Technische Universität Darmstadt

Fachbereich Informatik
Fachgebiet Knowledge Engineering
Prof. Dr. J. Fürnkranz

Diplomarbeit

Lernen von Evaluierungsfunktionen für Schachvarianten

Bearbeiter : Sacha Droste
Betreuer : Prof. Dr. J. Fürnkranz

Datum : 12. März 2008

Ehrenwörtliche Erklärung

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

Darmstadt, den 12. März 2008

Sacha Droste

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Einstieg

Schach nimmt im Bereich der künstlichen Intelligenz eine besondere Rolle ein. Das Spiel bietet ein klar strukturiertes Problem, ist aber aufgrund seiner inhärenten Komplexität auch bei heutigen Rechnerkapazitäten nicht vollständig lösbar (siehe dazu auch [LHHS+91]). In der Vergangenheit haben sich verschiedene Forscher mit dem Thema beschäftigt und David E. Wilkins schreibt: "In Anbetracht des Mangels an finanzieller Ausstattung für Schach ist erstaunlich, dass es zu so vielen Ergebnissen geführt hat." [LHHS+91] In der Disziplin des maschinellen Lernens eignet sich Schach zur Erprobung von Algorithmen des Überwachten Lernens wenn aus vorhandenen Spieldatenbanken gelernt wird oder zur Betrachtung von Konzepten aus dem Verstärkungslernen wenn ein Programm durch eigene Partien lernt. Ein wichtiger Algorithmus ist hier das Temporal Difference Learning (TD(λ)), das vor allem in einer speziellen Form, dem Q-Learning, bekannt ist.

1.2 Aufgabe

Diese Arbeit soll in Anlehnung an [BS97] und [BS99a] die Anwendung von Temporal Difference Learning zum Lernen einer Evaluierungsfunktion in Schachvarianten untersuchen. Dabei interessiert, wie sich die gelernten Werte intuitiv deuten lassen, d.h. ob Unterschiede zwischen den Werten in den Varianten und den Werten im Standardschach direkt aus den Regelunterschieden erklärt werden können.
In [BS99c] haben Beal und Smith mit Shogi zwar bereits eine Variation des Spiels untersucht, hier sollen allerdings einige Varianten betrachtet werden, deren regeltechnische Unterschiede zu Standardschach eher gering sind. Da beispielsweise die Zugmöglichkeiten der einzelnen Figuren in allen hier betrachteten Varianten denen im Standardschach gleichen, lassen sich die Varianten und Standardschach anschaulich nebeneinander betrachten. Insbesondere soll durch die Betrachtung von Piece-Square-Tables deutlich werden, wie sich die geringen Änderungen der Spielregeln auswirken können auf die Optimalkonfiguration der Evaluierungsfunktion. Inwiefern diese Optimalität tatsächlich mit TD(λ) bzw. TDLeaf(λ) erreicht wird soll ähnlich wie in [BS97] und [BS99a] anhand von Turnieren ermittelt werden, wobei sich das Problem fehlender klassischer Vergleichswerte stellt. Das Verfahren (Temporal Difference Learning) soll in eine bestehende Schachsoftware eingebaut werden. Schachvarianten, die die Software nicht beherrscht müssen dazu ebenfalls noch hinzugefügt werden.

1.3 Aufbau der Arbeit

Zunächst wird in den Kapiteln 2 und 3 Grundwissen vermittelt, dass nötig ist zum Verständnis der später durchgeführten Experimente. Dazu werden in Kapitel 2 die Spielregeln von Standardschach aufgeführt und anschließend die Schachvarianten Crazyhouse, Räuberschach und Atomschach vorgestellt. Die zweite Hälfte dieses Kapitels liefert einen Einstieg in die Arbeitsweise klassischer Schachprogramme und erklärt, wie dabei ein Spielbaum durchlaufen wird.
Der später zu verwendende Algorithmus TD(λ) wird in Kapitel 3 erläutert. Es wird die Formel vorgestellt, mit der die Veränderung der zu lernenden Werte berechnet wird sowie die darin enthaltenen Parameter. In diesem Zusammenhang werden auch Zweck und Gestalt der Stauchungsfunktion erläutert, die Werte aus dem Intervall $(-\infty, \infty)$ auf das Intervall $(0, 1)$ abbildet.
In Kapitel 4 werden Arbeiten vorgestellt, die sich mit dem Einsatz von TD(λ) zum Lernen von Evaluierungsfunktionen im Standardschach beschäftigt haben und somit eine wichtige Grundlage darstellen für die Experimente dieser Arbeit.
In Kapitel 5 wird der Aufbau der durchzuführenden Experimente und damit die Herangehensweise an die gestellte Aufgabe erklärt. Dort wird darauf eingegangen, wie die einzelnen Parameter des Algorithmus eingestellt wurden und wie die Experimente abgelaufen sind. In zwei eigenen Abschnitten werden auch die verwendeten Programme, das Schachprogramm und die Experimentierumgebung, vorgestellt.
Die Resultate aus den Experimenten werden in Kapitel 6 aufgelistet; dabei wird aus den Ergebnissen für Menschen verständliches Schachwissen abgeleitet, und in manchen Fällen in denen dies nicht gelingt wird ein alternativer Erklärungsansatz zu den gewonnenen Werten geliefert.
Den Abschluss bildet das Kapitel 7 mit einer Zusammenfassung über die gesamte Arbeit. Dort sind auch nocheinmal die interessantesten Ergebnisse aus Kapitel 6 zusammengetragen.

2 Schach

Da die Kenntnis der Schachregeln zum Verständnis einiger Passagen dieser Arbeit nötig ist, werden sie im Folgenden kurz erklärt. Die offiziellen Schachregeln der FIDE (Fédération Internationale des Échecs) finden sich unter [dE]. In einem Unterabschnitt werden auch die Regeln der Schachvarianten erklärt, die in dieser Arbeit betrachtet werden.
In der zweiten Hälfte des Kapitels wird die Funktionsweise eines klassischen Schachprogramms erklärt. Dazu gehört eine Vorstellung des grundlegenden Minimax-Algorithmus mit dem der Spielbaum durchlaufen wird, sowie einiger anderer heutzutage üblicher Techniken.

2.1 Spielregeln

Das heute verbreitete (Standard-)Schach wird immer mit zwei Spielern und auf einem quadratischen Spielfeld aus acht mal acht Feldern gespielt. Beide Spieler erhalten zu Beginn 16 Spielfiguren, wobei die Figuren des einen Spielers weiß, die des anderen schwarz sind. Die 16 Figuren setzen sich bei beiden Spielern zusammen aus acht Bauern (♙), zwei Springern (♘), zwei Läufern (♗), zwei Türmen (♖), einer Dame (♕) und einem König (♔). Die Figuren werden am Anfang wie in Darstellung 2.1 gezeigt aufgestellt.
Die Spieler müssen nun abwechselnd jeweils eine ihrer eigenen Figuren bewegen, wobei

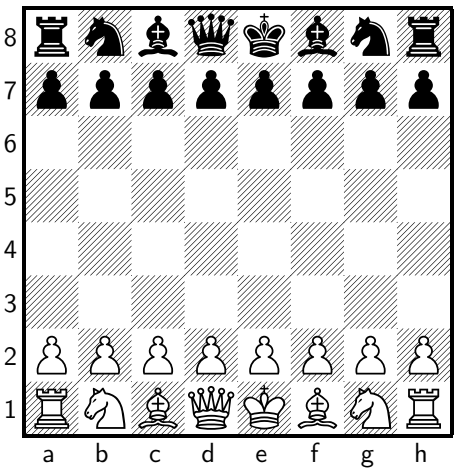


Abbildung 2.1: Anfangsaufstellung beim Schach

der Spieler mit den weißen Figuren anfängt. Die Weise, in der die Figur bewegt werden darf, ist durch ihren Typ vorgeschrieben (s.u.). Wird dabei eine Figur geschlagen, so wird

diese Figur vom Spielfeld entfernt.
Wird eine Figur so bewegt, dass sie den gegnerischen König bedroht[1], ist der gegnerische Spieler mit dem Wort "Schach!" darauf aufmerksam zu machen. Befindet sich der König nach diesem Zug allerdings in einer unabwendbaren Bedrohung so hat der Spieler (der den gegnerischen König *matt* gesetzt hat) das Spiel gewonnen und verkündet dies mit dem Ausruf "Schach Matt!".

Die Bewegungsmöglichkeiten der einzelnen Figuren sind wie folgt:

Bauer ♙ Der Bauer bewegt sich normalerweise ein einzelnes Feld geradeaus (in Richtung des gegnerischen Spielers). Wurde der Bauer in diesem Spiel noch nicht bewegt und befindet er sich noch auf seiner Anfangsposition so darf er alternativ auch einen Doppelschritt machen, d.h. er bewegt sich zwei Felder geradeaus; dabei darf er allerdings keine Figur überspringen. In beiden Fällen aber muss das Zielfeld leer sein.

Ein Bauer kann eine gegnerische Figur schlagen, indem er das Feld betritt auf dem sich diese Figur befindet. Die Figur muss dazu auf einem diagonal an die Front des Bauern angrenzenden Feld stehen. Ein Bauer auf auf dem Feld d4 bedroht beispielsweise die Felder c5 und e5.

Die letzte Zugmöglichkeit für den Bauern stellt das Schlagen En Passant (also "im vorbeigehen") dar: Hat der gegnerische Spieler mit seinem letzten Zug einen seiner Bauern mit einem Doppelschritt bewegt und einer der eigenen Bauern bedroht das übersprungene Feld, so kann der gegnerische Bauer geschlagen werden als hätte er nur einen Einzelschritt gemacht.

Betritt ein Bauer durch seinen Zug die letzte Reihe des Spielfelds[2], so tauscht der entsprechende Spieler die Bauernfigur nach Wahl aus durch einen Springer, einen Läufer, einen Turm oder eine Dame.

Springer ♘ Der Springer bewegt sich immer zwei Felder in eine Richtung (nicht diagonal) und dann ein drittes Feld senkrecht dazu. Dabei darf er auch beliebige Figuren überspringen und eine evtl. auf dem Zielfeld befindliche gegnerische Figur wird geschlagen.

Läufer ♗ Der Läufer bewegt sich immer über eine beliebig lange Strecke diagonal. Er darf aber weder zwischendurch die Richtung ändern noch jegliche Figuren überspringen. Ist eine gegnerische Figur auf dem Zielfeld, so wird diese geschlagen.

Turm ♖ Der Turm bewegt sich wie der Läufer mit der Änderung, dass er sich horizontal oder vertikal bewegt statt diagonal.

Dame ♕ Die Dame darf sich sowohl diagonal bewegen wie ein Läufer als auch horizontal oder vertikal wie ein Turm.

König ♔ Der König bewegt sich wie die Dame, hat dabei aber nur eine Reichweite von einem Feld. Zusätzlich steht dem König ein Spezialzug zur Verfügung, die Rochade.

