gehalten werden. Durch diese Änderung tritt ein Lerneffekt auf. Anzumerken ist jedoch auch der erhöhte Speicherbedarf.

Der letzte Algorithmus ist das bereits vorgestellte Monte Carlo Tree Search. Ähnlich wie im vorherigen Verfahren werden hier auch Q-Werte gesammelt. Diese werden allerdings in den Knoten des Baumes abgelegt.

3.3 Tests

Die umgesetzen Algorithmen sind mit einer Simulationszahl von 1000 bis 20000 und jeweils 90 Spielen pro Konfiguration getestet worden. Für Monte Calro Tree Search ist die Simulationszahl zwischen Anzahl der Explorationen neuer Knoten, und Anzahl von Simulationen für jeden einzelnen Knoten aufgeteilt worden. Ein Beispiel wäre, dass 20 mal ein Knoten (Spielzustand) ausgewählt wird und für jeden ausgewählten Knoten 20 Spiele simuliert werden um neue Q-Werte zu erhalten. Dabei können Knoten mehrfach gewählt werden.

Algorithmus	Siege	Niederlagen	Sieg %
Grundlegendes Monte Carlo	37	53	41%
On Policy Monte Carlo	43	47	47%
Monte Carlo Tree Search	44	46	48%
Zufall (Referenz)	20	70	22%

Table 1. Testergenisse mit der maximalen Anzahl an Simulationen je Algorithmus

3.4 Ergebnisse

Tabelle 1 zeigt die Ergebnisse für jeden Algorithmus mit der jeweils größten Anzahl an Simulationen. Alle Varianten sind mit zunehmender Simulationszahl besser geworden, Die Verbesserung für das grundlegende Verfahren war jedoch sehr gering. Dies lässt auf ein mögliches Plateau des Verfahrens schließen. Die Ergebnisse von OPMC und MCTS liegen sehr nah beieinander, was aufgrund der Ähnlichkeit der Verfahren nicht überraschend ist. MCTS hat jedoch mehr potenzielle Optimierungen und ist daher am erfolgversprechendsten für die weitere Arbeit.

Während des Grundprojektes haben sich weitere Problemstellungen ergeben, insbesondere im Bezug auf die Modellierung der Spielzustände. Die verwendete Modellierung beinhaltet die Historie der vergangenen Züge nicht, lediglich die Information welche Karten bereits gespielt worden. Die bedeutet, dass die Markov-Eigenschaft potenziell nicht erfüllt ist. Eine Möglichkeit diese Informationen aufzunehmen wäre eine Modellierung mit Features.

Insgesamt konnte das Grundprojekt jedoch erfolgreich genutzt werden um die Arbeitsumgebung aufzubauen und erste Schlüsse zu der Thematik zu ziehen.

4 Weitere Analysen

Ein zentraler Aspekt der geplanten Masterarbeit ist die Entwicklung eines Ansatzes für die künstlichen Intelligenz einer Skat-KI. Während sich im Grundprojekt die

Arbeit auf die Schaffung einer Arbeitsumgebung konzentriert hat, soll im Hauptprojekt nun dieser Aspekt näher betrachtet werden.

4.1 Bewertung von Verfahren

Um die Wahl einer Methode zu unterstützen soll ein systematischer Vergleich zwischen bekannten Ansätzen durchgeführt werden. In dieser Ausarbeitung werden dazu die zu analysierenden Aspekte definiert. Diese Arbeit wird dann im Hauptprojekt aufgegriffen. Die folgende Punkte sollen verglichen werden.

