TECHNISCHE UNIVERSITÄT DARMSTADT
Fachbereich Informatik
Fachgebiet Knowledge Engeneering
Prof. Johannes Fürnkranz

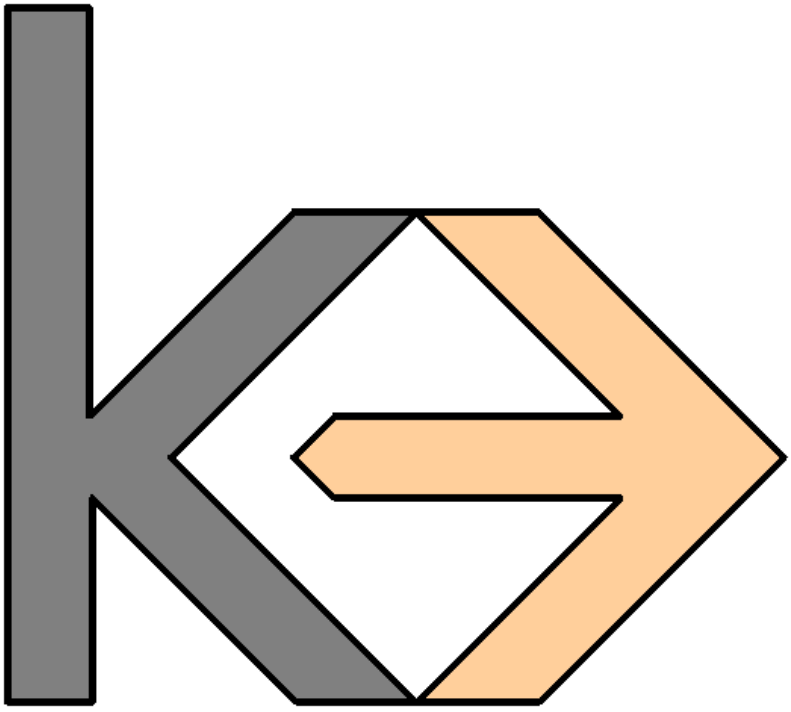
Lernen Abalone zu Spielen

Bachelorarbeit

Betreuer: Prof. Johannes Fürnkranz

Torsten Ehrhardt

Darmstadt, 21.10.2010



Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe alle Stellen, die ich aus den Quellen wörtlich oder inhaltlich entnommen habe, als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, am 20.10.2010

Abstract

This paper presents an agent for the game of Abalone. The agent uses self play and Reinforcement Learning techniques. The algorithms used are quite similar to those used in Abalearn. A Risc-Sensitive variation of Tesauro's TD(lambda) algorithm is used. The main focus is placed on finding an efficient yet expressive board representation. To this end, different combinations of popular board features are evaluated. Resulting in an extremely compact board representation. Different reward functions are tried and compared.

To evaluate the resulting agents, they are pitted against a fixed reference agent and against one another.

Inhaltsverzeichnis

1 Einführung

Atome spalten ist ein Kinderspiel, verglichen mit einem Kinderspiel.
(Albert Einstein, 1879-1955)

Der Mensch spielt nur, wo er in voller Bedeutung des Wortes Mensch ist,
und er ist nur da ganz Mensch, wo er spielt.
(Friedrich Schiller, 1795)

1.1 Warum beschäftigt man sich mit Spielen in der Wissenschaft?

Die ältesten bekannten Brettspiele wurden vor rund 5500 Jahren gespielt (Piccione 1980). Seit je her sind Brettspiele ein integraler Bestandteil menschlicher Zivilisation. Sie werden zum Zeitvertreib und zum Training abstrakter Denkstrukturen gespielt. So können Spiele ein nützliches Kognitionstraining darstellen. Die Forschung an Spielen ist ein großer Teilbereich der künstlichen Intelligenz. Die Gründe hierfür sind vielfältig. Einerseits bietet sich durch die Beliebtheit von Brettspielen ein recht großer Markt für Programme die spielen können. Andererseits ist die Beschäftigung mit Spielen nicht nur reiner Selbstzweck.

Für die Erforschung künstlicher Intelligenz bilden Brettspiele klar definierte und strukturierte Domänen. Wissen, das bei der Erforschung von Spielen gewonnen wird, lässt sich oft für die Lösungen ernsthafterer Probleme adaptieren.

1.2 Ziele dieser Arbeit

Das Hauptziel dieser Arbeit ist es mit Methoden des verstärkenden Lernens (eng. reinforcement learning) einen Agenten zu trainieren, der Abalone spielt. Dabei kann auf die Ergebnisse von Campos und Langlois (Campos und Langlois 2003), sowie von Lee und Noh (Lee und Noh 2007) zurückgegriffen werden. Während sich Campos und Langlois hauptsächlich mit der Risikobereitschaft ihres Agenten beschäftigten, untersuchten Lee und Noh die Auswirkungen unterschiedlicher Grundstellungen auf das Lernverhalten. In der vorliegenden Arbeit werden unterschiedliche Repräsentationen des Spielbrettes untersucht. Während die vorangegangenen Arbeiten sich auf eine beziehungsweise drei unterschiedliche Repräsentationen beschränkten, werden hier mehrere hundert Repräsentationen verglichen.

Zudem werden alternative Belohnungsfunktionen evaluiert und die Auswirkungen unterschiedlich schnell sinkender Lernraten auf die Lerngeschwindigkeit getestet.

1.3 Überblick

Diese Arbeit ist in sieben Abschnitte eingeteilt. Der zweite Abschnitt stellt das Spiel Abalone mit Regeln und einigen wichtigen Eigenschaften vor. Abschnitt drei be-

schäftigt sich mit existierenden Abaloneprogrammen, wobei besonderer Wert auf die jeweilig verwendete Spielbrettrepräsentation gelegt wird. Im vierten Abschnitt werden die Lernumgebung und die verwendeten Algorithmen erläutert. Abschnitt fünf beschreibt detailliert, welche Experimente durchgeführt werden. Die Ergebnisse dieser Experimente werden in Abschnitt sechs illustriert. Zum Abschluss werden diese Ergebnisse in Abschnitt sieben diskutiert.

2 Abalone

Abalone wurde 1987 von Michael Lalet und Laurent Lévi entwickelt. Es ist ein strategisches Brettspiel für zwei Personen. Seit seiner Entwicklung hat es verschiedene Preise gewonnen, darunter den "Super As d'Or" im Jahre 1989 und den "Mensa Select" im Jahre 1990. Es ist ein Nullsummenspiel mit vollständiger Information.

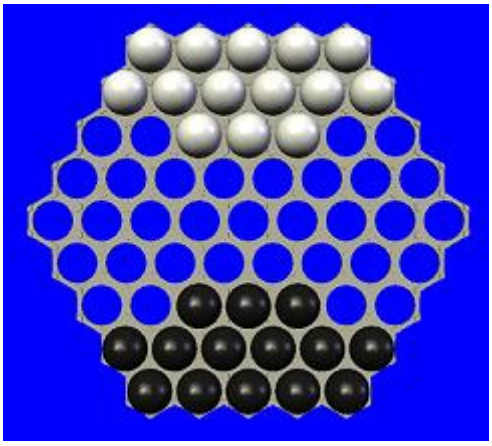


Abbildung 1: Das Spielfeld in Grundstellung

2.1 Die Regeln des Spiels

Abalone wird auf einem sechseckigen Spielfeld mit 61 Löchern gespielt(siehe Abbildung 1). Ziel des Spieles ist es mit den eigenen Murmeln die Murmeln des Gegners vom Brett zu werfen. Die Spieler ziehen abwechselnd. In jedem Zug bewegt man eine, zwei oder drei eigene Murmeln. Sieger ist, wer zuerst sechs gegnerische Murmeln vom Brett verdrängt hat.

Züge

Es ist möglich, Murmeln in alle sechs Richtungen zu bewegen. Man wählt eine, zwei oder drei eigene Murmeln, welche benachbart sind und in einer Reihe liegen. Diese kann man um ein Feld in eine beliebige (aber für alle Murmeln gleiche) Richtung bewegen.

Schieben gegnerischer Murmeln

Das Verschieben gegnerischer Murmeln ist möglich, wenn mehr eigene Murmeln bewegt werden als gegnerische. Züge in denen gegnerische Murmeln bewegt werden, nennt man Sumitos. Es gibt drei Arten von Sumitos, welche in Abbildung 2 illustriert werden.

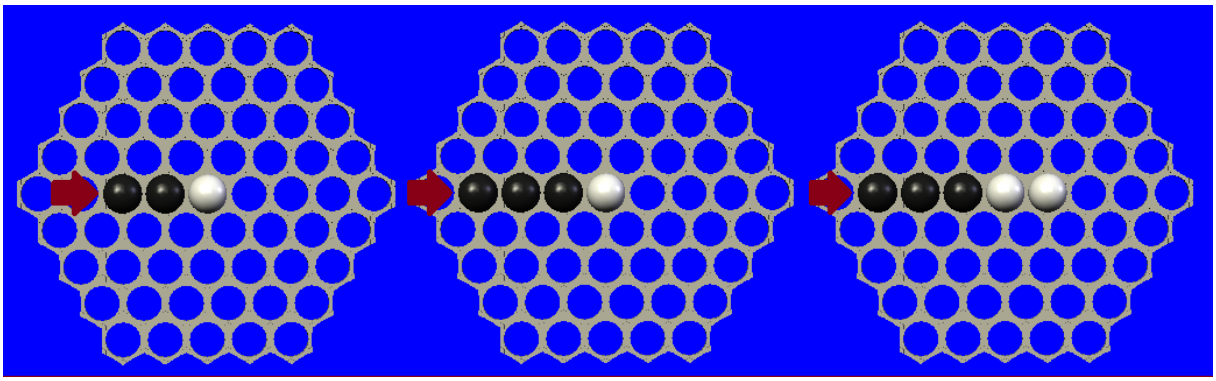


Abbildung 2: von links 2-1-Sumito, 3-1-Sumito und 3-2-Sumito

2.2 Komplexität des Spiels

Bei der Komplexität von Brettspielen spricht man von drei unterschiedlichen Faktoren. Diese werden in folgendem Abschnitt erklärt. Danach wird ein Vergleich zwischen Abalone und anderen Zwei-Spieler Nullsummenspielen mit vollständigen Informationen geführt.