[1] Eine Figur *bedroht* eine andere, wenn sie sie mit ihrem nächsten Zug schlagen könnte.
[2] Für den weißen Spieler ist dies die Reihe acht, für den schwarzen Spieler ist dies die Reihe eins.

Bei der Rochade werden sowohl der König als auch ein Turm bewegt. Dazu müssen aber zunächst einige Bedingungen erfüllt sein:

1. Der König wurde in diesem Spiel noch nicht bewegt.
2. Der gewählte Turm wurde in diesem Spiel noch nicht bewegt.
3. Die Felder zwischen dem König und dem gewählten Turm sind leer.
4. Der König wird nicht bedroht.
5. Das Zielfeld des Königs wird nicht bedroht.
6. Das Feld, das der König bei der Rochade überspringt wird nicht bedroht.

Der Turm wird nun seitlich an den König herangezogen und der König daraufhin über den Turm direkt an dessen andere Seite bewegt.

2.1.1 Schachvarianten

Schachvarianten sind Spiele, die zwar grundlegend den Schachregeln folgen, dabei aber einige (üblicherweise recht kleine) Änderungen enthalten. Die folgenden drei stellen lediglich eine Auswahl aus einer Unzahl existenter Schachvarianten dar.

Räuberschach

Von Räuberschach gibt es verschiedene Varianten. Hier soll diejenige betrachtet werden, die im Englischen unter dem Namen Suicide bekannt ist[3].
Ziel des Spiels bei Räuberschach ist es nicht, den gegnerischen König Matt zu setzen, sondern sämtliche eigene Figuren zu verlieren. Der König wird zwar so bewegt wie im Standardschach auch, er kann aber normal geschlagen werden und sein Verlust hat keine besondere Bedeutung. Dies bedeutet auch, dass Bauern auch zu Königen befördert werden können und die Rochade komplett entfällt.
Hat ein Spieler die Möglichkeit, eine gegnerische Figur zu schlagen, so *muss* er dies auch tun (Schlagzwang). Hat er mehrere Möglichkeiten gegnerische Figuren zu schlagen, so darf er zwischen diesen frei wählen.
Es gibt Pattsituationen in denen keiner der beiden Spieler mehr einen gültigen Zug übrig hat. Nach den Regeln, die auf dem FICS (Free Internet Chess Server) gelten, wird dies als Gewinnsituation betrachtet für den Spieler, der zu dem Zeitpunkt weniger Figuren übrig hat. Haben beide Spieler gleichviele Figuren übrig wird die Situation als Unentschieden behandelt.

Crazyhouse

Wird in Crazyhouse eine Figur geschlagen, so wechselt diese ihre Farbe und wandert auf die Hand des schlagenden Spielers. Hat ein Spieler eine oder mehrere Figuren auf der Hand, so kann er alternativ zu einem normalen Schachzug auch eine der Figuren aus seiner Hand an fast beliebiger Stelle auf dem Spielfeld platzieren; es dürfen lediglich keine

[3] Wie auf der Internetseite von Sjeng (`http://www.sjeng.org/indexold.html`) zu lesen ist gibt es außerdem die Varianten Giveaway und Losers

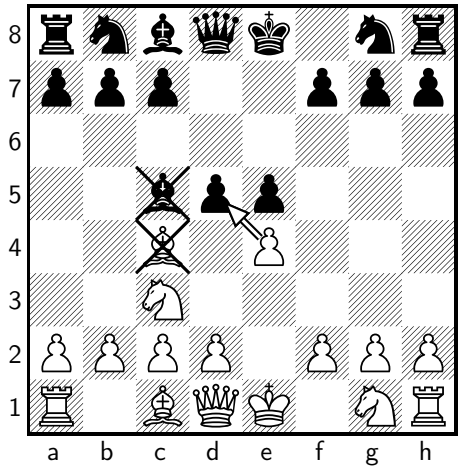


Abbildung 2.2: Schlagen einer Figur in Atomschach

Bauern in die erste oder letzte Reihe gesetzt werden.
Es kann vorkommen, dass eine Figur geschlagen wird, die irgendwann vorher ein Bauer war und erst durch Beförderung (siehe Bauernregeln auf Seite 4)zu einem Springer, Läufer, Turm oder einer Dame wurde. In diesem Fall wird nicht die *geschlagene Figur* auf die Hand genommen, sondern ein *Bauer*
Wird der König Matt gesetzt, so ist das Spiel wie herkömmlich entschieden. Der König wird nicht geschlagen und dementsprechend nicht auf die Hand genommen.

Atomschach

In Atomschach führt das Schlagen einer Figur zu einer Explosion, deren Zentrum das Feld der geschlagenen Figur ist. Die Explosion erstreckt sich auf alle horizontal, vertikal und diagonal direkt angrenzenden Felder. Die geschlagene Figur wird wie üblich entfernt, zusätzlich aber werden noch die schlagende Figur und alle Nicht-Bauern in Explosionsreichweite vom Spielfeld genommen. Wird der König auf diese Weise (indirekt) geschlagen, so hat der schlagende Spieler gewonnen als er hätte er den König Matt gesetzt. Dies wird auch als indirektes Matt bezeichnet.

2.2 Computerschach

Wenn es darum geht, in einer gegebenen Schachsituation den nächsten Zug festzulegen, dann gehen klassische Schachprogramme und gute menschliche Spieler sehr unterschiedlich vor[Cla73]. Ein guter menschlicher Spieler erkennt in der Stellung der Figuren auf dem Schachbrett Muster, die durch lange Erfahrung in vielen Spielen gelernt wurden. Er versucht dann die Muster auf dem Feld in andere ihm bekannte Muster zu überführen. Dabei geht er sehr selektiv vor und zieht einen Großteil der regeltechnisch gültigen Züge überhaupt nicht Erwägung.

Klassische Schachprogramme[4] arbeiten dagegen mit einem Spielbaum. Die *Grundidee* bei diesem Spielbaum besteht darin, auf einer internen Kopie des Schachbretts solange alle möglichen Züge und Folgezüge durchzuprobieren, bis schließlich keine Züge mehr möglich sind, weil das Spiel beendet ist. Die Wurzel des Spielbaums ist also die aktuelle Situation auf dem Brett und die Blätter sind alle möglichen Ausgänge des Spiels. Ein einfaches Spiel, das mit dieser Methode gut lösbar ist, ist Tic Tac Toe. Beim Schachspiel sind die möglichen Züge und Spielverläufe allerdings so zahlreich, dass das Spiel bei dem heutigen Stand der Computertechnik noch lange nicht vollständig lösbar ist, d.h. der entstehende Spielbaum wäre so groß und sein Aufbau so zeitaufwändig, dass dieses Vorgehen in der Praxis nicht möglich ist. Um dennoch zu einer Lösung (also einer Entscheidung für einen Schachzug) zu kommen beschränkt man sich darauf den Spielbaum nur bis zu einer gewissen Tiefe aufzubauen. Wenn man nun beispielsweise eine Suchtiefe von fünf Halbzügen festlegt, dann wird der Aufbau des Spielbaums gestoppt, sobald eine Tiefe von fünf erreicht ist. Die Blätter des Baums sind nun in den meisten Fällen normale (nicht spielentscheidende) Schachsituationen. Da bei diesen aber nicht bekannt ist, ob sie zum Verlust oder Gewinn des Spiels führen, muss dies (qualifiziert) abgeschätzt werden. Dazu wird eine Evaluierungsfunktion verwendet, die als Eingabe die Situation auf dem Brett erhält und als Ausgabe einen Skalar liefert.

$$eval(\vec{f}) = e$$

Die Situation auf dem Spielbrett wird als Vektor von Merkmalen (Features) kodiert. Diese Merkmale können z.B. sein die Anzahl Bauern der einen Seite abzüglich der Anzahl der Bauern der anderen Seite. Je positiver die Evaluierung ist, desto größer ist die Wahrscheinlichkeit, dass der Spieler, der in der entsprechenden Situation einen Zug auszuführen hat, das Spiel gewinnt. Eine Null zeigt eine Gewinnwahrscheinlichkeit von 50% an.
Die Evaluierungswerte aus den Blättern werden dann mit dem Minimax-Algorithmus durch den Baum nach oben gereicht bis schließlich in der Wurzel die Evaluierung nur eines Blattes, der Principal Position, ankommt. Es wird dann der Zug gewählt, der zu diesem Blatt führt.

2.2.1 Minimax-Algorithmus

Dem Minimax-Algorithmus ([Sch86]) liegt die Annahme zugrunde, dass der aktuelle Spieler innerhalb des Spielbaums immer den für ihn günstigsten Zug wählt während der Gegenspieler immer den widrigsten Zug wählt. Bei einem Zug, der vom aktuellen Spieler durchgeführt wird, wird daher die Zugmöglichkeit propagiert, die die höchste Evaluierung erhalten hat. Bei einem gegnerischen Zug wird die Zugmöglichkeit mit der niedrigsten Evaluierung propagiert. Dies soll an einem einfachen Beispiel verdeutlicht werden (siehe Abbildung 2.3). Von den Werten in den Blättern werden jeweils die kleinsten weitergegeben (4 und 7), da dieser Zug vom Gegenspieler durchgeführt wird. Zur Wurzel wird dann der größte Wert weitergereicht (7), da dieser Zug vom aktuellen Spieler durchgeführt

[4] Einen alternativen Weg schlägt das Projekt Morph ein.[LHHS+91]

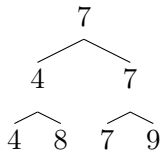


Abbildung 2.3: Minimalbeispiel für einen Spielbaum

wird.
Von diesem grundlegenden Verfahren wird üblicherweise eine Variante eingesetzt, die als Alpha-Beta-Algorithmus bezeichnet wird. Es handelt sich dabei um eine verlustfreie[5] Verkleinerung des Spielbaumes. Ohne auf Implementierungsdetails einzugehen soll das Prinzip hier mit Hilfe eines Beispiels verdeutlicht werden (Abbildung 2.4):

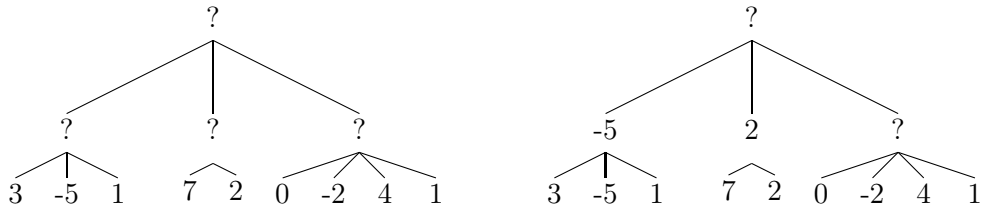


Abbildung 2.4: Propagieren von Knotenwerten in Alpha-Beta

Im ersten Zweig wird wie gewohnt der kleinste Wert (-5) an den darüberliegenden Knoten weitergegeben. Dieser Wert wird als Alpha gespeichert und an die Durchsuchung des zweiten Zweiges als Beta weitergereicht. Die Werte im zweiten Zweig sind alle nicht besser (d.h. kleiner) als das Beta (-5) und deshalb wird wieder ganz normal das Minimum aus den Blättern gebildet (2). Auf der Ebene darüber wird nun festgestellt, dass die zwei besser (also größer) ist als das bisherige Alpha; daher wird das alte Alpha (-5) durch die zwei ersetzt und diese als Beta an die Durchsuchung des dritten Zweiges weitergegeben. Im Dritten Zweig ist gleich der erste Wert besser (d.h. kleiner) als das Beta (da $0 < 2$) und die Suche dieses Zweiges kann damit abgebrochen werden. Da beim Minimieren das Ergebnis auf garkeinen Fall noch größer (d.h. schlechter) werden konnte als 0, ist dieser gesamte Zweig uninteressant. Auf der Ebene darüber wird ohnehin das Maximum gebildet aus dem Alpha (hier die 2) und einer Zahl ≤ 0, was auf jeden Fall 2 ergibt. Der Ergebnisbaum ist in Abbildung 2.5 zu sehen.