- *Klasse*: Zunächst müssen die Methoden klassifiziert werden um eine Übersicht zu schaffen. Vor Allem bei hybriden Verfahren muss darauf geachtet werden alle Aspekte aufzunehmen. Beispiele für Klassen sind Reinforcement Learning Verfahren oder Baumsuchen.
- *Eignung*: Die Eignung eines Verfahrens ist die Frage inwieweit die Methodik zur Problemstellung passt. Monte Carlo Verfahren sind beispielsweise nur mit episodischen Problemen sinnvoll. Dieser Aspekt ist jedoch nicht binär, da das Problem ggf. umdefiniert werden kann oder nur Teilaspekte abgedeckt sind.
- *Qualität*: Dies beschreibt der Qualität der Spielergebnisse. Dies ist einer der zentralen Punkte der Arbeit, da die Qualität dem tatsächlichen Nutzen für den Endnutzer entspricht. Um diesen Aspekt sinnvoll zu bewerten müssen voraussichtlich einige Referenzimplementierungen umgesetzt werden. Im besten Fall finden sich jedoch nutzbare Arbeiten mit gleicher oder ähnlicher Problemstellung als Vergleichswert.
- *Performance*: Der Speicherbedarf, sowie die Anzahl der Berechnungen, und damit auch die Zeit die realistisch gewartet werden muss bevor die KI agiert. Dieser Punkt steht im Verhältnis zur Qualität, da auch eine maximale Qualität keinen Nutzen bringt wenn die Performance nicht realistisch ist.
- *Voraussetzungen*: Einige Verfahren sind nur unter gewissen Voraussetzungen nutzbar. So werden eventuell Testdaten oder Expertenwissen benötigt um einen Ansatz weiter verfolgen zu können. Bei der Bewertung muss hier darauf geachtet werden inwieweit die Anforderung in der Masterarbeit erfüllt werden können.
- *Kombinierbarkeit*: Ein zusätzlicher Aspekt ist die Frage wie komplex es ist Ansätze miteinander zu kombinieren. Eine klare Qualitative Bewertung ist hier nur schwer machbar, dennoch ist es wünschenswert diese Frage jetzt schon zu betrachten um die weitere Arbeit zu vereinfachen. Eine Möglichkeit wäre zu betrachten ob ein Teilproblem durch das Verfahren gelöst werden kann.

4.2 Abstraktion

Wie bereits in Abschnitt 3 erwähnt ist ein weiterer Punkt der sich im Laufe der ersten Implementierung ergeben hat die Frage nach der Abstraktion des Spiels um den Zustandsraum einzuschränken. Dies kann durch die Definition von Features gewährleistet werden. Der Spielzustand wird dann durch die Werte der Features beschrieben.

Features auf Basis von Domänenwissen Eine Möglichkeit Features zu bestimmen ist Domänenwissen zu nutzen. Abbildung 3 zeigt die definierten Features von SkatBot [6], einer KI für Skat auf Basis von Support Vector Machines. Ein Großteil der Features ist binär gewählt, um den Featurevektor möglichst klein zu halten. Dieser Ansatz ist sehr intuitiv umzusetzen, hat allerdings den Nachteil, dass es signifikantes Fachwissen benötigt. Letztlich wird bei dieser Methode entschieden welches Wissen hervorgehoben bzw. ignoriert wird. Dies hat entsprechend große Auswirkungen auf die Qualität des Spielens [9].

Lernen von Features Alternativ können Deep Learning Verfahren verwendet werden um automatisiert Features zu generieren. Dies kann zum Beispiel durch Neural Networks umgesetzt werden. Diese Variante ist nicht von Fachwissen abhängig, ist allerdings weniger leicht zu durchschauen, da die gelernten Features in der Regel nicht mit einfachen realen Eigenschaften wie Farbe und Wert einer Karte übereinstimmen [4].

Suit features: (i = {♦, ♥, ♠, ♣})	Rank features: (j = {7, 8, 9, Q, K, 10, A, J})
1-4: # cards in suit i on hand	1: Am I playing first?
5-8: # cards in suit i remaining & not on hand	2: # points on the table
9-12: Have winning card in suit i?	3: Has opponent played?
13-16: Has opponent played a different suit on top of suit i?	4: Has teammate played?
17-20: Has teammate played a different suit on top of suit i?	5: Is my team winning?
21: Am I playing first?	6: Has opponent played a different suit on top of this suit?
22: # points on the table	7: Has teammate played a different suit on top of this suit?
23: Has opponent played?	8: How many cards of suit are remaining in game and not on hand?
24: Has teammate played?	9-19: Do I have this card j
25: Is my team winning?	20-30: Is card j a winning card?
26-29: Do I have a card with more than 10 points in suit i?	31-41: If I have card j, does card j beat opponent's card on the table?