- **Verzweigungsfaktor (eng. branching factor)**
  Der Verzweigungsfaktor gibt an wie viele mögliche Züge es durchschnittlich in einer Stellung gibt. Bei Abalone befindet er sich im Bereich von 60-80 (Lee und Noh 2007).
- **Zustandsraumkomplexität (eng. state space)**
  Die Komplexität des Zustandsraumes gibt an wie viele verschiedene Spielzustände erreichbar sind. Bei Abalone sind es ungefähr $6{,}5 * 10^{23}$ unterschiedliche Zustände. (Lemmens 2005)
- **Spielbaumkomplexität (eng. game tree)**
  Ein Spielbaum ist ein Baum, der Spielzustände als Knoten und Züge als Kanten darstellt. Der vollständige Spielbaum eines Spiels ist der Spielbaum, bei dem ausgehend von der Grundstellung alle möglichen Züge aus jeder erreichbaren Stellung enthalten sind. Die Anzahl der Blätter dieses Baumes wird als Spielbaumkomplexität bezeichnet. Sie gibt an auf wie viele Arten das Spiel gespielt werden kann. Da ein Abalonespiel theoretisch unendlich lange weitergeführt werden kann, muss die durchschnittliche Länge eines Spiels verwendet werden. Lemmens (Lemmens 2005) gibt die durchschnittliche Länge eines Spiels mit 87 Plys an. Daraus ergibt sich in Verbindung mit dem Verzweigungsfaktor eine Spielbaumkomplexität von ungefähr $5 * 10^{154}$.

Komplexität im Vergleich

In Tabelle 1 wird die Komplexität von Abalone verglichen mit den Komplexitäten anderer bekannter Spiele. Der Verzweigungsfaktor von Backgammon ist aufgrund der stochastischen Natur des Spiels strittig.

Es ist interessant, dass Abalone sich im Verzweigungsfaktor und in der Spielbaumkomplexität zwischen Xiangqi und Go ansiedelt. Dahingegen die Zustandsraumkomplexität noch unter der von Mühle liegt. Die relativ niedrige Zustandsraumkomplexität lässt hoffen, dass sich Abalone eines Tages durch Brute-Force lösen lässt.

Tabelle 1: Vergleich der Komplexitäten verschiedener Zwei-Spieler Nullsummenspiele mit vollständiger Information

Spiel	Verzweigungsfaktor	log(Zustandsraumkomplexität)	log(Spielbaumkomplexität)
Dame	8-10	17-21	31
Othello	~5	28	58
Schach	30-40	46	123
Backgammon	~420	20	144
Xiangqi	75	75	150
Abalone	60-80	23	154
Go	360	160-172	360

2.3 Schwäche im Spieldesign

Es existiert eine dem Spiel inhärente Schwäche im Design dieses Spiels, die dafür sorgt das passive, defensive Spieler mit einer gewissen Erfahrung selbst gegen hervorragenden Spieler ein Patt herausspielen können. Wenn in der in Abbildung 3 gegebenen Position der weiße Spieler pur defensiv spielt kann er niemals besiegt werden.

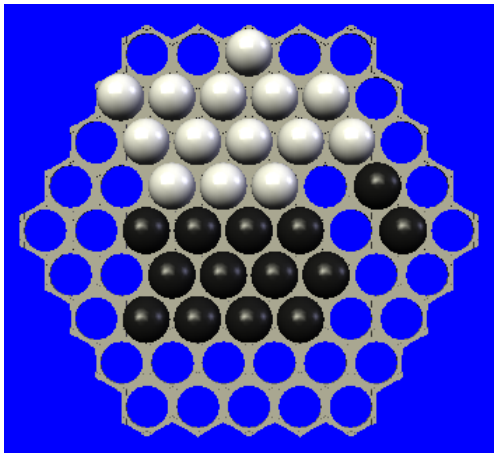


Abbildung 3: Weiß kann sich beliebig lange verteidigen

Um dieses Problem zu lösen, besteht zwischen menschlichen Spielern die Konvention, dass man nicht defensiv spielt. Da diese jedoch nicht bindend ist, gibt es verschiedene Ansätze, diese Situation zu verbessern:

1. In Turnieren kann ein Schiedsrichter Zeitstrafen für zu defensives Spiel verteilen. Es bleibt das Problem, dass "zu defensives Spiel" subjektiv ist.

2. Es gibt Varianten des Spiels, welche die Regeln ändern. In der "Save Princess" Variante wird Beispielsweise eine neutrale Murmel in die Mitte des Feldes gelegt(siehe Abbildung 4). Gewonnen hat hier, wer zuerst die neutrale Murmel vom Feld schiebt.

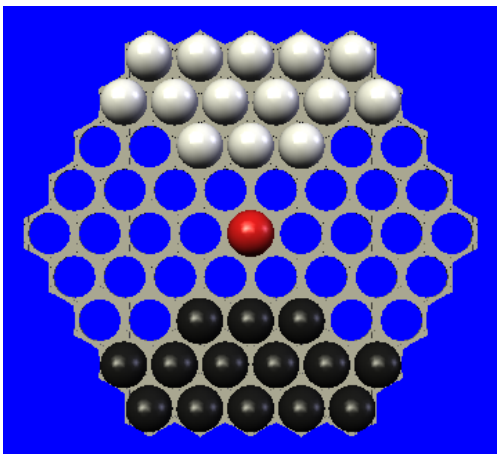


Abbildung 4: "Save Princess"-Variante

3. Es existieren verschiedene Startaufstellungen, welche aggressives Spielen fördern sollen (siehe Abbildung 5). Hierbei bleibt die Frage, ob sich dadurch ein zu starker Vorteil für den beginnenden Spieler ergibt. Es gibt Spieler die schon in der Grundstellung bemängeln, dass der beginnende Spieler in nur drei Zügen die Mitte relativ sicher erobern kann. Bei anderen Startaufstellungen können hier potentiell größere Unausgewogenheiten entstehen. Seit dem Jahr 1999 wird bei der Abalone-Mind-Sport-Olympiade die Startaufstellung „Belgian Daisy“ verwendet. Sie führt zu interessanteren Stellungen und eine passiv, defensive Spielweise führt hier nicht zum Erfolg.

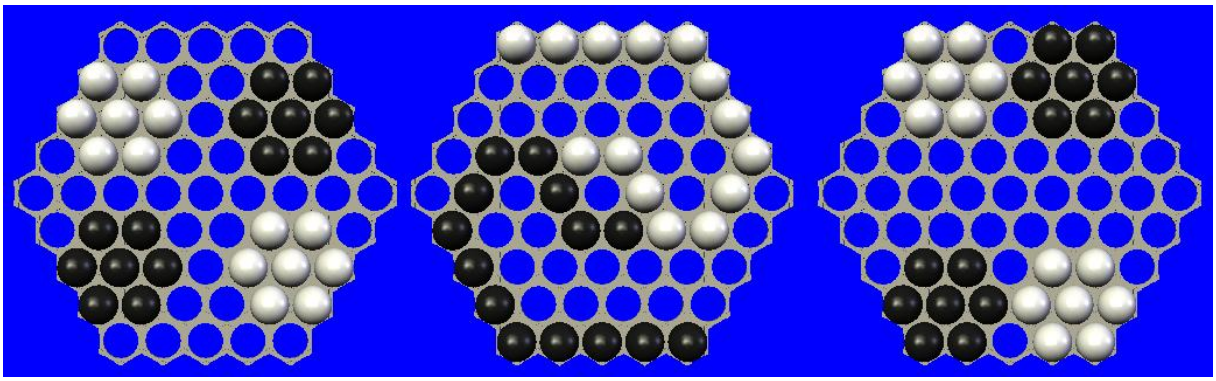


Abbildung 5: Alternative Grundstellungen von links: "German Daisy", "Snakes", "Belgian Daisy"

2.4 Strategien

Die Folgenden Strategien wurden von Ender Ozcan und Berk Hulagu (Hulagu und Ozcan 2004) aus verschiedenen Internetforen zusammen getragen.

- *Die Anzahl der zwei oder drei Murmeln in einer Reihe macht die Armee eines Spielers mächtig.*
- *Ein Hexagon von Murmeln zwischen Murmeln des Gegners erlaubt es in alle Richtungen anzugreifen und zu verteidigen.*
- *Eine Murmel an einen Ort zu ziehen, an der sie nicht an andere eigene Murmeln grenzt, ist keine gute Strategie.*
- *Die Murmeln des Gegners an den Rand des Brettes zu schieben ermöglicht es, jederzeit Rauswürfe zu erzielen.*
- *Es ist nicht zu jeder Zeit ratsam, murmeln des Gegners hinauszuwerfen. Wenn dadurch die eigene Verteidigung oder Angriffskraft gemindert wird, ist es manchmal besser zu warten.*
- *Eine möglichst starke Verteidigung ist grundlegend für den Erfolg. Mit der Zeit könnte daraus ein Angriffsvorteil entstehen, wenn der Gegner unachtsam wird.*
- *Teile und herrsche; die gegnerische Armee in mehrere Teile zu zerschlagen senkt seine Angriffsstärke. Es ist einfacher mit mehreren schwachen Armeen fertig zu werden, als mit einer starken.*

Diese Strategien lassen sich auf zwei grundlegende Prinzipien reduzieren:

1. Man behalte seine Murmeln beisammen und zwinge den Gegner seine aufzuteilen.
2. Man behalte seine Murmeln in der Mitte des Spielfeldes und zwinge den Gegner an den Rand.