2.2.2 Weitere Verfahren

Es gibt noch weitere Mechanismen, mit denen die Suche optimiert werden kann. Unter Quiescence versteht man eine Technik bei der die angestrebte Suchtiefe an einzelnen Stellen das Suchbaums erhöht wird. Anlass dazu könnte die Möglichkeit sein eine gegnerische

[5] Das bedeutet, dass der Baum zwar weniger Knoten enthält und daher schneller durchsucht werden kann, das Ergebnis der Suche allerdings nicht verändert wird.

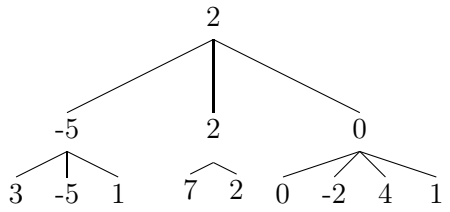
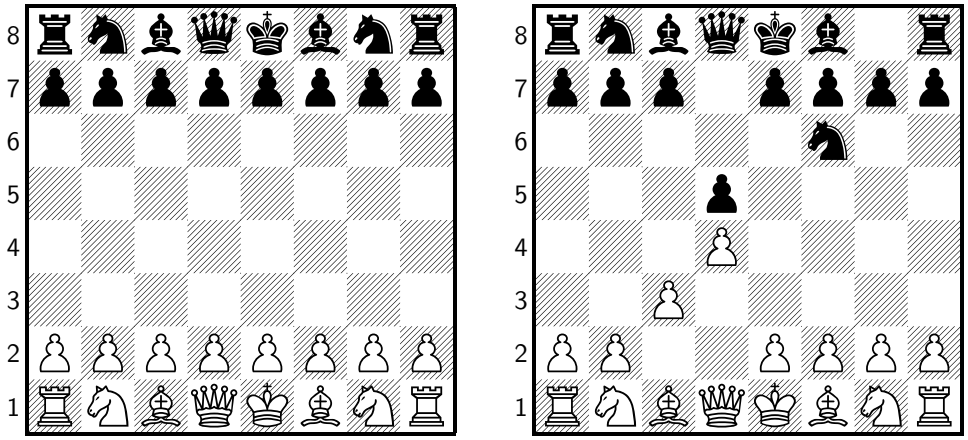


Abbildung 2.5: Propagierte Knotenwerte in Alpha-Beta

Figur zu schlagen oder die Tatsache, dass einer der beiden Spieler im Schach (nicht Matt) steht.
Transpositionstabellen ([BU]) helfen dabei den Suchaufwand bei Transpositionen gering zu halten. Von einer Transposition spricht man, wenn im Spielbaum eine Position auftaucht, die bereits an anderer Stelle im gleichen Spielbaum gesehen und durchsucht wurde. Dies ist der Fall, wenn von einer Ausgangsstellung aus eine andere Stellung auf dem Spielfeld durch verschiedene Zugfolgen, d.h. über verschiedene Wege durch den Suchbaum, erreicht werden kann. Als Beispiel seien die Ausgangs- und Zielstellungen in Abbildung 2.6 gegeben. Von der Ausgangsstellung ist die Zielstellung über verschiedene



2.6.1: Ausgangsstellung

2.6.2: Zielstellung/Transposition

Abbildung 2.6: Transpositionsbeispiel

Zugfolgen zu erreichen:

1. ♙d2→d4; ♙d7→d5; ♙c2→c3; ♘g8→f6
2. ♙c2→c3; ♙d7→d5; ♙d2→d4; ♘g8→f6
3. ♙d2→d4; ♙g8→f6; ♙c2→c3; ♘d7→d5
4. ♙c2→c3; ♙g8→f6; ♙d2→d4; ♘d7→d5

Während der Spielbaum durchsucht wird werden gesehene Situationen in der Transpositionstabelle eingetragen. Üblicherweise ist das eine Hashtabelle, die zu einer gespeicherten Position Daten enthält, wie einen Hash der Position, das Ergebnis der Durchsuchung dieses Knotens und die dabei erreichte Suchtiefe.
Weitere Beispiele (die hier aber nicht weiter erklärt werden sollen) sind Nullzüge ([TN02]) und Multi-cut Pruning ([BM99]).

3 Maschinelles Lernen

Nachdem im vergangenen Kapitel Grundwissen um das Thema Schach herum vermittelt wurde, stellt dieses den Algorithmus vor, der in den Experimenten verwendet wurde. Im ersten Abschnitt wird das Grundprinzip des Verstärkungslernens vorgestellt und im zweiten eine spezielle (einfache) Ausprägung von Temporal Difference Learning. Der dritte Abschnitt geht auf Temporal Difference Learning selbst ein und behandelt in diesem Zusammenhang auch die Stauchungsfunktion, die in den Experimenten dieser Arbeit verwendet wurde.

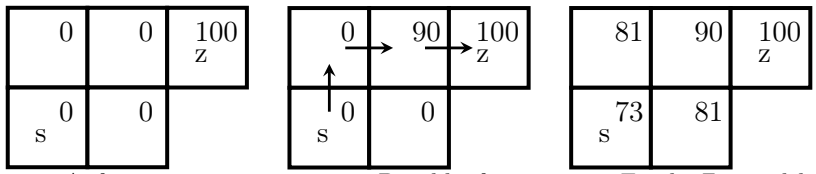
3.1 Verstärkungslernen

Temporal Difference Learning ist eine Methode des Verstärkungslernens (Reinforcement Learning), einem Typ des maschinellen Lernens. Beim Verstärkungslernen soll ein Programm oder Agent seine Handlungen so optimieren, dass die darauf folgende Belohnung maximiert wird[Lug01]. Bei einem Schachcomputer sind die Handlungen die Entscheidungen für Züge, und die Belohnung (oder Bestrafung) ist der Gewinn (oder Verlust) des Spiels. Das Programm sammelt seine Erfahrungen durch Ausprobieren von Zügen in verschiedenen Durchläufen. Dabei ist wichtig, dass einerseits die Fülle an Handlungsmöglichkeiten weiträumig erforscht wird (Suchraumexploration), andererseits aber Handlungen die bereits positiv bewertet wurden etwas bevorzugt werden.
Um tatsächlich optimierte Werte zu lernen sind mehrere (unterschiedliche) Programmdurchläufe notwendig, in denen der Agent seine Umwelt (z.B. ein Schachbrett einschließlich der dazugehörigen Figuren und Regeln) nach und nach erkundet und die Handlungsmöglichkeiten darin anhand der Konsequenzen (Belohnungen) zu bewerten lernt.

3.2 Q-Learning

Eine Methode, mit der das oben beschriebene realisiert werden kann ist das Q-Learning. Wie auch sonst beim Verstärkungslernen, werden hier eigentlich nicht die Handlungen bewertet, sondern die vorhandenen Situationen bzw. Zustände der Umwelt und des Agenten. Der Wert einer Handlung entspricht dann dem Wert des Zustandes zu dem sie führt. Beim Q-Learning werden zunächst alle Zustände als neutral betrachtet. Wenn eine Kette von Zuständen zu einer Belohnung geführt hat, wird der letzte in dieser Kette mit der Belohnung attribuiert. In späteren Durchläufen wird dann die Bewertung von Zuständen immer ein wenig angepasst an die Bewertung des jeweiligen Nachfolgerzustands.
Angenommen, der Agent muss sich vom Startfeld S zum Zielfeld Z bewegen (siehe Abbildung 3.1). Die Belohnung bei Erreichen des Feldes Z seien 100 Punkte. Dann sieht die

Bewertung der Felder nach dem ersten Durchlauf aus wie in Abbildung 3.1.1. Nimmt er beim zweiten Durchlauf den mit den Pfeilen markierten Weg so ändert sich die Bewertung wie in Abbildung 3.1.2 dargestellt bis nach mehreren Durchläufen schließlich die Bewertungen in Abbildung 3.1.3 zustande kommen.
Eine detailliertere Erklärung zu Q-Learning findet sich in [Mit97].



3.1.1: Anfangssituation 3.1.2: Durchlauf 2 3.1.3: Finale Zustandsbewertungen

Abbildung 3.1: Zustandsbewertungen beim Q-Learning

3.3 Temporal Differnce Learning

Temporal Difference Learning ist eine Verallgemeinerung von Q-Learning. Statt die Bewertung eines Zustandes nur gemäß ihres direkten Nachfolgers zu verändern werden gleich mehrere aus der Kette der Nachfolger betrachtet. Hier liegt aber die Vermutung nahe, dass der direkte Nachfolger stärkeren Einfluss haben sollte als der Nachfolger des Nachfolgers. Um diese zeitliche Nähe der Nachfolgezustände zum aktuell anzupassenden Zustand zu berücksichtigen wird der Parameter λ verwendet. Abgekürzt wird Temporal Difference Learning daher auch üblicherweise als TD(λ) bezeichnet. Das λ liegt zwischen 0 und 1, und TD(0) entspricht Q-Learning, da bei einem λ von 0 nur aus dem direkten Nachfolgezustand gelernt wird.
Wie in Abschnitt 2.2 erwähnt, werden die einzelnen Zustände in Merkmalsvektoren kodiert. Die Bewertung eines Zustandes erfolgt dann über eine Funktion, die abhängig ist von den Ausprägungen der einzelnen Merkmale und der Bewertung dieser Merkmale. Damit die übliche Aktualisierungsformel (siehe Gleichung 3.2 unten) für die Gewichte bei TD(λ) funktioniert, muss die Ausgabe der Bewertungsfunktion zwischen 0 und 1 liegen, die Evaluierungsfunktion liefert aber üblicherweise ganze Zahlen in einem Bereich der zumindest während des Lernvorganges kaum einzuschränken ist. Daher wird eine Stauchungsfunktion verwendet, die als Eingabe die Ausgabe der Evaluierungsfunktion übernimmt und als Ausgabe einen Wert zwischen 0 und 1 liefert. Eine übliche Stauchungsfunktion, die auch im Rahmen dieser Arbeit verwendet wurde, ist:

$$P(\nu) = \frac{1}{1 + e^{-\omega\nu}} \tag{3.1}$$

Mögliche Verläufe dieser Funktion sind in Abhängigkeit von ω in Abbildung 3.2 dargestellt. Sie gibt im Kontext von Schach die Gewinnwahrscheinlichkeit zu einer gegebenen Evaluierung an. Beispielsweise führt ein ausgeglichener Zustand zu einer Evaluierung von 0 und P(0) = 50%. Die Ausgaben dieser Funktion über den Verlauf eines Spieles können