Fig. 3. Features für Skat aus [6]

5 Zukünftige Arbeit

In diesem Kapitel soll die zukünftige Masterarbeit skizziert werden. Zunächst werden die Ziele der Arbeit vorgestellt. Danach soll das Vorgehen der Arbeit beschrieben werden. Anschließend werden erste Gedanken bezüglich eines Testszenarios vorgestellt.

5.1 Ziele

Das Ziel der gesamten Arbeit ist es einen im Kontext von Skat neuartigen Ansatz für die künstliche Intelligenz zu definieren und umzusetzen. Relevante Fragestellungen welche beantworten werden sollen sind die Übertragbarkeit von Ansätzen wie MCTS von Spielen wie Go auf Spiele mit den Eigenschaften von Skat und wie nah die Qualität des Spiels dem eines menschlichen Spielers kommt.

5.2 Bestimmung der Methode

Der erste zentrale Teil der Arbeit ist es den umzusetzenden Ansatz zu bestimmen. Dies geschieht unter Anwendung der bereits durchgeführten Analyse von möglichen Methoden. Zusätzlich ist es notwendig die Eigenschaften von Skat zu berücksichtigen. Zu betrachtende Fragestellungen wären unter Anderem:

- Kann das gleiche Verfahren für Biet- und Spielphase angewandt werden?
- Was sind Interaktionen zwischen der Biet- und Spielphase?
- Wie muss sich die Implementierung zwischen Allein- und Teamspieler unterscheiden?
- Wie wird der Spielzustand modelliert?
- Inwiefern muss zwischen den Spieltypen unterschieden werden?

Unter der Berücksichtigung dieser Fragen kann dann ein Szenario für die Arbeit bestimmt werden und ein Konzept der umzusetzenden Methode erstellt werden.

Erste Ideen Aufgrund der Struktur von Skat ist es wahrscheinlich, dass der umzusetzende Spieler einen hybriden Ansatz verfolgen wird. So ist es naheliegend, dass die Bietphase größtenteils unabhängig vom Rest des Spiels behandelt werden kann. Abbildung 4 zeigt die Grundstruktur eines möglichen Ansatzes. Das Reizens könnte mit einem Double Dummy Solver umgesetzt werden [2]. Für das Spiel wäre eine Kombination aus Neural Network und Monte Carlo Tree Search denkbar. Das NN wäre in dieser Variante das zugrundeliegende Lernverfahren, während MCTS primär als Suchverfahren eingesetzt wird.

5.3 Realisierung

Die Implementierung der Arbeit wird in Java durchgeführt. Dazu wird das JSkat Framework als Skatimplementierung verwendet. Sofern ein Ansatz gewählt wird für den Testdaten nötig sind, müssen diese geschaffen werden. In der vorherigen Analyse muss jedoch darauf geachtet werden, dass dies realistisch im Rahmen der Arbeit machbar ist.