3 Verwandte Arbeiten

In diesem Abschnitt werden zwei Arten von Arbeiten besprochen. Zunächst zwei der verbreitetesten Abaloneprogramme. Anschließend zwei Arbeiten welche sich mit Abalone und verstärkendem Lernen beschäftigen. Bei allen wird besonderer Wert auf die verwendete Repräsentation des Spielfeldes gelegt.

3.1 AbaPro

AbaPro von Timo Werner (2002) ist eines der spielstärksten heute verfügbaren Abaloneprogramme (Aichholzer, Aurenhammer and Werner 2002). Es verwendet eine Minimax-Suche mit heuristischem pruning. 2003 gewann AbaPro in der 8. Computer Olympiade der ICGA die Goldmedaille.

Züge, die als "uninteressant" erkannt wurden, werden im Folgenden nicht weiter betrachtet. AbaPro ist als Freeware erhältlich auf:

http://www.ist.tugraz.at/staff/aichholzer/research/rp/abalone/

Als Bewertungsheuristik wird folgende verwendet:

1. Berechne die Massenzentren der weißen und der schwarzen Murmeln.

2. Nimm einen gewichteten Durchschnitt dieser Zentren und des Brettmittelpunktes. Benenne diesen Referenzpunkt R.

3. Summiere die Distanzen aller weißen Murmeln zu R (analog für die schwarzen Murmeln). Distanzen werden entlang der sechs Zugrichtungen berechnet. Dies ist eine hexagonale Version der bekannten Manhattan Distanz.

4. Die Differenz der beiden Summen ergibt die Bewertung der Position.

3.2 MyLovelyAbalone

MyLovelyAbalone von David Malek ist ein anderes Minimaxsuche-Programm. Auf der Seite: http://moggames.net/production/bin-release/MIGS326.aspx kann man Online gegen das MLA antreten. Unter den verwendeten Bewertungsheuristiken finden sich folgende (Stand 4.12.2006):

- Adjazenz(Verbundenheit)
- Anzahl der Murmeln
- Anzahl der Sumitos
- Anzahl der Sumitos, welche den Brettmittelpunkt berühren
- Anzahl der Verdrängungsbedrohungen
- Massenschwerpunkt (im Sinne von AbaPro)

Quelle hierfür sind die Aussagen des Autors in einem Internetforum (Malek 2006).

3.3 Abalearn

Abalearn von Pedro Campos und Thibault Langlois ist die erste Arbeit, die sich mit Temporal Difference Learning und Abalone beschäftigt. Nach dem Vorbild von TDGammon verwendeten sie den TD(λ)-Algorithmus, den sie jedoch um den in "Risk-Sensitive Reinforcement Learning" (Mihatsch und Neuneier 2002) vorgestellten Risikoparameter erweiterten. Dieser Risikoparameter half der in 2.3 erwähnten Passivität entgegen zu wirken. Da der Zustandsraum von Abalone zu komplex ist, um komplett betrachtet zu werden, verwendeten sie als Bewertungsfunktion ein vorwärtsgerichtetes mehrstufiges neuronales Netz mit Backpropagation. In dieser Arbeit wurden die folgenden Brettzustandseigenschaften als Eingaben in das Netz verwendet:

1. Anzahl der Murmeln im Zentrum des Brettes
2. Anzahl der Murmeln in der Mitte des Brettes
3. Anzahl der Murmeln am Rand des Brettes
4. Materialvorteile
5. Protektion (Wie viele eigene Murmeln auf allen Seiten von eigenen Murmeln benachbart sind)
6. Durchschnittliche Distanz der Murmeln zur Mitte des Brettes
7. Anzahl der Bedrohungen

Sowohl für die weißen als auch für die schwarzen Murmeln.

3.4 Abalone Project

Diese Arbeit von Benson Lee und Hyun Joo Noh (Lee und Noh 2007) baut auf den Ergebnissen von Abalearn auf. Das Hauptaugenmerk dieser Arbeit liegt auf der Frage ob Heuristiken, welche die Symmetrie des Spielbrettes ausnutzen, wirklich angebracht sind. Sie vertraten die These, dass unter Verwendung der Standard-Grundstellung jeder Spieler die Seite auf der er beginnt anders behandeln sollte, als die auf der der Gegner startet. Ihre Ergebnisse lassen jedoch darauf schließen, dass dies keine tragende Rolle spielt.

Als Netzeingaben wurden hier folgende Brettzustandseigenschaften verwendet:

1. Durchschnittliche Manhattan Distanz der Murmeln zur Mitte des Brettes
2. Anzahl der Murmeln im Zentrum des Brettes
3. Anzahl der Murmeln in der Mitte des Brettes
4. Anzahl der Murmeln am Rand des Brettes
5. Anzahl der bedrohten Murmeln. Eine Murmel gilt als bedroht, wenn sie im nächsten gegnerischen Zug verdrängt werden kann
6. Kompaktheit (Summe der Manhattan Distanzen zwischen den Murmeln eines Spielers)
7. Protektion

4 Verwendete Algorithmen

In dieser Arbeit wird eine in Abalearn (Campos und Langlois 2003) vorgestellte Variante von Tesauros TD(λ) verwendet. Sie unterscheidet sich vom ursprünglichen TD(λ)-Algorithmus darin, dass ein zusätzlicher Parameter für die Risikofreudigkeit des Agenten hinzugefügt wird.

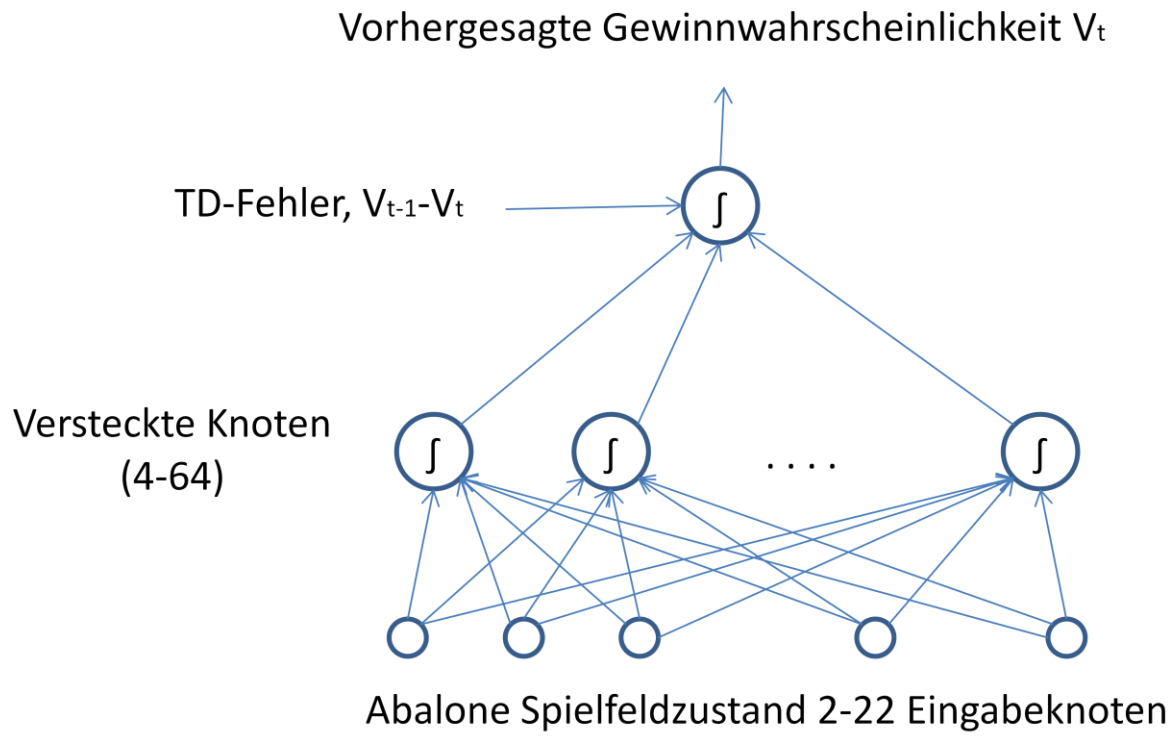


Abbildung 6: Das Neuronale Netzwerk sowie die maximale und die minimale Anzahl an verwendeten Knoten für die Eingabe- und Ausgabeschicht

4.1 Bewertungsfunktion

Mit Hilfe eines neuronalen Netzes eine Bewertungsfunktion V(s) gelernt. Eine perfekte Bewertungsfunktion gäbe für jeden Spielzustand die Gewinnwahrscheinlichkeit des Spielers an. Da das Spiel, wie in 2.2 gezeigt, ca. 10^23 Zustände hat, ist es nicht möglich diese Funktion in Form einer Tabelle darzustellen. Stattdessen wird ein neuronales Netz verwendet, um die Bewertungsfunktion zu approximieren. Dies führt zu gewissen Schwierigkeiten. Im Falle, dass ein nichtlinearer Funktionsapproximator (wie ein neuronales Netz) verwendet wird, ist die Konvergenz von TD(λ) nicht garantiert.

4.2 Neuronales Netzwerk

Um die Bewertungsfunktion anzunähern wird ein vorwärtsgerichtetes Neuronales Netzwerk mit einer versteckten Schicht und Backpropagation verwendet. Die versteckte Schicht umfasst zwischen acht und zweiunddreißig Neuronen. Als Aktivie-

rungsfunktion der versteckten Schicht wird die Sigmoidfunktion verwendet. Für die Ausgabeschicht hingegen eine lineare Funktion. Alle Gewichte werden mit zufälligen Werten zwischen -0,01 und 0,01 initialisiert.