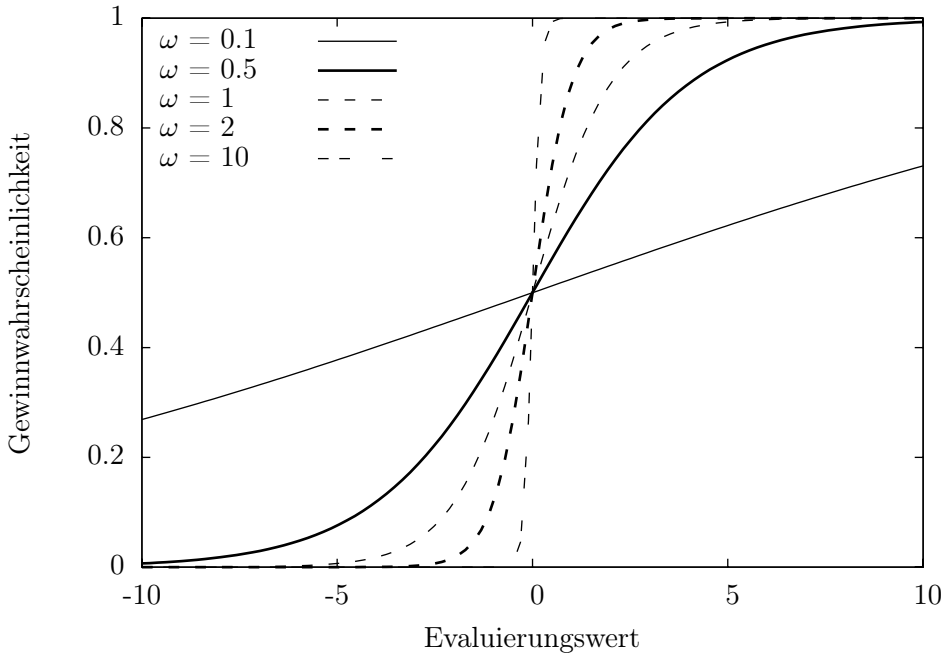


Abbildung 3.2: Verlauf der Stauchungsfunktion

auch als Folge $P_1, \ldots, P_n$ aufgefasst werden.
Die Aktualisierungsformel, mit der die einzelnen Werte für die Zustandsmerkmale nach jedem Zeitschritt angepasst werden (z.B. nach jedem Spiel), lautet für den Zeitschritt t:

$$\Delta W_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} (\lambda^{t-k} \nabla_W P_k) \tag{3.2}$$

α ist dabei die Lernrate und liegt zwischen 0 und 1. Es könnte beispielsweise bei konstant 0,5 liegen. In [BS97] liegt α zu Beginn des Lernvorgangs bei 0,1 und wird dann im Laufe des Verfahrens schrittweise auf 0 reduziert. $\nabla_W P_k$ ist der Gradient von $P(\nu_k)$ und ν_k ist der Evaluierungswert des Zustandes k. Die Ableitung von $P(\nu)$ ergibt sich aus:

$$\begin{aligned} \frac{dP}{d\nu} &= -(1 + e^{-\omega\nu})^{-2} * -e^{-\omega\nu} \\ &= \frac{e^{-\omega\nu}}{(1 + e^{-\omega\nu})^2} \\ &= \frac{1 + e^{-\omega\nu}}{(1 + e^{-\omega\nu})^2} - \frac{1}{(1 + e^{-\omega\nu})^2} \\ &= P(1 - P) \end{aligned} \tag{3.3}$$

Die Evaluierungsfunktion ν wird üblicherweise einfach gehalten (aus Performanzgründen) und hat daher eine lineare Form:

$$\nu = \sum_{Merkmale} w_{Merkmal} c_{Merkmal} \tag{3.4}$$

$w_{Merkmal}$ ist hier das gelernte Gewicht für ein Merkmal und $c_{Merkmal}$ ist die Ausprägung des Merkmals im aktuell zu bewertenden Zustand.
Hat der weiße Spieler beispielsweise einen Vorsprung von 2 Bauern und einen Rückstand von einem Springer könnte die Evaluierung ν etwa wie folgt aussehen:

$$\begin{aligned}
\nu = &\quad w_{Bauer} * 2 + w_{Springer} * (-1) + w_{Laeufer} * 0 + w_{Turm} * 0 + w_{Dame} * 0 \\
\nu = &\quad 1 * 2 + \quad 3 * (-1) + \quad 3 * 0 + \quad 5 * 0 + \quad 9 * 0 \\
\nu = &-1
\end{aligned}$$

Die Ableitung von ν lautet:

$$\frac{d\nu}{dw_{Merkmal}} = c_{Merkmal} \tag{3.5}$$

Mit der Ableitung von ν in Gleichung 3.5 und der Ableitung in Gleichung 3.3 ergibt sich:

$$\nabla_W P_k = c_{Merkmal} P_k (1 - P_k) \tag{3.6}$$

Die Aktualisierung der Gewichte w_i kann zu verschiedenen Zeitpunkten erfolgen. Bei Schach erfolgt sie üblicherweise nach jedem Spiel. Es wäre hier aber auch möglich, alle 10, 20 oder 30 Züge zu aktualisieren. Dies kann vor allem dann von Vorteil sein wenn die einzelnen Zustandsfolgen sehr lang sind und eine häufigere Aktualisierung das Lernverfahren schneller zu einem Ergebnis kommen lassen kann.

4 Arbeiten zu verwandten Themen

In diesem Kapitel werden Arbeiten vorgestellt, die eine besondere Relevanz haben für das Thema dieser Arbeit. Den Anfang macht eine Arbeit von Baxter, Tridgell und Weaver, in der es darum geht, wie man TD(λ) geschickt einsetzt bei Verwendung des Minimaxverfahrens zum Durchsuchen eines Spielbaums. TD(λ) dient dabei dazu, die verwendete Evaluierungsfunktion zu lernen.
Die Arbeit im zweiten Abschnitt schildert ein Verfahren, mit dem die Lernrate α (siehe Gleichung 3.2) im Laufe des Verfahrens automatisch angepasst wird. Damit wird die Lernrate um so kleiner, je stärker sich das zu lernende Gewicht seinem Zielwert nähert.
Im dritten Abschnitt werden drei Arbeiten von Beal und Smith vorgestellt, in denen Gewichte für eine Evaluierungsfunktion mittels TD(λ) gelernt wurden. In einer Arbeit wurden Materialwerte für die Figuren beim Schach gelernt, in einer anderen wurden die Werte für Figuren in Shogi, einem japanischen Spiel, das dem Schach sehr ähnelt, gelernt. In der dritten Arbeit wurden schließlich Evaluierungsfunktionen für Schach gelernt, die auch die Positionen der Figuren auf dem Spielbrett berücksichtigen.

4.1 TDLeaf(λ)

Baxter, Tridgell und Weaver schlagen in [JB98] eine Anpassung von TD(λ) vor, die speziell für die Verwendung in Kombination mit dem Minimaxverfahren gedacht ist.
Zunächst definieren sie ein Verfahren, das sie TD-directed(λ) nennen. Dabei wird die Zustandsfolge nicht zufällig sondern mit Hilfe von Minimax bestimmt, d.h. das Programm spielt während des Lernens bereits nach dem gleichen Verfahren wie später in normalen Spielen. Bei der Aktualisierung der Gewichte allerdings werden die dafür nötigen Daten (wie $c_{Merkmal}$ und $P(\nu_t)$) aus den Zuständen genommen die im Verlauf des Spiels aufgetreten sind.
Bei TDLeaf(λ) wird nicht mehr aus den Zuständen selbst gelernt, sondern aus den Principal Positions der Zustände. Die Principal Position ist in Minimax das Blatt des Spielbaums, dessen Wert bis zur Wurzel propagiert wurde. Es wird also beispielsweise zur Berechnung von ν_t nicht mehr ν (die Evaluierungsfunktion) auf den Zustand t angewendet, sondern auf den Zustand, der erwartungsgemäß von Zustand t aus erreicht wird, also die Principal Position
Es ist zu beachten, dass bereits gezeigte Konvergenzeigenschaften (siehe dazu [Sut88]) von TD(λ) nicht ohne weiteres übertragbar sind auf TDLeaf(λ) ([JB98]); das Verfahren hat sich allerdings bewährt in [JB98] und in dieser Arbeit.

4.2 Temporal Coherence

Temporal Coherence ist eine in [BS99b] beschriebene Technik zur Steuerung der Höhe der Lernrate α in TD(λ). Bei jeder Aktualisierung der Gewichte wird folgende Variation von Gleichung 3.2 auf Seite 13 verwendet:

$$\Delta w_i = c\alpha_i \sum_{t=1}^{ende-1} r_{i,t} \tag{4.1}$$

$r_{i,t}$ stellt die empfohlene Veränderung von w_i für den Zeitschritt t dar. Berechnet werden diese Veränderungen mit:

$$r_{i,t} = (P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_{w_i} P_k$$

Die α_i in Gleichung 4.1 sind lokale Lernraten für die einzelnen Gewichte. Die Gesamtlernrate für das Gewicht w_i ergibt sich damit aus dem Produkt der lokalen Rate α_i und der globalen konstanten Rate c. Zu Beginn des Verfahrens werden die α_i mit 1 initialisiert und später dann berechnet mit:

$$\alpha_i = \frac{|N_i|}{A_i} \tag{4.2}$$

Nach jeder Aktualisierung werden die Werte N_i und A_i entsprechend den Regeln 4.3 und 4.4 angepasst.

$$N_i \leftarrow N_i + \sum_{t=1}^{ende-1} r_{i,t} \tag{4.3}$$

$$A_i \leftarrow A_i + \sum_{t=1}^{ende-1} |r_{i,t}| \tag{4.4}$$

A_i ist die richtungsunabhängige Summe aller empfohlenen Änderungen $r_{i,t}$; N_i ist die Nettoänderung. Zu Beginn des Verfahrens, wenn w_i noch stark von seinem Zielwert abweicht und die empfohlenen Änderungen (fast) alle das gleiche Vorzeichen haben, unterscheiden sich N_i und A_i kaum. Gemäß Gleichung 4.2 führt das zu einer hohen Lernrate. Später im Verfahren werden die $r_{i,t}$ zunehmend durch Rauschen bestimmt; N_i bleibt etwa konstant während A_i weiter steigt. Dadurch nähert sich α_i mit zunehmender Konvergenz dem Wert 0.