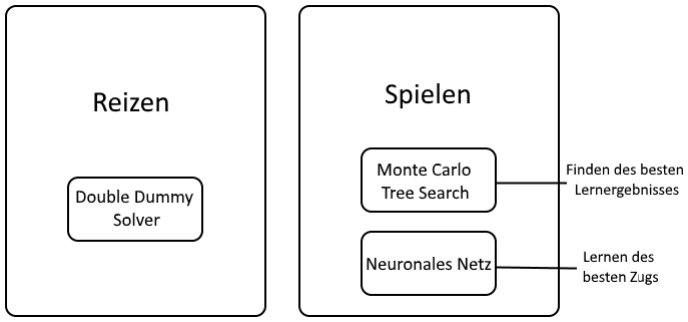

Fig. 4. Eine mögliche Struktur des Systems

5.4 Testszenarien

Nach der Umsetzung müssen Testszenarien definiert werden um anschließend Schlussfolgerungen ziehen zu können. Die Kernhypothese der Arbeit ist eine Verbesserung der Spielqualität gegenüber anderen Ansätzen. Eine Möglichkeit dies zu ermittelt wäre ein quantitativer Test. Bei einem solchen Test spielt die umgesetzte KI automatisiert gegen andere KI-Spieler. Wichtig ist hierbei, dass vor dem Test diverse Konfigurationen festgelegt werden. Mögliche Variablen wären zum Beispiel die Anzahl der Spiele, die Lerndauer vor dem Test, Ob die Bietphase miteinbezogen wird und welche KI als Gegenspieler gewählt wird. So ist es zum Beispiel wünschenswert zu ermitteln wie sich das Ergebnis zwischen 10000 und 100000 Lernduchläufen verändert und ob, und wie lange, eine solche Änderung linear ist. Es muss auch eine ausreichend große Anzahl an Tests durchgeführt werden, damit Ausreißer in der Auswertung nicht mehr signifikant sind. Im folgenden werden einige Kriterien für die Auswertung eines quantitativen Tests aufgelistet.

- *Siege/Niederlagen*: Das Verhältnis von Sieg und Niederlagen ist eine der Kernindikatoren für die Spielqualität.
- *Durchschnittliche Punktzahl*: Die Punktzahl ist nützlich, da damit die Konsistenz des Spiels bewertet werden kann. Ein Spieler, der über viele Spiele durchschnittlich 30 Punkte erreicht ist konsistenter als ein Spieler mit 20 Punkten. Zusätzlich werden knapp ausgegangene Spiele so noch miteinbezogen, während Sieg und Niederlage unabhängig sind von der Punktedifferenz.
- *Performance*: Die Performance des Spielers ist wichtig, da das Spiel in einem realistischen Szenario gespielt werden soll. Somit sollte die Auswahl des nächsten Zuges nicht signifikant länger brauchen als ein menschlicher Spieler benötigen würde. Um die Performance zu messen ist es nötig die Hardware auf der der Spieler läuft zu dokumentieren und die Zeit für die Auswahl des Zuges festzuhalten.

Zusätzlich zu einem quantitativen Test soll auch ein qualitativer Test durchgeführt werden. Dies ist nötig um einen Vergleich zu menschlichen Spielern erfolgre-

ich herstellen zu können, da es unrealistisch ist mehrere hundert Spiele gegen menschliche Spieler diverser Qualität aufzeichnen zu können. Ein quantitativer Test ist besonders für die Bewertung des Bietverfahrens nützlich. Dieses kann nicht objektiv bewertet werden, da ein gutes Spielverfahren die Nachteile von schlechten Geboten potenziell ausgleicht. Daher ist es für eine sinnvolle Evaluierung sinnvoll das Ergebnis der ersten Phase unabhängig vom Rest des Spiels zu betrachten. Hierfür wird allerdings Expertenwissen benötigt. Andere Aspekte die mit einer qualitativen Analyse betrachtet werden sollen sind zum Beispiel Unterschiede in Spielverhalten und Qualität zwischen den Spielvarianten und das Verhalten im kooperativen Spiel mit einem menschlichen Partner. Die Unterscheidung zwischen quantitativen und qualitativen Test wird explizit getroffen, da der quantitative Test automatisiert werden kann und somit eine große Anzahl von Spielen ausgewertet werden können. Der qualitative Test hat eine geringere Aussagekraft, kann dafür aber spezifischere Aspekte betrachten.

6 Risikoanalyse

Nachdem im vorherigen Abschnitt der Ablauf der weiteren Arbeit beschrieben wurde sollen nun im diesem Teil nun mögliche Hindernisse und Probleme aufgezeigt werden. Dabei soll auch betrachtet werden, wie diese Risiken minimiert werden können. Anschließend werden die Chancen, welche sich mit der Arbeit ergeben aufgelistet.

6.1 Risiken

Es wird während der Analyse kein Ansatz gefunden In diesem Fall würde sich der weitere Teil der Arbeit mit der Frage befassen warum die bekannten Methoden sich nicht auf Skat übertragen lassen.