4.3 Bestärkendes Lernen

Bestärkendes Lernen ist ein Teilbereich des maschinellen Lernens. Anders als bei überwachtem oder unüberwachtem Lernen wird erst nach mehreren Aktionen des Agenten Feedback gegeben. Gelernt wird im Allgemeinen eine Strategie, die jedem Zustand eine entsprechende Aktion zuweist. Dabei wird versucht, die Belohnungen zu maximieren. Im Vorliegenden Fall wird jedoch nicht direkt eine Strategie gelernt, sondern eine Bewertungsfunktion:

$$Strategie: \pi(s) = a$$
$$Bewertungsfunktion: V(s) = x$$

Dabei ist s ein Spielzustand, a eine in s verfügbare Aktion und x die Bewertung der Stellung in Zustand s. Die Verwendete Strategie ergibt sich aus der Bewertungsfunktion. Es wird diejenige Aktion ausgewählt, deren Folgezustand die beste Bewertung hat.

4.4 TD(λ)

Der TD(λ) Algorithmus wurde von Richard S. Sutton (Sutton und Barto 1998) entwickelt und in TD-Gammon mit großem Erfolg eingesetzt. Es handelt sich hierbei um einen Algorithmus bestärkenden Lernens. Er bietet eine Vorschrift, nach der die Bewertungsfunktion gelernt werden kann. Die Änderungen der Bewertung eines Zustands hängen dabei von der Differenz, seiner Bewertung und der Bewertung der später im Spiel auftretenden Zustände ab. Die Vorhersage der Gewinnwahrscheinlichkeit aus einem Zustand heraus, hängt dementsprechend von den Vorhersagen der folgenden Zustände ab.

Die Änderungen der Bewertungsfunktion ergeben sich für den Fall $\lambda = 0$:

$$V_t(s_t) = V_{t-1}(s_t) + \alpha[R(s_t, a) + \gamma V_{t-1}(s_{t+1}) - V_{t-1}(s_t)]$$

Dabei sind:

- V_t die Bewertungsfunktion zur Zeit t
- s_t der Zustand zur Zeit t
- α die Lernrate $0 < \alpha \leq 1$
- a der Zug, der von s_t nach s_{t+1} führt
- $R(s, a)$ die Belohnung für den Zustand s
- γ der Discount-Faktor mit $0 < \gamma \leq 1$

Die Lernrate ist ein Parameter, welcher angibt, wie stark die Gewichte bei einem Update geändert werden.

Die Belohnungsfunktion R gibt eine Belohnung von 1 für das Verdrängen gegnerischer Murmeln und -1 für den Verlust einer eigenen Murmel zurück. In allen anderen Fällen gibt sie 0 zurück.

$$R(s,a) = \begin{cases} 1, & Zug\ a\ verdrängt\ gegnerische\ Murmel \\ -1, & die\ Reaktion\ auf\ a\ verdrängt\ eigene\ Murmel \\ 0, & sonst \end{cases}$$

Der Discount-Faktor zollt der Tatsache Rechenschaft, dass Zustände in der Zukunft nicht zwangsweise erreicht werden müssen. Daher sind die Belohnungen in der Zukunft unsicher und werden durch γ abgewertet. Für γ wird ein Wert von 0,9 verwendet.

Im Fall $\lambda > 0$ wird nicht nur der nächste Folgezustand zum Update der Bewertungsfunktion herangezogen, sondern auch weiter in der Zukunft liegende Folgezustände. Der Parameter λ gibt dabei an, wie stark weiter entfernt liegende Zustände in die Berechnung eingehen.

4.5 TD(λ) mit Risikofaktor

Der von Mihatsch und Neuneier (Mihatsch und Neuneier 2002) vorgestellte und von Campos und Langlois erstmals in (Campos und Langlois 2003) verwendete Risikoparameter soll helfen die Risikobereitschaft des Agenten zu steuern. Zur Berechnung wird die folgende sogenannte Tranformationsfunktion verwendet:

$$\chi^{\kappa}: x \rightarrow \begin{cases} (1-\kappa)x\ , wenn\ x > 0 \\ (1+\kappa)x\ \ , sonst \end{cases}$$

Sie wird verwendet um die temporalen Abweichungen entsprechend zu transformieren. Die entsprechend variierten Bewertungsfunktionsupdates sehen wie folgt aus:

$$V_t(s_t) = V_{t-1}(s_t) + \alpha \boldsymbol{\chi^{\kappa}}[R(s_t) + \gamma V_{t-1}(s_{t+1}) - V_{t-1}(s_t)]$$

Daraus folgt, dass für $\boldsymbol{\kappa} < 0$ Züge mit negativen temporalen Abweichungen überbewertet werden. Während für $\boldsymbol{\kappa} > 0$ die Züge mit positiver temporaler Abweichung überbewertet werden. Das bedeutet, dass ein Agent der mit $\kappa < 0$ trainiert wird risikofreudig ist, während ein $\kappa > 0$ zu einem Risiko vermeidenden Verhalten führt.

4.6 Exploration

Wenn gierig immer die bestbewerteten Züge gewählt werden, dann wird immer das gleiche Spiel gespielt und keine neuen Zustände werden betrachtet. Deshalb wird ein Explorationsfaktor ε genutzt. In jedem Zug wird mit einer Wahrscheinlichkeit von ε ein zufälliger Zug gewählt. Mit einer Wahrscheinlichkeit von 1- ε der bestbewertete.

4.7 Self-Play

Self-Play ist die Bezeichnung einer Trainingstechnik. Dabei werden Spiele, aus denen gelernt wird erzeugt, indem der Agent gegen sich selbst spielt.

Dies hat den Vorteil, dass er sich nicht auf einen festen Gegner einstellt und nicht nur versucht dessen Strategie zu schlagen. Ein Nachteil ist die Tatsache, dass zu Beginn beide Seiten kein Wissen über das Spiel besitzen und rein zufällig spielen. Das kann zu längeren Trainingszeiten führen.

4.8 Referenzspieler

Um die Qualität der resultierenden Agenten bewerten und vergleichen zu können, bedarf es eines festen Referenzpunktes. Er muss ein gewisses Spielniveau erreichen, was einen zufällig spielenden Agenten ausschließt. Deshalb wird in dieser Arbeit ein schneller heuristischer Minimaxspieler verwendet. Er sucht bis in eine Tiefe von vier Plys. Dabei werden in jedem Schritt nur die drei bis dahin besten Züge weiterbetrachtet. Zur Bewertung wird eine optimierte Linearkombination zweier sehr einfacher Heuristiken verwendet:

- Die Verdrängungsheuristik, die 1 ist wenn eine Murmel vom Brett geschoben wird und sonst 0.
- Die Mächtigkeitsheuristik, die die Anzahl der bewegten Murmeln angibt.

Haben verschiedene Züge dieselbe Bewertung, so erfolgt die Auswahl zufällig.

4.9 Spielzustandsrepräsentation

Es ist für effizientes Lernen unabdingbar eine gute Repräsentation des Spielfeldes zu haben. Die intuitivste Repräsentation ist wohl jene, die ein Eingangsneuron für jedes Feld des Spielbrettes besitzt und bei der die Belegung eines Neurons eins ist für schwarze, minus eins für weiße und null für ein leeres Feld. Diese Repräsentation wurde in Abalearn (Campos und Langlois 2003) untersucht. Leider lernt sie nicht besonders gut. Es werden deshalb alternative Repräsentationen gesucht, die auch in der Lage sind die Symmetrien des Spielfeldes auszunutzen. Die hier Verwendung findende Zusammenstellung von Heuristiken zur Zustandsbeschreibung ist eine Auswahl aus den Heuristiken, die von Abapro, My Lovely Abalone, Abalearn sowie dem Projekt von Lee und Noh verwendet werden.

Die folgenden elf Heuristiken werden verwendet:

1. ***Zweier*:** Anzahl der nebeneinanderliegenden Paare von eigenen Murmeln
2. ***Dreier*:** Anzahl des Vorkommens von drei eigenen Murmeln in einer Reihe
3. ***Zentrum*:** Anzahl der Murmeln im Zentrum des Brettes. Manhattan Distanz von höchstens eins zum Mittelpunkt.
4. ***Rand*:** Anzahl der Murmeln am Rand des Brettes.
5. ***Verdrängungen*:** Anzahl verdrängter Murmeln.
6. ***Geschützte*:** Anzahl der auf allen Seiten von eigenen Murmeln umgebenen Murmeln.
7. ***Isolierte*:** Anzahl der von allen anderen eigenen Murmeln isolierten Murmeln.
8. ***Sumitos*:** Anzahl der möglichen Sumitos.
9. ***Bedrohungen*:** Anzahl der Möglichkeiten im nächsten Zug eine gegnerische Murmel vom Brett zu verdrängen.
10. ***Distanz zur Mitte*:** Durchschnittliche Distanz der Murmeln zum Mittelpunkt des Brettes.
11. ***Zusammenhalt*:** Durchschnittliche Distanz der Murmeln zum Schwerpunkt der Murmeln.

5 Experimentbeschreibung

5.1 Parameterauswahl

Zunächst werden Optimalwerte für einige Parameter bestimmt. Die betrachteten Parameter sind:

1. *Größe der versteckten Schicht*
2. *Explorationsrate*
3. *Lernrate*
4. *Lambda*
5. *Risikoparameter*

In Tabelle 2 werden die verschiedenen getesteten Belegungen illustriert. Die Auswahl möglicher Belegungen orientiert sich an den Werten, die von Lee und Noh (Lee und Noh 2007) als optimal gemeldet wurden. Um die optimalen Belegungen zu finden werden für jede der 243 Kombinationen drei Durchläufe zu je 1500 Spielen durchgeführt.