4.3 Anwendungen von TD(λ) im Schach

4.3.1 Lernen von Materialwerten

In [BS97] haben Beal und Smith in einem Versuch TD(λ) verwendet, um Materialwerte für Schachfiguren bzw. eine rein materialbasierte Evaluierungsfunktion zu lernen. Dazu

wurde ein Verfahren verwendet, das dem oben beschriebenen TD-directed(λ) (siehe Abschnitt 4.1) entspricht. Die Evaluierungsfunktion entspricht Gleichung 3.4 auf Seite 14. Es gibt für jede der fünf Figuren[1] ein Merkmal, und dessen Ausprägung ist die Anzahl dieser Figur, die der aktuelle Spieler auf dem Brett hat, abzüglich der Anzahl dieser Figur, die der gegnerische Spieler auf dem Brett hat. Wenn z.B. Weiß der aktuelle Spieler ist, gilt:

$$c_Bauer = \text{(weiße Bauern auf dem Brett)} - \text{(schwarze Bauern auf dem Brett)}$$

Die Stauchungsfunktion entspricht Gleichung 3.1.
Die Werte, die Beal und Smith mit diesem Versuch ermittelt haben, wurden anschließend so normiert, daß der Wert eines Bauern bei eins lag. Die Ergebnisse entsprechen in etwa den traditionellen Werten, die im Schach bekannt sind. Abschließend hat man die Werte

Figur	Gelernter Wert	Traditioneller Wert
Bauer	1	1
Springer	2,628	3
Läufer	3,218	3
Turm	4,546	5
Dame	8,862	9

getestet indem man ein Turnier veranstaltet hat, bei dem beide Seiten vom gleichen Programm gespielt wurden, die eine Seite aber mit traditionellen Werten gespielt hat und die andere mit gelernten. Dabei hat das Programm mit den gelernten Werten von 2000 Spielen etwa 57% für sich entschieden. Unentschiedene Spiele wurden als halber Punkt für beide Programme gezählt. Das Ergebnis zeigt zum einen, dass TD(λ) für das obige Szenario funktioniert und zum anderen, dass die gelernten Werte tatsächlich (etwas) besser sind als die traditionellen.

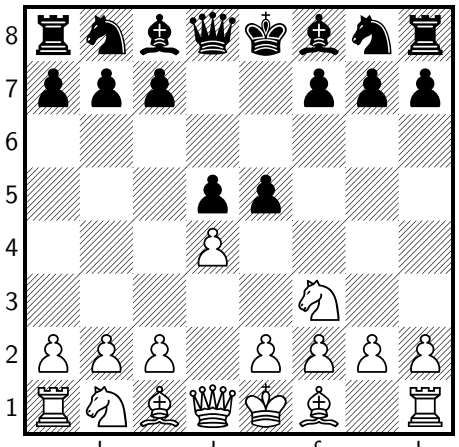
Lernen von Materialwerten in Shogi

In [BS99c] beschreiben Beal und Smith wie sie für das Spiel Shogi Werte für die Figuren haben lernen lassen. Shogi ist Schach recht ähnlich; man könnte es auffassen als eine Schachvariante mit einem etwas größeren Spielfeld und anderen Figurentypen.
Auch hier wurde TD(λ) verwendet, und die Stauchungsfunktion und ν waren ebenfalls die gleichen wie in [BS97]. Zum Abschluss wurde wieder die Tauglichkeit der gelernten Werte betrachtet durch paarweise Turniere zwischen vier Programmen. Der Unterschied zwischen den Programmen lag wieder ausschließlich in den Figurwerten die in der Evaluierungsfunktion verwendet wurden. Da es in Shogi allerdings keine allgemein üblichen Werte für die Figuren gibt, wurden für die Testgegner Werte aus anderen Shogiprogrammen verwendet. Ein Turnier umfasste jeweils 2000 Spiele zwischen zwei Programmen, und das Programm mit den gelernten Werten gewann je nach Gegner 52%–62% der Spiele.

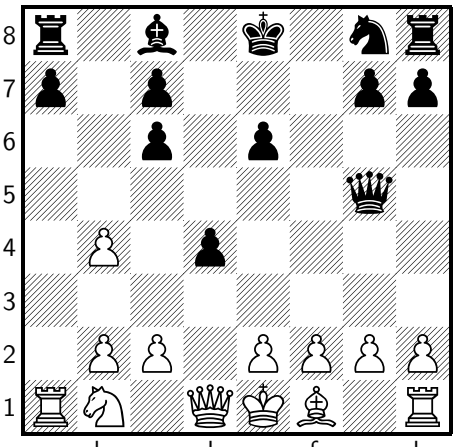
[1] Im Standardschach macht es keinen Sinn einen Wert für den König zu bestimmen.

4.3.2 Lernen von Piece-Square-Tables

In [BS99a] führen Beal und Smith den Versuch aus [BS97] in erweiterter Form durch. Die Evaluierungsfunktion ist jetzt nicht mehr rein materialbasiert sondern berücksichtigt auch die Positionen der Figuren auf dem Schachbrett. Dadurch können Programm und Algorithmus Stellungen unterscheiden, die vom Material her gleich sind. Die Situationen



4.1.1: Beispielsituation 1



4.1.2: Beispielsituation 2

Abbildung 4.1: Unterschiedliche Situationen mit gleicher Evaluierung (materialbasiert)

in Abbildung 4.1.1 und 4.1.2 erhalten bei rein materialbasierter Evaluierung exakt den gleichen Wert; bei Verwendung positionsabhängiger Evaluierungsmerkmale können die Situationen aber unterschieden und entsprechende Gewichte gelernt werden. Dazu werden drei Möglichkeiten betrachtet die Evaluierungsfunktion zu erweitern und somit die Anzahl der zu lernenden Werte zu erhöhen:

1. Figurenzentralität und Bauernränge
2. Halbe Piece-Square-Tables
3. Piece-Square-Tables

Algorithmus und Stauchungsfunktion sind wieder die gleichen wie in [BS97].

Figurenzentralität und Bauernränge

Bei der Verwendung der Bauernränge wird für jeden Rang, auf dem ein Bauer stehen kann, ein neues Merkmal zur Evaluierungsfunktion hinzugefügt. Die Ausprägung des Merkmals ist dann die Anzahl eigener Bauern auf diesem Rang abzüglich der Anzahl gegnerischer Bauern auf diesem Rang[2].
Für die anderen Figuren wird die Figurenzentralität verwendet. Das Spielfeld wird in

[2]Für die Figuren des schwarzen Spielers wird das Spielfeld dabei entsprechend gespiegelt, d.h. ein Bauer auf dem 2. Rang zählt als Bauer auf dem 7. Rang.

vier konzentrische quadratische Ringe eingeteilt. Dadurch ergeben sich für jede der vier übrigen Figuren jeweils vier neue Merkmale. Die Ausprägung von $c_{Springer/Innen}$ ist z.B. die Anzahl eigener Springer auf den mittleren vier Feldern abzüglich der Anzahl gegnerischer Springer auf diesen Feldern.
Die Werte in Tabelle 4.1 sind entstanden, indem auf die Werte der positionalen Merkmale (Bauernränge und Figurenzentralität) die Basiswerte der entsprechenden Figuren selbst addiert wurden. Anschließend wurden alle Werte normalisiert, so dass der Gesamtwert eines Bauern auf dem zweiten Rang 1,00 beträgt. An den Ergebnissen ist deutlich ab-

Bauernrang	Rang 2	Rang 3	Rang 4	Rang 5	Rang 6	Rang 7
Bauer	1,00	1,06	1,17	1,33	2,18	2,96
Figurenzentralität	Ring 1	Ring 2	Ring 3	Ring 4		
Springer	2,86	3,51	3,62	4,01		
Läufer	3,39	3,98	3,98	4,02		
Turm	5,21	5,72	5,84	5,86		
Dame	10,31	10,67	10,82	10,72		

Tabelle 4.1: Zusammengerechnete Figurenwerte (Ring 1 ist Außen, Ring 4 ist Innen)

zulesen, dass der Wert einer Figur je nach Position auf dem Spielbrett variieren kann. Besonders der Springer verliert deutlich an Wert je näher er am Rand steht. Dieses Ergebnis bestätigt den Ausspruch: "Springer am Rand bringt Schimpf und Schand!"
Das Turnier zum Abschluss bestätigt wieder die Güte der Werte. Spielt ein so trainiertes Programm gegen eine Version die rein materialbasiert evaluiert (mit gelernten Werten), dann gewinnt die neue Version des Programms 94% der Spiele. Daran ist zu erkennen, wie wichtig es ist, in die Evaluierungsfunktion etwas mehr Merkmale aufzunehmen als nur die positionsunabhängigen Materialsummen.
Zum Vergleich mit den traditionellen Werten können auch aus diesen Zahlen wieder reine Figurenwerte berechnet werden. Dabei muss nur beachtet werden, dass die verschiedenen Positionen unterschiedlich häufig besetzt sind. Stehen Bauern beispielsweise nur sehr selten auf dem siebten Rang dann geht dieser Wert auch nur zu einem entsprechend geringen Anteil in den Gesamtwert eines Bauern ein. Um dies zu berücksichtigen, wird über den Lernprozess hin mitgezählt, wie oft die Figuren auf den Rängen bzw. Ringen stehen; aus dieser Zählung ergibt sich Tabelle 4.2.
Nachdem die Werte zusammengerechnet und normiert wurden ergibt sich Tabelle 4.3.
Auch diese Werte weisen wieder große Ähnlichkeit auf mit den traditionellen.