Performanceprobleme erlauben keine umfangreichen Tests Im Fall, dass die Implementierung eine zu schlechte Performance liefert um umfangreiche quantitative Tests durchzuführen (mehrere 100 Spiele in verschiedenen Varianten) müsste das Testszenario eingeschränkt werden und sich vorwiegend auf qualitative Tests beschränkt werden. Im Fall das die Implementierung zu langsam ist um gegen sie zu spielen muss die Problematik analysiert und in der Arbeit näher beschrieben werden.

Die benötigten Ressourcen können nicht beschafft werden Im Fall, dass der ausgewählte Ansatz Testdaten oder Ähnliches benötigt und diese nicht beschafft werden können wird der Ansatz entweder abgeändert, in dem beispielsweise von einer supervised learning zu einer unsupervised Methode gewechselt wird oder wenn nicht anders möglich die Arbeit auf theoretischer Ebene mit einer Teilimplementierung durchgeführt.

Die Implementierung bietet eine schlechte Spielqualität In diesem Fall muss eine umfangreiche Analyse durchgeführt werden. Zunächst muss eine fehlerhafte Implementierung ausgeschlossen werden. Danach soll sich in diesem Fall intensiv mit der Frage beschäftigt werden warum die vorher definierten Erwartungen nicht erfüllt werden konnten.

Der zeitliche Rahmen wird gesprengt Sollte die Zeit nicht für die geplante Umsetzung ausreichen muss das Szenario eingeschränkt werden. Eine Möglichkeit wäre das Bietverfahren nicht zu betrachten. Alternativ könnte das kooperative Spielen ignoriert werden. Sollte sich der analytische Teil der Arbeit als weitaus komplexer als geplant herausstellen wird die Implementierung minimal gehalten und der Fokus auf die Entwicklung einer Methodik gelegt.

6.2 Chancen

Ein Ansatz wird entwickelt Die zu verwendeten Verfahren wie Monte Carlo Tree Search in Kombination mit anderen Methoden sind gerade sehr aktuell und noch nicht in vielen Arbeiten umgesetzt worden. Die Arbeit bietet die Chance zumindest in einem Teilaspekt der Kombination einen neuartigen Aspekt umzusetzen und zu evaluieren.

Ein Spieler wird umgesetzt Die fertige Umsetzung lässt sich als Vergleichswert für zukünftige Implementierungen verwenden.

Die Abdeckung von Skat wird verbessert Skat ist bisher nur wenig in der künstlichen Intelligenz betrachtet worden. Die Arbeit könnte einen signifikanten Beitrag dazu leisten bekannte Verfahren im Kontext von Kartenspielen wie Skat und Bridge einzuordnen.

References

1. Browne, C., Powley, E.: A survey of monte carlo tree search methods. Intelligence and AI 4(1), 1–49 (2012)
2. Chang, M.S.: Building a fast double-dummy bridge solver. Tech. rep., New York, NY, USA (1996)
3. Chaslot, G., Bakkes, S., Szita, I., Spronck, P.: Monte-carlo tree search: A new framework for game ai. In: Darken, C., Mateas, M. (eds.) AIIDE. The AAAI Press (2008)
4. Ghaderi, A., Athitsos, V.: Selective unsupervised feature learning with convolutional neural network (S-CNN). CoRR abs/1606.02210 (2016)
5. Guo, X., Singh, S.P., Lewis, R.L., Lee, H.: Deep learning for reward design to improve monte carlo tree search in ATARI games. CoRR abs/1604.07095 (2016), http://arxiv.org/abs/1604.07095
6. Leung, Sung, M.: Skatbot: Teaching a computer to play cards (2013)
7. Robilliard, D., Fonlupt, C., Teytaud, F.: Monte-Carlo Tree Search for the Game of "7 Wonders", pp. 64–77. Springer International Publishing, Cham (2014)
8. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature 529(7587), 484–489 (2016)
9. Sturtevant, N.R., White, A.M.: Feature Construction for Reinforcement Learning in Hearts, pp. 122–134. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
10. Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning. MIT Press, Cambridge, MA, USA, 1st edn. (1998)