Verwendete Heuristiken sind hierfür:

1. *Anzahl der Murmeln im Zentrum des Brettes*
2. *Anzahl der Murmeln in der Mitte des Brettes*
3. *Anzahl der Murmeln am Rand des Brettes*
4. *Anzahl verdrängter Murmeln*
5. *Durchschnittliche Distanz der Murmeln zur Mitte des Brettes*
6. *Durchschnittliche Distanz der Murmeln zum Schwerpunkt der Murmeln*

Tabelle 2: Parameterwerte

Parameter	Verschiedene Belegungen		
Epsilon	0,01	0,05	0,2
Lernrate	0,1	0,05	0,005
Lambda	0	0,35	0,7
Größe der versteckten Schicht	8	16	32
Risikoparameter	0	-0,35	-0,7

Gespielt wird mit einem Discount-Faktor von 0,9. Um das Lernen zu Beginn zu beschleunigen, werden die ersten 100 Spiele gegen einen zufälligen Spieler gespielt. Dies dient hauptsächlich dazu, zu lernen Murmeln vom Brett zu verdrängen.

5.2 Bewertung der Zustandsbeschreibungen

Ziel dieses Experimentes ist es, eine Bewertung des Nutzens der einzelnen Heuristiken zu erhalten. Aus diesem Grund wird die Potenzmenge der Menge der Heuristiken durchsucht. Diese enthält $2^{11} = 2048$ Elemente. Aus Zeitgründen können nicht alle 2048 Agenten trainiert werden. Es wird einerseits die Auswirkung des Weglassens einzelner Heuristiken, sowie von Paaren von Heuristiken untersucht.

Die verwendeten Parameter entsprechen dem Ergebnis aus 5.1. Jeder Agent wird mit 1000 Spielen trainiert. Um die Agenten vergleichbar zu machen, wird jeder von ihnen in 100 Spielen gegen den Referenzspieler getestet.

5.3 Variationen der Belohnungen

Alle bisherigen Untersuchungen wurden mit einer Belohnung von +1 für das vom Brett Verdrängen einer gegnerischen Murmel und -1 für einen Murmelverlust vergeben. Das Spiel endet aber erst nachdem sechs Murmeln eines Spielers das Spielbrett verlassen haben. Das sofortige Verdrängen gegnerischer Murmeln ist nicht immer von Vorteil. Manchmal ist es besser, die eigene Position zu festigen. Gleichermaßen ist es nicht zwingend nützlich, eine Murmel um jeden Preis zu beschützen. Es wäre daher besser nur dann Belohnungen und Strafen zu verteilen, wenn das Spiel auch endet. In (Campos und Langlois 2003) stellen Campos und Langlois die These auf, dass es zu schwierig sei, einen Agenten mit derart wenig Feedback zu trainieren. Diese These wird hier untersucht. Indem Agenten die direkt nach verdrängen einer Murmel Feedback erhalten mit solchen verglichen werden, die erst am Ende des Spiels belohnt oder bestraft werden.

Darüber hinaus werden unsymmetrische Belohnungen betrachtet. Beispielsweise +2 für das Verdrängen gegnerischer und -1 für den Verlust eigener Murmeln. Eine These ist, dass eine entsprechend höhere Gewichtung der Belohnung gegenüber der Bestrafung den Risikoparameter überflüssig machen könnte.

Es wird eine Reihe von Agenten trainiert, für verschiedene Feedback-Funktionen und Werte von κ.

Tabelle 3: Experimente zu variierten Belohnungen

Nummer	Belohnung für Verdrängen	Strafe für verdrängt werden	Belohnung für das Gewinnen	Strafe für das Verlieren	Risikoparameter κ
1	1	-1	0	0	-0,7
2	1	-1	0	0	0
3	3	-1	0	0	-0,7
4	3	-1	0	0	0
5	1	-3	0	0	-0,7
6	1	-3	0	0	0
7	0	0	1	-1	-0,7
8	0	0	1	-1	0
9	0,5	-0,5	1	-1	-0,7
10	0,5	-0,5	1	-1	0

5.4 Vergleich mit anderen Abaloneprogrammen

Bislang wurden alle Agenten nur gegeneinander oder gegen den Referenzspieler geprüft. Dadurch konnte nur relative Spielstärken ermittelt werden. Um sich einer absoluten Bewertung der Spielstärke anzunähern wird ein Agent der die Ergebnisse aller vorherigen Experimente nutzt trainiert. Dieser wird gegen MyLovelyAbalone und Aba-Pro spielen.

6 Ergebnisse

6.1 Parametertests

Jede Parameterbelegung wurde mit 100 Spielen gegen den Referenzspieler getestet. Als Bewertungsmaß wurden nicht die gewonnen Spiele, sondern die Differenz der gewonnenen und verlorenen Murmeln herangezogen. Die besten 10% davon werden weiter betrachtet. Tabelle 4 zeigt das Auftreten der einzelnen Belegungen nach Variablen.

Tabelle 4: Häufigkeit von Parameterbelegungen in den besten 10% der Agenten

Parameter	Belegung	Vorkommen
versteckte Knoten n	8	4
	16	11
	32	7
Explorationsrate ε	0,01	10
	0,05	7
	0,2	5
Lernrate α	0,005	4
	0,05	12
	0,1	6
λ	0	5
	0,35	8
	0,7	9
Risikoparameter κ	0	5
	-0,35	6
	-0,7	11

Diese besten 10% gelernten Agenten spielten ein Turnier in dem jeder in 100 Spielen gegen jeden antrat. Die Gewinner des Turniers werden in

Tabelle **5** vorgestellt.
Jede gewonnene Murmel brachte einen Punkt jede verlorene einen Punkt Abzug.

Tabelle 5: Die fünf Topplatzierten des Turnieres zur Parameterwahl

Platzierung	Punkte	n	E	α	Λ	K
1.	3052	8	0,01	0,1	0,7	-0,7
2.	2922	16	0,01	0,005	0,7	-0,7
3.	2653	16	0,05	0,05	0,35	-0,7
4.	2507	16	0,05	0,1	0,35	-0,35
5.	1906	16	0,2	0,05	0,7	0

Interpretation der Ergebnisse:

- *Größe der versteckten Schicht n:*
  Die Ergebnisse bezüglich der versteckten Schicht lassen sich damit erklären, dass eine Trainingsdauer von 1500 Spielen wahrscheinlich nicht ausreicht um 32 versteckte Knoten anzupassen. Da vier der fünf besten Agenten 16 versteckte Knoten verwenden, ist wird diese Anzahl weiter verwendet.
- *Explorationsfaktor ε:*
  Ein Wert von 0,2 scheint zu hoch zu sein. Es wird ein Wert von 0,01 gewählt, da dieser sowohl in den besten 10% am Häufigsten auftritt, als auch jener ist, den die beiden Erstplatzierten verwenden.
- *Lernrate α:*
  Die Ergebnisse lassen vermuten, dass eine Lernrate von 0,005 nicht geeignet ist, um schnell zu lernen. Daher wird ein Wert von 0,05 gewählt um die Lerngeschwindigkeit zu erhöhen.
- *TD Parameter λ*:
  Es zeigt sich, dass eine Verwendung von $\lambda > 0$ offenbar Sinn macht. Der Wert von 0,7 bietet sich dafür an.
- *Risikoparameter κ:*
  Eine Belegung mit $\kappa = -0{,}7$ hat die besten Ergebnisse gebracht und wird somit genutzt.

6.2 Auswahl der Heuristiken

Zunächst wurde ein Agent mit allen verfügbaren Heuristiken trainiert. Abbildung 7 zeigt seine Performanz gegen den Referenzspieler im Verlauf der 1000 Trainingsspiele.

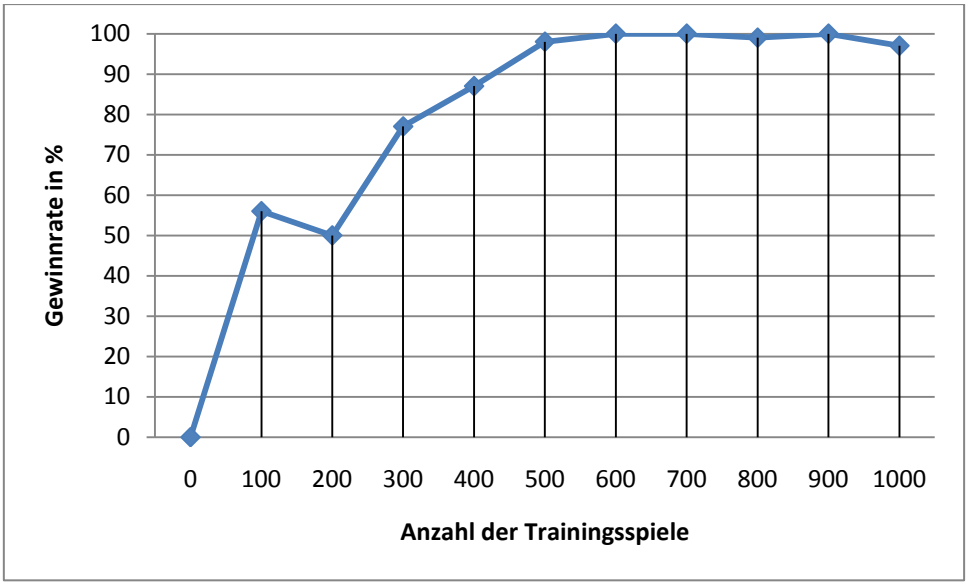


Abbildung 7: Anteil der Siege eines Agenten, der mit allen Heuristiken trainiert wurde, gegen den Referenzspieler

Nun wurden Agenten trainiert, die einzelne Heuristiken nicht verwendeten. Abbildung 8 zeigt sehr deutlich, dass die einzelnen Heuristiken in dieser Menge unterschiedlich viel zum Sieg beitrugen. Der Agent, der ohne *Dreier*-Heuristik gelernt wurde, gewann zu keinem Zeitpunkt während des gesamten Trainings mehr als 50% der Spiele. Hingegen scheint ein Weglassen der *Zweier*-Heuristik das Lernen eher zu beschleunigen.

Bei der Zusammenhalt-Heuristik ist das ähnlich. Eine mögliche Erklärung hierfür ist, dass verschiedene Heuristiken ähnliche Eigenschaften abdecken. Beispielsweise sind sowohl *Geschützte* als auch Zusammenhalt Maße für das Zusammenhalten der eigenen Murmeln.