Halbe Piece-Square-Tables

Um die Positionen der Figuren auf dem Spielbrett noch flexibler zu berücksichtigen können Piece-Square-Tables verwendet werden. In [BS99a] werden zunächst nur halbe Piece-Square-Tables betrachtet.
Die Evaluierungsfunktion erhält für jeden Figurentyp 32 neue Merkmale. So ein Merkmal

Bauernrang	Rang 2	Rang 3	Rang 4	Rang 5	Rang 6	Rang 7
Bauer	40,7%	27,2%	23,6%	6,5%	1,6%	0,5%
Figurenzentralität	**Ring 1**	**Ring 2**	**Ring 3**	**Ring 4**		
Springer	22,6%	23,8%	36,9%	16,7%		
Läufer	37,4%	30,5%	22,8%	9,3%		
Turm	65,1%	15,8%	14,7%	4,4%		
Dame	38,9%	31,3%	25,5%	4,3%		

Tabelle 4.2: Prozentuale Vorkommenshäufigkeit der Figuren auf den Positionen

Figur	Errechneter Wert
Bauer	1,00
Springer	3,16
Läufer	3,40
Turm	4,90
Dame	9,56

Tabelle 4.3: Aus den Tabellen 4.1 und 4.2 errechnete Figurenwerte

gibt an, wieviele Figuren eines Typs auf einem Feld in der linken Spielfeldhälfte stehen. Die rechte Spielfeldhälfte wird einfach als gespiegelte linke betrachtet. $C_{Bauer/A2} = 1$ könnte also bedeuten, dass ein weißer Bauer auf A2 und kein schwarzer auf A7[3] oder H7 steht. Stünde ein weißer Bauer auf A2 und ein schwarzer auf H7 so ergäbe sich $C_{Bauer/A2} = 1 + (-1) = 0$. Analog zu Tabelle 4.3 können auch hier wieder durchschnittliche Figurenwerte berechnet werden (Tabelle 4.4). Die geringen Abweichungen zu Tabelle

Figur	Errechneter Wert (Halbe Piece-Square-Tables)	Errechneter Wert Piece-Square-Tables
Bauer	1,00	1,00
Springer	2,67	2,76
Läufer	3,07	3,08
Turm	4,72	4,62
Dame	9,39	9,46

Tabelle 4.4: Aus halben und ganzen Piece-Square-Tables errechnete Figurenwerte

4.3 sind kaum relevant. Derartige Unterschiede sind statistischer Natur und können sich bereits zwischen verschiedenen Durchläufen des gleichen Experiments ergeben.
Das abschließende Turnier zeigt, dass halbe Piece-Square-Tables erfolgreicher sind als Bauernränge und Figurenzentralität. Die Programmversion mit den halben Piece-Square-Tables gewinnt gegen die Version mit gelernten Werten für die Ränge und Ringe von 2000

[3] Für die Figuren des schwarzen Spielers wird das Spielbrett wieder an der mittleren Querachse gespiegelt betrachtet.

Spielen etwa 61%.

Piece-Square-Tables

Bei Piece-Square-Tables wird im wesentlichen das gleiche gemacht wie bei halben Piece-Square-Tables. Der Unterschied besteht lediglich darin, dass nun für jedes Feld ein eigenes Merkmal gelernt wird. Nach Beal und Smith unterscheiden sich die Ergebnisse allerdings nicht nennenswert von denen aus den halben Piece-Square-Tables. Die einzige Ausnahme ist die Dame. Ihre Piece-Square-Table ist erkennbar nicht symmetrisch. Die errechneten Figurenwerte stehen in Tabelle 4.4.

Lernen der Steilheit der Stauchungsfunktion In [BS99a] wird auch versucht den Parameter ω der Stauchungsfunktion (siehe Gleichung 3.1 auf Seite 12) zu lernen. Wie in Abbildung 3.2 auf Seite 13 zu sehen ist, lässt sich über ω der Verlauf von $P(\nu)$ steuern. Eine Veränderung von ω führt zu einer Skalierung der gelernten Werte. Beal und Smith kommen dementsprechend zu dem Schluss, dass das Lernen von ω kaum Einfluss hat auf die endgültigen normierten Werte.

5 Herangehensweise

Dieses Kapitel behandelt Aufbau und Ablauf der Experimente, mit deren Hilfe auf die gestellte Aufgabe eingegangen wurde. Dazu werden die Einstellungen grundlegender Parameter beschrieben, die in der Durchführung der Experimente eine Rolle spielen, und das kleine und das große Experiment vorgestellt. Dabei handelt es sich um die zwei Experimente, die für jede der Schachvarianten durchgeführt wurden.
In den Abschnitten zwei und drei wird die verwendete Software beschrieben. Zum einen ist das das verwendete Schachprogramm und zum anderen eine Experimentierumgebung, die zwei Instanzen des Schachprogramms startet und so oft gegeneinander spielen lässt wie es in dem jeweiligen Experiment gewünscht ist.

5.1 Experimente

Für diese Arbeit wurden für jede betrachtete Schachvariante (einschließlich Standardschach) ein kleines und ein großes Experiment durchgeführt. Jedes Experiment bestand aus mehreren Durchläufen, über deren Ergebnisse zum Abschluss der Durchschnitt berechnet wurde.
Die Suchtiefe wurde in allen Experimenten auf 5 eingestellt, unterlag aber Veränderungen durch Quiescence (siehe Abschnitt 2.2.2). In allen Experimenten hat DaSunsetter (das hier verwendete Schachprogramm; siehe dazu Abschnitt 5.2.2) ausschließlich aus automatischem Spiel gegen sich selbst bzw. gegen Alterego (siehe Abschnitt 5.2.2) gelernt. Es wurde nie gegen einen menschlichen Spieler gespielt, und es wurden auch keine Endspieldatenbanken oder Eröffnungsbibliotheken verwendet; d.h. es ist kein Expertenwissen in den Lernprozess eingeflossen.
Um während des Lernvorgangs Wiederholungen von Spielen zu reduzieren (oder sogar ganz zu vermeiden) wurde der in einer gegebenen Situation durchzuführende Zug zufällig bestimmt, wenn das Suchverfahren mehrere Züge gleich bewertet hat.
Die zu lernenden Gewichte (z.B. die Figurenwerte) wurden zu Beginn eines Durchlaufs immer alle auf 1 gesetzt (analog zu [BS97]).

5.1.1 Kleine und Große Experimente

Die kleinen Experimente bestanden aus 10 Durchläufen zu je 2000 Spielen und lernten eine rein materialbasierte Evaluierungsfunktion (analog zu [BS97]). Die Anzahl zu lernender Gewichte ergab sich dabei aus den Varianten selbst. Bei Standard- und Atomschach wurde für jede Figur außer dem König ein Wert gelernt. Bei Crazyhouse wurden zusätzlich Werte für die Figuren auf der Hand gelernt, und bei Räuberschach wurde für jede Figur (einschließlich König) ein Wert gelernt.

Die großen Experimente bestanden aus 20 Durchläufen zu je 10000 Spielen und lernten eine Evaluierungsfunktion mit Materialwerten und ganzen Piece-Square-Tables (siehe Abschnitt 4.3.2). Bei allen Varianten wurde für jede Figur[1] eine eigene Piece-Square-Table gelernt. Beal und Smith schreiben in [BS99a] auf Seite 3 zwar, es sei allgemein bekannt, dass es von der *Spielphase* abhängig ist, ob der König in der eigenen Aufstellungszone oder in der Mitte des Spielbretts besser aufgehoben ist, auf Schachvarianten ist dies aber nicht unbedingt übertragbar. Daher (und zu Vergleichszwecken) wurde bei allen Varianten auch eine Piece-Square-Table für den König gelernt. Dies führte in den großen Experimenten zu $6 * 64 = 384$ zusätzlichen Gewichten.

5.1.2 Turnier

Zur Betrachtung der Güte der gelernten Werte aus beiden Experimenten einer Schachvariante wurden die Werte zum Abschluss in einem Turnier erprobt. In Anlehnung an Beal und Smith haben zwei Programme immer 2000 mal gegeneinander gespielt, und die Ausgabe der Evaluierungsfunktion wurde um einen geringen Wert (1% des Evaluierungswertes) zufällig verändert, wenn DaSunsetter mit Piece-Square-Tables gespielt hat. Dadurch sollten Wiederholungen von Spielen reduziert werden, denn ohne diesen Faktor hätte das Programm in der gleichen Situation wahrscheinlich jedes mal den gleichen Zug ausgewählt, da (durch die Piece-Square-Tables) nahezu jede mögliche Situation auf dem Spielbrett zu einer anderen Evaluierung führt.
Die Suchtiefe wurde immer für beide Programme auf 5 Halbzüge eingestellt und eine eventuell vorhandene Eröffnungsbibliothek oder Endspieldatenbank eines fremden Programmes wurde entfernt.

5.1.3 Standardschach

Die Experimente mit Standardschach dienen hier Vergleichszwecken. Bereits während der Arbeit an DaSunsetter zeigte sich, dass Mechanismen, die die Spielweise des Programms verändern, auch die Höhe der gelernten Werte beeinflussen können. Daher eignen sich die gelernten Standardschachwerte eines anderen Programms nicht unbedingt zum Vergleich mit Werten, die das hier verwendete Programm für die Varianten lernt bzw. gelernt hat.

5.2 Das Schachprogramm

Zur Durchführung der Experimente dieser Arbeit wurde ein Schachprogramm benötigt, das heute übliche Mechanismen (wie Alpha-Beta und Transpositionstabellen) ebenso beherrscht, wie die zu betrachtenden Schachvarianten und den Lernalgorithmus. Die Beherrschung von Alpha-Beta und weiteren Techniken sollte einerseits die Experimente beschleunigen und andererseits den Spielen eine praxisnahe Umgebung bieten. Letzteres war wichtig, weil manche der Techniken (z.B. Quiescence) einen Einfluss haben können auf die Höhe der gelernten Werte.

[1] In Crazyhouse wurden nur Piece-Square-Tables gelernt für Figuren auf dem Spielbrett.

5.2.1 Sunsetter

Das Schachprogramm dieser Arbeit, DaSunsetter, wurde erstellt auf der Grundlage von Sunsetter, einem Open-Source-Programm, das über Sourceforge[2] erhältlich ist. Dort gibt es eine Version C10 vom 09.11.2003, die Standardschach spielt, und eine Version 7e vom 19.10.2002, die Crazyhouse und Bughouse[3] spielt. Das Programm verwendet Alpha-Beta zur Spielbaumsuche und enthält sowohl Transpositionstabellen als auch eine Form von Quiescence, die dort Search Extensions genannt wird. Sunsetter beherrscht auch einen Lernmechanismus, der allerdings lediglich in persistenten Transpositionstabellen besteht. Außerdem kann das Programm eine Endspieldatenbank verwenden, die aber im Paket auf Sourceforge nicht enthalten ist.

5.2.2 DaSunsetter

Zur Erfüllung aller oben genannten Kriterien wurde Sunsetter umgebaut zu DaSunsetter. Das Programm wurde von C zu C++ konvertiert und mittels Polymorphie ein System eingebaut, durch das DaSunsetter *mehrere* Schachvarianten beherrschen kann, von denen vor Spielbeginn über einen XBoard-Befehl (siehe Appendix A) eine ausgewählt werden kann. Für Crazyhouse wurde zum Großteil der Code aus der Version 7e verwendet, der Code für Räuber- und Atomschach dagegen ist neu.
Weiterhin wurde ein Lernalgorithmus eingebaut und nicht benötigte Komponenten entfernt.