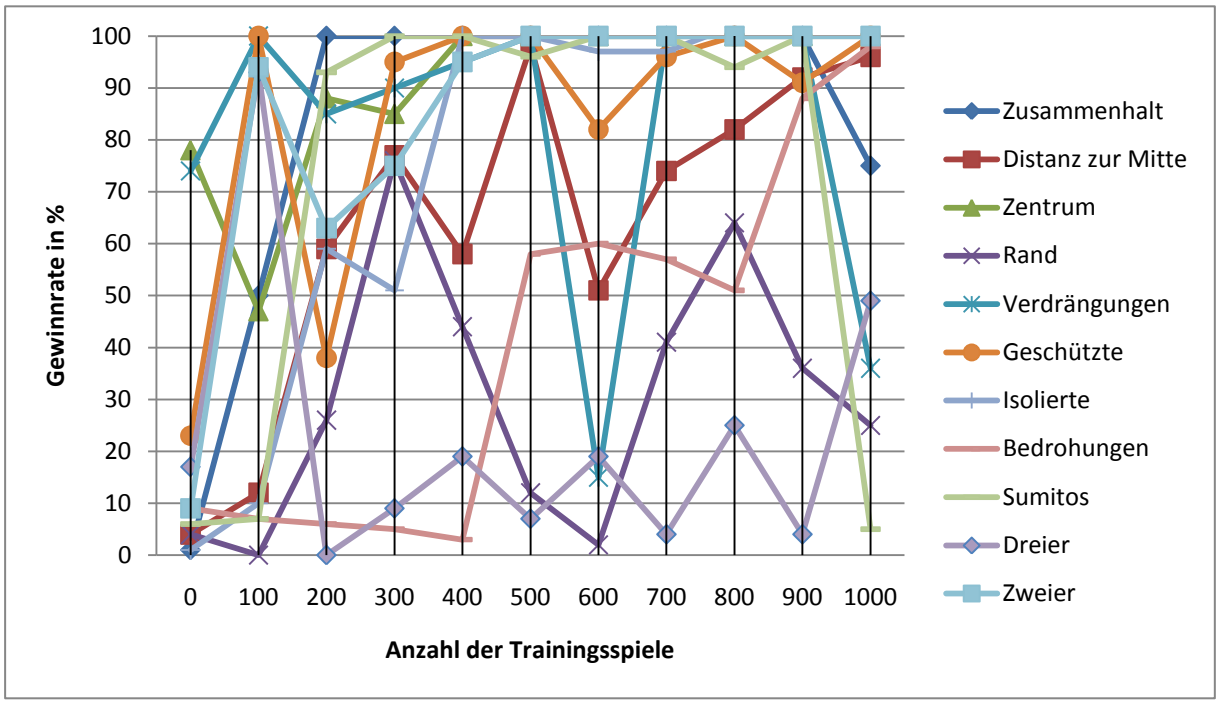


Abbildung 8: Gewinnraten der Agenten, die ohne eine spezielle Heuristik trainiert wurden

Wenn man Agenten mit nur neun der verwendeten Heuristiken lernen lässt, gewinnt man Informationen, darüber welche Paare von Heuristiken wichtig sind. Im Umfang dieser Ausführung ist es nicht möglich alle Ergebnisse vorzustellen, deshalb werden nur die interessantesten ausgewählt.

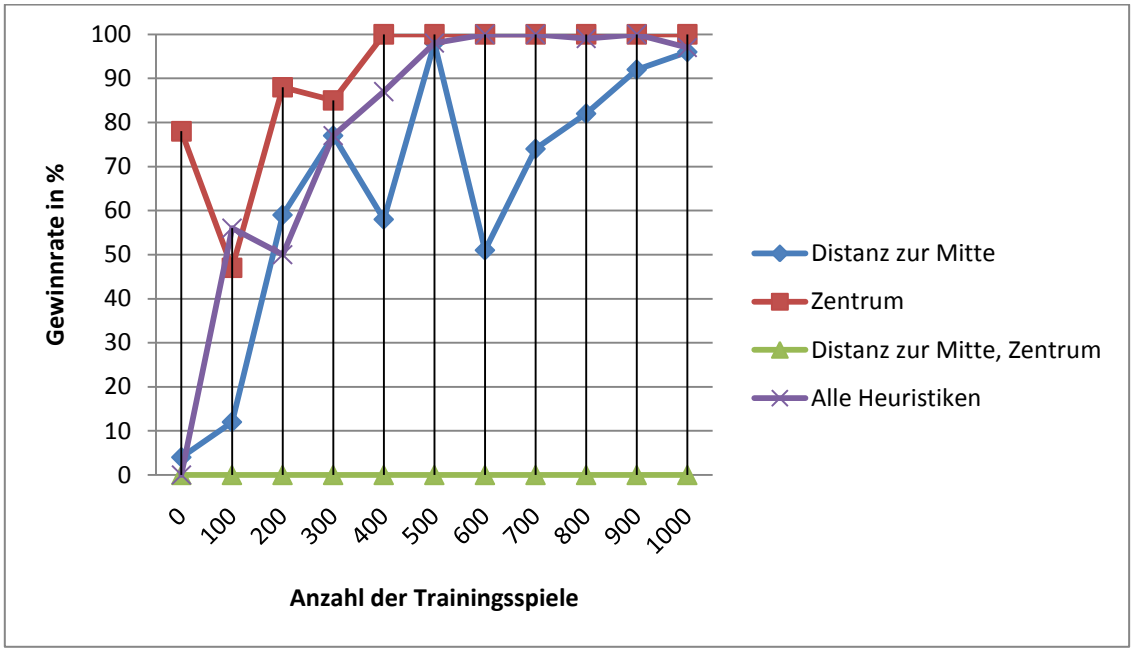


Abbildung 9: Änderungen im Lernverhalten durch *Distanz zur Mitte* und *Zentrum*

Abbildung 9 zeigt beispielhaft, dass weder *Zentrum* noch *Distanz zur Mitte* benötigt werden um den Spielzustand angemessen zu repräsentieren. Ein Fehlen beider Heu-

ristiken führt jedoch dazu, dass keine nennenswerten Lernerfolge zu verzeichnen sind. Ähnlich sieht es bei *Bedrohungen* und *Sumitos* aus, siehe Abbildung 10. Als Begründung kann dienen, dass sowohl Zentrum als auch Distanz zur Mitte Aussagen darüber treffen wer die Mitte beherrscht. Diese Information ist in keiner anderen Heuristik enthalten. Bei Sumitos und Bedrohungen ist die Sache noch eindeutiger, da jede Bedrohung auch ein Sumito ist.

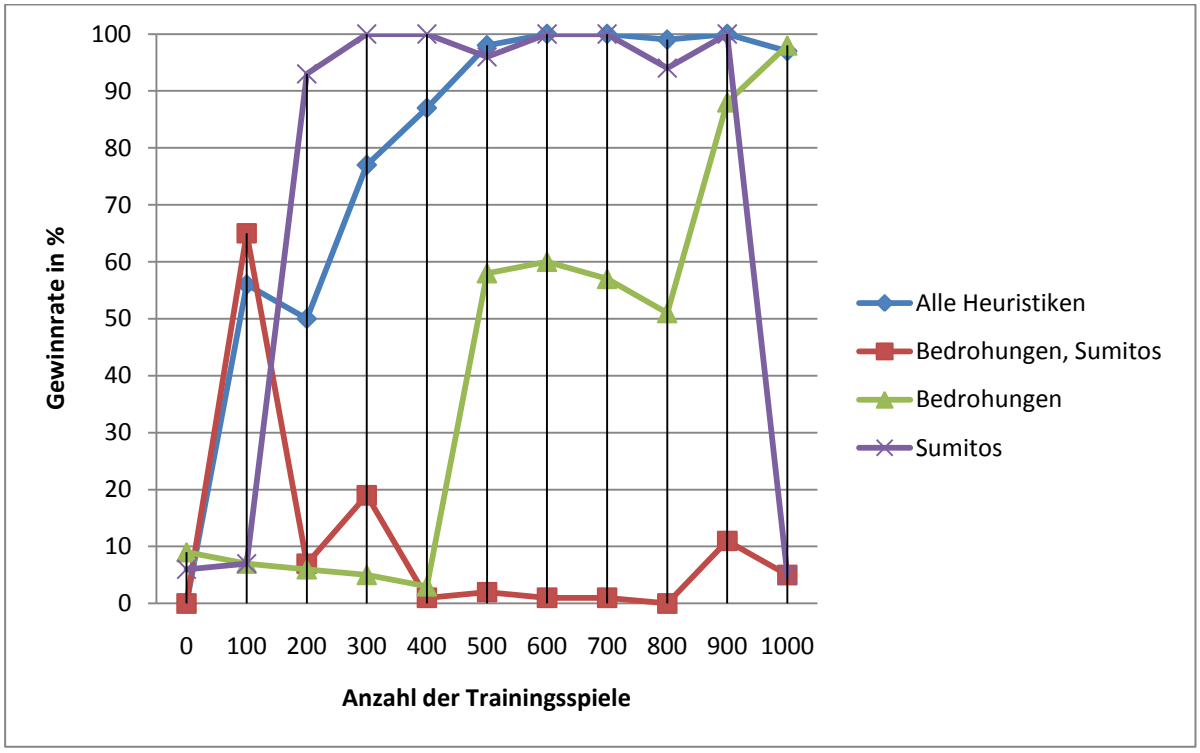


Abbildung 10: im Lernverhalten durch *Bedrohungen* und *Sumitos*

Mit den hier gewonnenen Informationen wird ein neuer Agent mit einer schlankeren Spielfeldrepräsentation gelernt. Aufgrund der genannten Überlegungen wird vermutet, dass die Lernfähigkeit durch das Entfernen unwichtiger Heuristiken nicht sinkt. Der Berechnungsaufwand wird jedoch deutlich geringer.

Jede weggelassene Heuristik beschleunigt die Evaluation eines Spielzustands.

Aufgrund der zuvor dargestellten Ergebnisse wird eine neue Auswahl von Heuristiken getroffen. In jedem Fall verwendet werden jene Heuristiken, die für sich genommen die Lerngeschwindigkeit positiv beeinflussen. Diese sind *Rand*, *Dreier*, *Bedrohungen*. Desweiteren konnte gezeigt werden, dass je eine von *Distanz zur Mitte* und *Zentrum*, sowie *Bedrohungen* und *Sumitos* verwendet werden sollte. Gewählt werden *Distanz zur Mitte* und *Bedrohungen*, da sie anscheinend mehr Einfluss auf das Ergebnis haben.

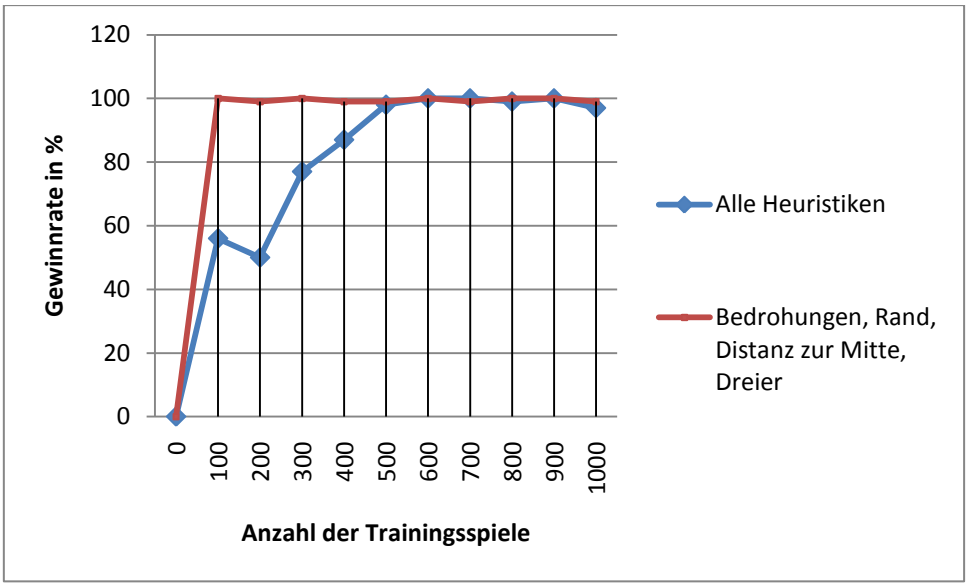


Abbildung 11: Vergleich der Gewinnraten gegen den Referenzspieler

Der Agent, der aus dieser Auswahl gelernt wurde, wird in Abbildung 11 mit dem Agenten verglichen, der alle Heuristiken verwendet. Es ist zu sehen, dass der Agent mit der kleineren Auswahl an Heuristiken schneller lernt. Da jedoch beide relativ schnell einen Punkt erreichen an dem sie beinahe nicht mehr vom Referenzspieler besiegt werden, ist dieser Vergleich nicht besonders aussagekräftig. Deshalb zeigt Abbildung 12 das Ergebnis, wenn die beiden Agenten direkt gegeneinander spielen.

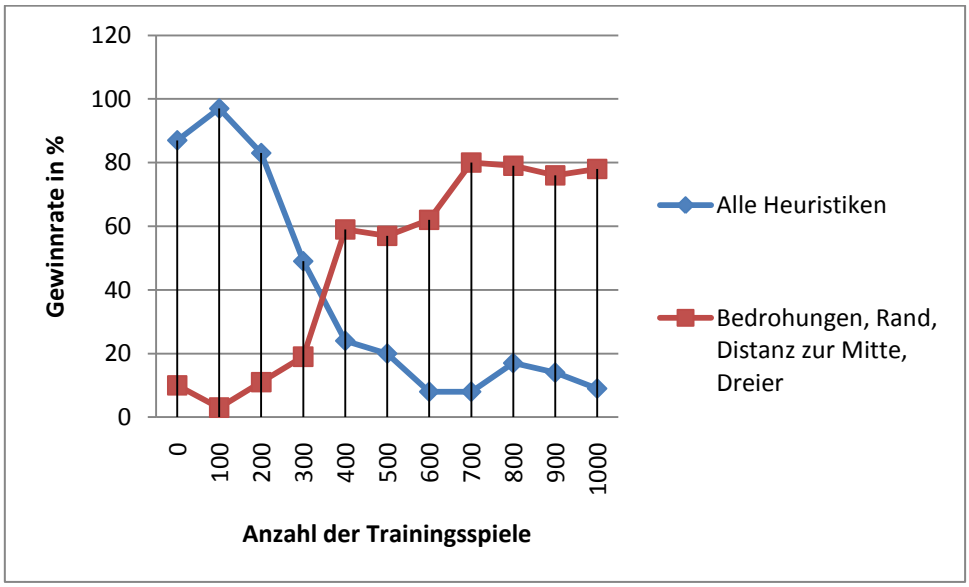


Abbildung 12: Direkter Vergleich zwischen dem Agenten mit allen Heuristiken und dem mit *Rand*, *Dreier*, *Bedrohungen* und *Distanz zur Mitte*

6.3 Variationen der Belohnungen

Für jedes Experiment wird ein Agenten trainiert. Jeder Agent trainiert mit 2500 Spielen. Getestet wird jeder Agent mit 100 Spielen gegen den Referenzspieler. Abkürzend lassen sich Belohnungsfunktionen schreiben als:

$$(x, y, a, b)$$

Mit

x: Belohnung für Verdrängen
y: Strafe für Murmelverlust
a: Belohnung für Gewinn des Spiels
b: Strafe für Verlust des Spiels

(+1,-1,+1,-1) repräsentiert beispielsweise die bislang verwendete Funktion. Abbildung 13 zeigt noch einmal, dass unter Verwendung der bislang verwendeten Belohnungsfunktion, der Risikoparameter nützlich ist.

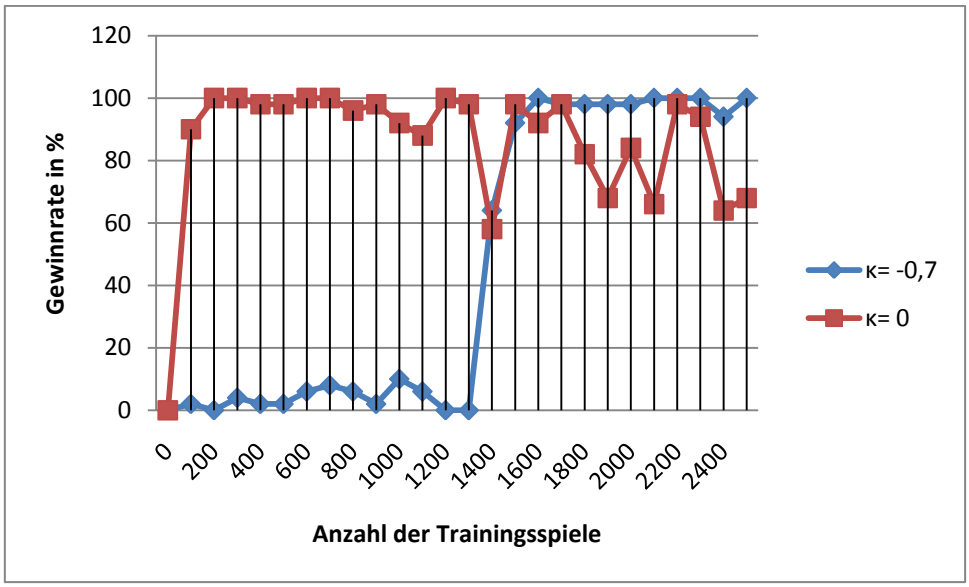


Abbildung 13: Verwendung von (+3,-1,+3,-1)

Abgesehen davon, dass der Risikoparameter für (+3,-1,+3,1) überflüssig ist, halten sich die Unterschiede symmetrischer und asymmetrischer Belohnungsfunktionen in Grenzen, siehe Abbildung 14.

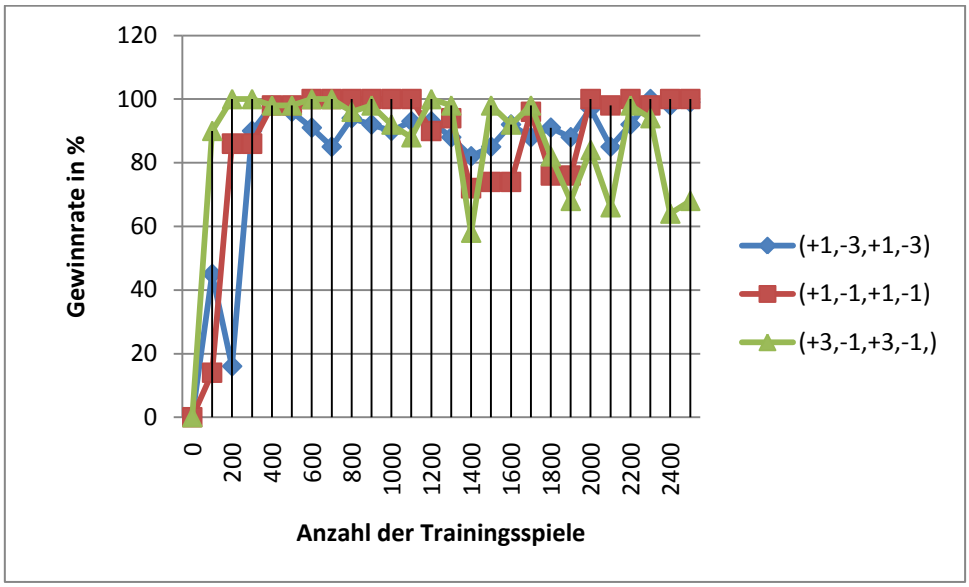


Abbildung 14: Vergleich symmetrischer und asymmetrischer Belohnungen

In Abbildung 15 ist zu sehen, dass es möglich ist zu lernen, selbst wenn erst am Ende des Spiels belohnt wird. Allerdings stimmt es auch, dass Belohnungen, die direkt nach dem Gewinn oder Verlust einzelner Murmeln gegeben werden, zu schnellerem und stabilerem Lernen führen. Dieses Ergebnis wird jedoch noch übertroffen, wenn man beides kombiniert, für den Gewinn oder Verlust des Spiels wird jedoch eine höhere Belohnung beziehungsweise Strafe verteilt. Der Agent, der mit (+0.5,-0.5,+1,-1) trainiert wurde, lernte schneller und stabiler als seine Konkurrenten.