TDLeaf(λ)

Der Lernmechanismus wurde in Form des oben (Abschnitt 4.1) beschriebenen TDLeaf realisiert. Dabei ist zu beachten, dass das Ergebnis der (in DaSunsetter veränderten) Evaluierungsfunktion nun nicht mehr nur aus der Evaluierung selbst besteht, sondern auch aus allen Daten, die dem Spielbrett entnommen wurden und später zum Lernen benötigt werden; das sind beispielsweise die Anzahlen der verschiedenen Figuren auf dem Spielbrett. Diese Daten müssen allerdings auch in der Transpositionstabelle (siehe Abschnitt 2.2.2) gespeichert werden. Da der Speicherverbrauch der zusätzlichen Daten recht hoch werden kann und die Transpositionstabelle in Sunsetter standardmäßig 1.048.576 Einträge vorsieht, wächst der Speicherverbrauch der Transpositionstabelle in astronomische Höhen. Der Speicherverbrauch (Arbeitsspeicher) von DaSunsetter lag durch diesen Effekt zeitweise bei etwa 1,5–2,0GB. Durch eine geschicktere Speicherung der Werte konnte dieser Bedarf auf etwa 120MB reduziert werden.

Temporal Coherence

Die Lernrate α des Lernverfahrens wird mittels Temporal Coherence (siehe Abschnitt 4.2) bestimmt. Dazu werden die benötigten Werte N_i und A_i zusammen mit den zu lernenden Werten in *einer* Datei gespeichert.

[2] `http://sourceforge.net/project/showfiles.php?group_id=61676`
[3] Bughouse wird im Deutschen üblicherweise als Tandemschach bezeichnet.

Zu Beginn des Verfahrens werden N_i und A_i immer mit 2 und 8 initialisiert, die globale Lernrate c beträgt 0,5. Daraus ergibt sich eine anfängliche Gesamtlernrate von $\frac{|2|}{8} * 0.5 = 0,125$.

Alterego

Um DaSunsetter gegen sich selbst spielen zu lassen, wurde eine Variante namens Alterego konstruiert. Spielt DaSunsetter bei eingeschaltetem Lernmodus gegen Alterego, so speichert Alterego während des Spiels alle Daten, die zum lernen nötig sind, in einer Datei. Wenn DaSunsetter nach dem Spiel aus den eigenen Daten sein ΔW_t berechnet hat, wiederholt es den Lernvorgang mit den Daten von Alterego und addiert das daraus ermittelte ΔW_t zu dem vorher berechneten.
Das mag für TD(λ) sinnlos erscheinen, da die beiden Programme Daten aus dem gleichen Spiel bzw. dem gleichen Spielverlauf gesammelt haben, für TDLeaf(λ) verhält es sich aber ein wenig anders. Die Daten entstehen bei TDLeaf(λ) aus dem Verlauf der Principal Positions statt aus dem der tatsächlich ereigneten Positionen. Dadurch ergibt sich ein Verlauf, in dem der Unterschied zwischen einer von Sunsetter betrachteten (Principal) Position und der im folgenden Halbzug von Alterego betrachteten (Principal) Position recht groß sein kann. Es handelt sich also nicht einfach um zwei Ansichten der gleichen Positionsfolge, sondern die Folgen können sich stellenweise recht stark unterscheiden.
Um die Wirkung von Alterego zu betrachten wurde ein Experiment durchgeführt, in dem

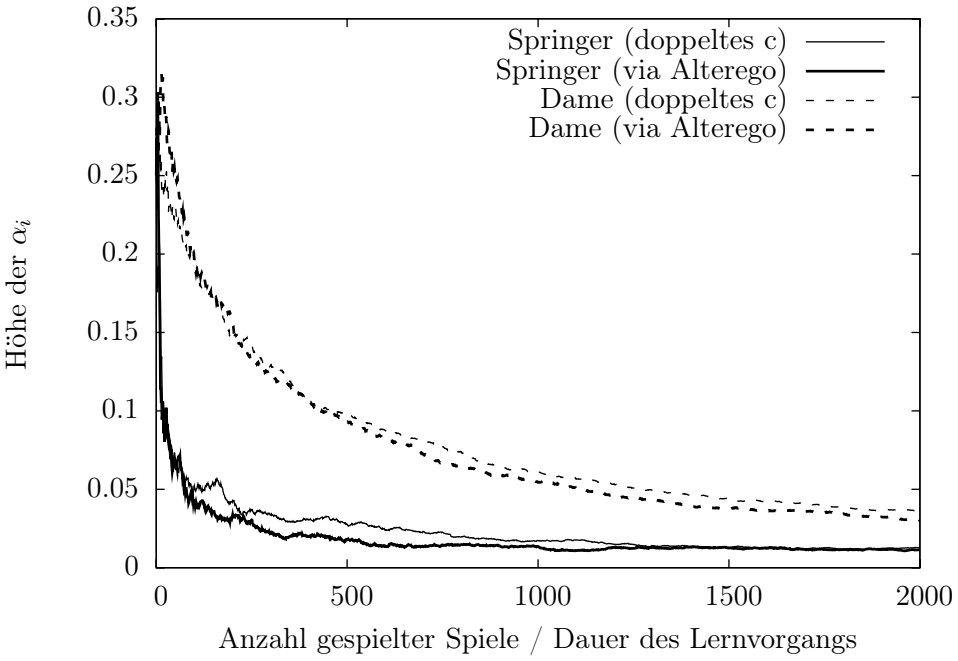


Abbildung 5.1: Vergleich der Alphas beim Lernen mit/ohne Alterego

eine rein materialbasierte Evaluierungsfunktion gelernt wurde. Das Experiment wurde

zweimal durchgeführt: Einmal spielte DaSunsetter gegen Alterego und das andere Mal gegen eine nicht lernende Version seiner selbst, wobei in letzterem Fall die globale konstante Lernrate c von 0,5 auf 1 verdoppelt wurde. Die Verdoppelung der Lernrate soll in diesem Fall den Effekt von Alterego ausgleichen, dass das Gesamtdeltas nach jedem Spiel etwa doppelt so hoch sind wie in Versuchen ohne Alterego. In zehn Durchgängen wurde über jeweils 2000 Spiele gelernt und anschließend der Schnitt aus den zehn Durchgängen berechnet. Hier sollen aber nicht die gelernten Werte interessieren, sondern die lokalen Lernraten α_i. In Abbildung 5.1 ist der durchschnittliche Verlauf von α_{Dame} und $\alpha_{Springer}$ aus beiden Experimenten dargestellt. Es ist zu erkennen, dass die α_i in dem Durchgang mit Alterego ein wenig schneller abfallen. Betrachtet man die lokalen Lernraten als Konvergenzindiz, so zeigt das Experiment, dass die Verwendung von Alterego sich zwar nur leicht aber dennoch positiv auf den Lernvorgang auswirkt.

5.2.3 Kürzungen

Der alte Lernmechanismus und die Schnittstelle zur Endspieldatenbank wurden entfernt. Dies wäre ohnehin eine Form von Expertenwissen gewesen, welches hier aber nicht mit einfliessen sollte (siehe Abschnitt 5.1). Auch Pondering funktioniert in DaSunsetter nicht mehr. Pondering ist ein Mechanismuses, durch den ein Schachprogramm versucht während des gegnerischen Zuges bereits den nächsten eigenen zu planen. Der Erhalt dieses Mechanismus beim Umbau des Programms wäre recht arbeitsaufwendig gewesen und hätte den Experimenten keinen Nutzen gebracht.
Unter den oben erwähnten Search Extensions ist auch eine Erweiterung, die die Suchtiefe erhöht, wenn ein Bauer auf dem siebten Rang, d.h. kurz vor einer Beförderung, steht. Es hat sich in frühen Experimenten mit DaSunsetter gezeigt, dass dadurch der Wert der Bauern für das Programm erhöht wird. Der Wert, der für einen Bauern gelernt wurde, war dadurch, gemessen an den klassischen Figurenwerten, sehr hoch. Das bedeutet eine Verzerrung, die den Vergleich mit klassischen Werten oder denen von Beal und Smith deutlich erschwert. Da diese Sucherweiterung außerdem, je nach Schachvariante, ihre Bedeutung ändern kann, wurde sie entfernt.
Hier zeichnete sich bereits früh ab, dass die Höhe der optimalen Werte u.a. abhängig ist von den in einer Schachsoftware eingesetzten Mechanismen, zumindest wenn diese die (taktische) Spielweise des Programms beeinflussen.

5.3 Experimentierumgebung

Zur Durchführung der Experimente wurde ein Programm benötigt, das die Schachprogramme startet und Nachrichten zwischen ihnen weiterleitet. Außerdem müssen manche XBoardnachrichten vor dem Weiterleiten übersetzt werden: Wird ein Zug *zum* Schachprogramm geschickt, so lautet die Nachricht z.B. `d2d4`, wird aber der gleiche Zug *vom* Schachprogramm geschickt, so lautet die Nachricht `move d2d4`.
Das Programm sollte in der Lage sein mehrere unabhängige Lerndurchläufe zu jeweils mehreren Spielen ablaufen zu lassen und abschließend evtl. noch den Durchschnitt aus

den Ergebnissen zu berechnen. Weiter sollte es die Möglichkeit bieten, zwei Schachprogramme in einem Turnier gegeneinander spielen zu lassen und anschließend anzuzeigen, wieviele Spiele die Programme jeweils gewonnen haben.

5.3.1 XBoards Turniermodus

Mit dem Parameter -mg bietet das Programm XBoard einen Turniermodus, über den man eine beliebige Zahl von Spielen zwischen zwei Programmen laufen lassen kann. Dieser Modus eignet sich zwar für einige wenige Spiele, für Experimente mit 10–20 Durchläufen zu je 2000–10000 Spielen jedoch ist er zu imperformant. Er bietet auch keine Möglichkeit mehrere Durchläufe zu starten und deren Lernergebnisse voneinander zu trennen.

5.3.2 ChessExperimenter

Da die Fähigkeiten von XBoard den oben beschriebenen Kriterien nicht gerecht werden, wurde für die Durchführung der Experimente ein Programm namens ChessExperimenter erstellt. Das Programm ist darauf ausgelegt, einen zügigen Ablauf der Spielserien zu ermöglichen und benennt nach jedem Durchlauf die von DaSunsetter erzeugte Lerndatei um, damit der Lernvorgang im nächsten Durchlauf wieder mit neuen Initialwerten beginnt. Weiterhin enthält ChessExperimenter Funktionen zur Durchschnittsberechnung der gelernten Werte über die Durchläufe und zur Berechnung der Standardabweichung bei den finalen Werten. Ein beispielhafter Aufruf des Programms für ein Experiment mit 10 Durchläufen zu je 2000 Spielen lautet:

```
./chessExperimenter -w ./sunsettermat -b ./alteregomat -g 2000 -r 10
```

Dadurch wird ein Experiment gestartet mit 10 Durchläufen zu je 2000 Spielen. Im ersten Spiel spielt DaSunsetter weiß und Alterego schwarz, danach werden die Seiten nach jedem Spiel vertauscht. Es ist auch ein Turniermodus vorhanden, in dem der Lernmodus der Programme nicht aktiviert wird, und an dessen Ende angezeigt wird, welches Programm wie oft gewonnen hat.