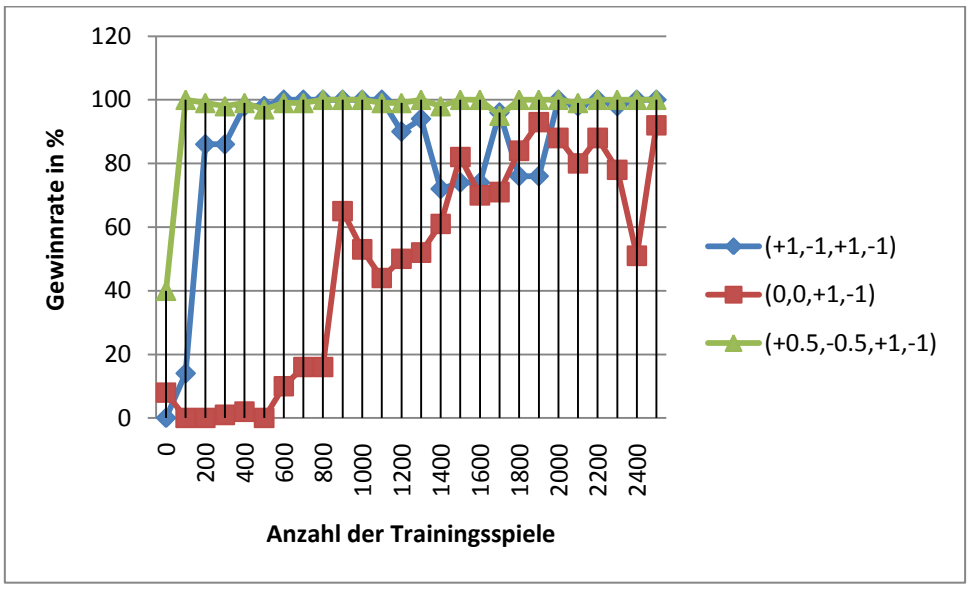


Abbildung 15: Späte Belohnungen

6.4 Vergleich mit anderen Abaloneprogrammen

Es wurde ein Agent mit 20.000 Spielen trainiert.

An dieser Stelle werden noch einmal alle verwendeten Einstellungen Zusammengefasst.

- Parameter:

n	**ε**	**α**	**λ**	**κ**
16	0,01	0,05	0,7	-0,7

- Zustandsbeschreibung:
  - *Rand, Dreier, Bedrohungen* und *Distanz zur Mitte*
- Belohnungsfunktion:
  - (+0.5,-0.5,+1,-1)

Es wurde je ein Spiel für jeden Schwierigkeitsgrad von MyLovelyAbalone sowie Aba-Pro gespielt. Spiele wurden abgebrochen sobald ein Zyklus auftrat.

MyLovelyAbalone verwendet fünf unterschiedliche Schwierigkeitsgrade. Tabelle 6 zeigt, dass nicht ein einziges Spiel gegen dieses Programm verloren wurde. Darüber hinaus wurde nur gegen den niedrigsten Schwierigkeitsgrad eine Murmel verloren. Dieses Spiel war auch das einzige, das regulär beendet wurde. MyLovelyAbalone gibt an auf Level 5 ungefähr 100.000 Positionen pro Zug zu Evaluieren. Der hier trainierte Agent evaluiert nur zwischen 60 und 80 Positionen und gewinnt dennoch.

Tabelle 6: Ergebnisse der Spiele gegen MyLovelyAbalone

Level	Gewonnene Murmeln	Verlorene Murmeln
1	6	1
2	4	0
3	5	0
4	5	0
5	4	0

In

Tabelle **7** sind die Ergebnisse gegen die ersten fünf Schwierigkeitsstufen von Aba-Pro. Wie man sieht ist Aba-Pro ab Level 4 nicht mehr geschlagen worden. Dennoch werden einige Murmeln gewonnen.

Tabelle 7: Ergebnisse der Spiele gegen Aba-Pro

Level	Gewonnene Murmeln	Verlorene Murmeln
1	5	2
2	5	2
3	4	4

4	3	6
5	3	6

7 Fazit und Ausblick

Es wurden einige in der Abaloneprogrammierung verbreitete Zustandsheuristiken evaluiert und eine aussagekräftige aber doch kleine und effizient berechenbare Zustandsrepräsentation gefunden. Des weiteren wurden verschiedene Belohnungsfunktionen evaluiert. Es konnte gezeigt werden, dass durch asymmetrische Belohnungsfunktionen der in Abalearn eingeführte Risikoparameter überflüssig wird. Belohnt man den Gewinn des Spiels höher als ein einfaches Verdrängen, so ist das Lernen leichter und stabiler. Der resultierende Agent spielt auf einem Level, dass Gelegenheitsspieler vor eine Herausforderung stellt.

Zukünftige Forschungen ließen sich in verschiedenen Richtungen anknüpfen. Zum einen wäre eine Kombination aus bestärkendem Lernen und einer Spielbaumsuche interessant. Der in KnightCap (Baxter, Tridgell und Weaver 1998) verwendete TD-Leaf Algorithmus wäre hierfür geeignet.

Auch wäre es nützlich, die Auswirkungen variabler Lernraten oder Risikoparameter zu untersuchen. Man könnte auch die Möglichkeit untersuchen verschiedene Agenten für verschiedene Phasen des Spiels zu trainieren. Beispielsweise je einen für Eröffnung, Mittelspiel und Endspiel.

Für jede Art weiterer Forschung wäre es nützlich, einen Server einzurichten auf dem die künstlichen Intelligenzen verschiedener Autoren gegeneinander spielen können.

Der Vergleich mit anderen Ergebnissen ist sehr wichtig und im Moment nur unter Mühen durchzuführen.

8 Glossar

Agent Ein Agent ist ein Computerprogramm, das zu gewissem eigenständigen Verhalten fähig ist.

Backpropagation Backpropagation ist ein verbreitetes Verfahren zum Einlernen künstlicher neuronaler Netzwerke.

Bestärkendes Lernen Bestärkendes Lernen bzw. Verstärkendes Lernen ist der Überbegriff für eine Reihe von Methoden des Maschinellen Lernens bei denen ein Agent den Nutzen von Aktionsabfolgen in einer Welt bestimmt.

Brute-Force Die Brute-Force-Methode ist ein Lösungsverfahren, das Probleme löst indem es alle Möglichkeiten durchprobiert

Heuristik Im Allgemeinen ist eine Heuristik ein zur Lösung eines Problems verwendetes Verfahren, das nicht garantieren kann, die exakte Lösung zu finden. In diesem Dokument werden die einzelnen Teilelemente einer Zustandsbeschreibung als Heuristik bezeichnet.

Manhattan Distanz Die Manhattan Distanz ist eine Metrik in der der Abstand zweier Punkte als die Summe der absoluten Differenzen ihrer Einzelkoordinaten definiert wird.

Ply Ein Ply ist ein Halbzug. Die Aktion von Schwarz ist ein Ply, die Aktion von weiß ein zweiter. Zwei Plys bilden einen Zug.

Sigmoidfunktion Eine Sigmoidfunktion ist eine mathematische Funktion, die einen S-förmigen Graphen besitzt.

9 Tabellenverzeichnis

10 Abbildungsverzeichnis

11 Literaturverzeichnis

Aichholzer, Oswin, Franz Aurenhammer, and Tino Werner. "Algorithmic Fun - Abalone." *Special Issue on Foundations of Information Processing of TELEMATIK*, 2002: 4-6.

Baxter, Jonathan, Andrew Tridgell, und Lex Weaver. „KnightCap: A chess program that learns by combining TD(lambda) with game-tree search.“ *Proceedings of the 15th International Conference on Machine Learning*, 1998: 28-36.

Campos, Pedro, und Thibault Langlois. „Abalearn: Efficient Self-Play Learning of the game Abalone.“ *European Conference on Machine Learning*, 2003: 35-46.

Ghory, Imran. „Reinforcement learning in board games.“ Department of Computer Science, University of Bristol, 2004.

Hulagu, Berk, und Ender Ozcan. „A Simple Intelligent Agent for Playing Abalone Game: ABLA.“ Proc. of the 13th Turkish Symposium on Artificial Intelligence and Neural Networks, 2004, 281-290.

Lee, Benson, und Hyun Joo Noh. „Abalone –Final Project Report.“ 2007.

Lemmens, Nyree. „Constructing an Abalone Game-Playing Agent.“ Bachelor Conference Knowledge Engineering, Universiteit Maastricht, 2005.

Malek, David. „Abalone Theory Forum.“ *Abalone Theory Forum.* 12. 4 2006. http://148009.aceboard.net/148009-512-5941-0-evaluation-function-terms.htm (Zugriff am 15. 10 2010).

Mihatsch, O, und R Neuneier. „Risk-Sensitive Reinforcement Learning.“ *Machine Learning*, 2002: 267-290.

Piccione, Peter A. „In Search of the Meaning of Senet.“ *Archaeology*, 1980: 55-58.

Sutton, Richard S., und Andrew G. Barto. *Reinforcement Learning: An Introduction.* Cambrigde, Massachusetts: The MIT Press, 1998.