6 Resultate

In diesem Kapitel werden die Resultate aus den Experimenten, die im vorigen Kapitel beschrieben wurden, dargestellt und analysiert. Das Kapitel ist in vier Abschnitte unterteilt von denen sich jeder einer eigenen Schachvariante widmet. Jeder Abschnitt beginnt mit einer Vorstellung der Resultate aus dem kleinen Experiment, in dem nur Materialwerte gelernt wurden. Anschließend folgt ein größerer Teil, der die Ergebnisse aus dem großen Experiment enthält, in dem auch Gewichte gelernt wurden für die Positionen der Figuren. In einem dritten Teil werden Ergebnisse gezeigt aus Turnieren, mit deren Hilfe unterschiedliche Figurenwerte und die beiden unterschiedlichen Evaluierungsfunktionen (aus dem kleinen und dem großen Experiment) verglichen wurden.

6.1 Standardschach

6.1.1 Kleines Experiment

In Abbildung 6.1 ist der durchschnittliche Verlauf der 5 gelernten Werte über die gesamte Lerndauer (2000 Spiele) zu sehen. Im Diagramm ist zu erkennen, dass die Werte sich ab dem 500. Spiel kaum noch geändert haben. Die Länge eines Durchlaufs ist mit 2000 Spielen also ausreichend hoch gewählt worden. Zu Vergleichszwecken und aus Gründen der Anschaulichkeit wurden alle für Standardschach gelernten Werte (sowohl im kleinen als auch im großen Experiment) so normiert, dass der Wert eines Läufers genau 3 beträgt. Die Normierung[1] der Bauern auf 1 wurde hier nicht verwendet, weil die Bauernwerte recht hoch ausgefallen sind und eine entsprechende Normierung *sämtliche* Werte verzerrt hätte.
Die finalen Durchschnittswerte sind mit Standardabweichung in Abbildung 6.3 dargestellt.

6.1.2 Großes Experiment

Die im großen Experiment gelernten Gewichte für die Piece-Square-Tables sind in Abbildung 6.2 dargestellt[2]. Das weiße Feld in einem Diagramm gibt die Position auf dem Schachbrett an, auf der die Figur den höchsten Wert hat, das schwarze Feld markiert die Stelle mit dem niedrigsten Wert. Bei jedem Diagramm sind der Minimal- und der Maximalwert angegeben, also der Wert der jeweiligen Figur auf dem weißen bzw. dem

[1] Die Normierung ändert nichts an der Gültigkeit der Werte, da für die Evaluierungsfunktion nur der relative Wert der Figuren zueinandner eine Rolle spielt.
[2] Der Diagrammstil entspricht dem in [BS99a] auf Seite 8. Tatsächlich sehen die einzelnen Diagramme in Abbildung 6.2 invertiert denen in [BS99a] sehr ähnlich.

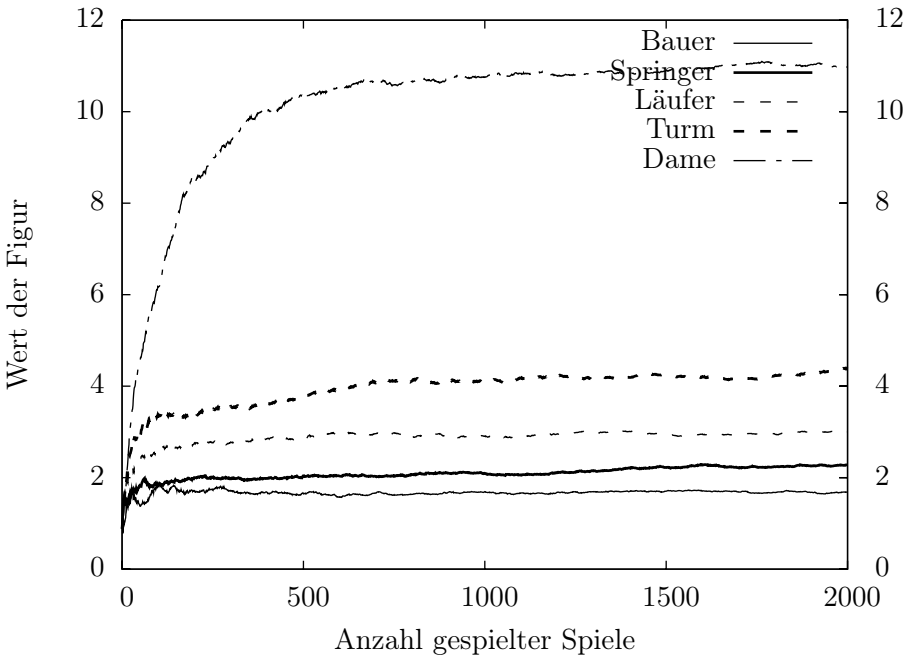


Abbildung 6.1: Entwicklung der Materialwerte

schwarzen Feld.
Abbildung 6.2.1 zeigt, wie der Wert eines Bauern mit seinem Rang auf dem Spielbrett steigt. Die Felder der Ränge 1 und 8 enthalten alle ihren Initialwert. Da diese Felder nie von Bauern betreten wurden, hat sich in der Aktualisierungsformel von $TD(\lambda)$ (Gleichung 3.2) nie eine Änderung der Werte ergeben. Beispielsweise gilt immer $c_{Bauer/A1} = 0$ in Gleichung 3.6, und in Gleichung 3.2 führt das zu $\Delta_{t/Bauer/A1} = 0$, unabhängig von t.
In Abbildung 6.2.2 ist zu sehen, dass ein Springer einen geringeren Wert hat je näher er sich am Rand befindet. Das lässt sich über die reduzierte Mobilität erklären. Im Gegensatz zu den anderen Offizieren (Läufer, Turm, Dame) hat ein Springer am Rand weniger gültige Züge als in der Mitte des Spielbretts.
Der Läufer verliert zum Rand hin zwar ebenfalls an Wert, wie man an den Werten für Minimum und Maximum erkennen kann, hat der Unterschied hier aber eine etwas geringere Relevanz.
Den erhöhten Wert des Turms auf den höheren Rängen erklären Beal und Smith in [BS99a] mit der Möglichkeit die gegnerischen Bauern und den König zu bedrohen.
Im Diagramm für den König beziehen Min und Max sich nicht auf den Gesamtwert der Figur sondern lediglich auf die normierten Piece-Square-Table-Einträge, da der Basiswert des Königs im Standardschach theoretisch bei ∞ liegt. Die hellen Felder im unteren Bereich des Diagramms resultieren vermutlich aus einer günstigen Position des Königs im Mittelspiel. Die hellen Felder im oberen Bereich (um F6 herum) beziehen sich dagegen auf eine günstige Position des Königs im Endspiel, wenn es für ihn wichtig ist, mobil zu

bleiben und nicht in eine Ecke gedrängt zu werden.

Aus den gelernten Basiswerten und den Piece-Square-Tables wurden durchschnittliche Figurwerte errechnet. Jeder Wert in einer Piece-Square-Table geht dabei zu einem Anteil in die Rechnung ein, der der relativen Vorkommenshäufigkeit der Figur auf dem jeweiligen Feld entspricht.[3] Die Vorkommenshäufigkeit der Figuren auf den Feldern wurde dazu während des Lernvorgangs mitgezählt. Die errechneten Materialwerte sind mit ihrer Standardabweichung[4] in Abbildung 6.3 dargestellt. Zum Vergleich sind in Tabelle 6.1 die Werte aus beiden Experimenten zusammen mit denen aus [BS97, BS99a] und den Originalwerten aus Sunsetter C10 angegeben. Die Werte in der Tabelle wurden allerdings alle auf die gleiche Weise normiert, um den Vergleich zu erleichtern. Die Werte stimmen

	Großes Exp.	Kleines Exp.	Sunsetter	[BS97] Trl C	[BS99a] Fullbrd
Bauer	1,05	1,20	0,96	0,96	0,97
Springer	2,82	2,43	2,87	2,14	2,69
Läufer	**3,00**	**3,00**	**3,00**	**3,00**	**3,00**
Turm	4,51	4,35	4,49	4,51	4,50
Dame	9,38	9,48	8,60	8,82	9,21

Tabelle 6.1: Vergleich verschiedener Sätze von Materialwerten (Standardschach)

weitestgehend miteinander überein, geringe Unterschiede können erklärt werden durch die statistische Natur von $TD(\lambda)$. Werden im Lernvorgang andere Spiele gespielt, so ändern sich auch die gelernten Werte leicht. Dies wird z.B. bestätigt durch die unterschiedlichen Ergebnisse der Trials A bis E in [BS97] und die dargestellten Standardabweichungen in Abbildung 6.3.

6.1.3 Turnier

Die Ergebnisse des Turniers sind in Tabelle 6.2 angegeben. Die Einträge in der Spalte Konfiguration haben folgende Bedeutungen:

gelernt DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den Materialwerten, die im kleinen Experiment gelernt wurden

klassisch DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und klassischen Materialwerten

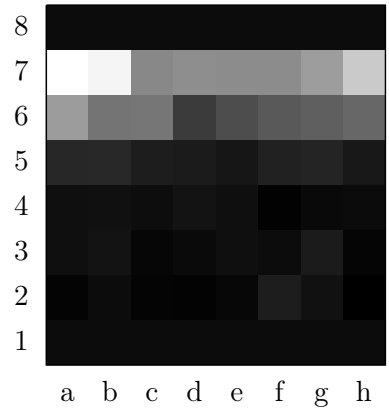
original DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den Materialwerten, die in Sunsetter C10 als Konstanten eingetragen sind

errechnet DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den errechneten Materialwerten des großen Experiments aus Tabelle 6.1

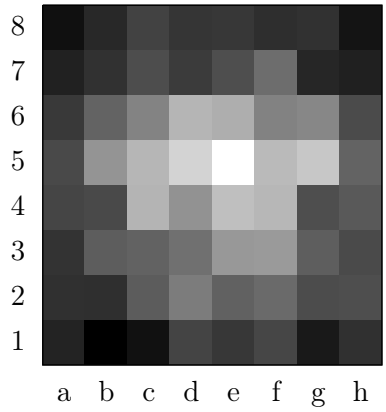
piece-square DaSunsetter mit der gleichen Evaluierungsfunktion wie im großen Experiment und den dort gelernten Werten

[3] Dies wurde analog zu der Berechnung in [BS99a] durchgeführt.

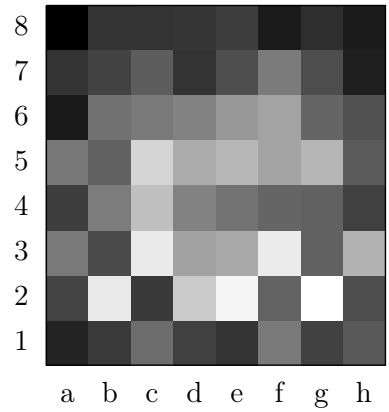
[4] Die Standardabweichung bezieht sich auf die Abweichung zwischen den 20 Durchläufen.



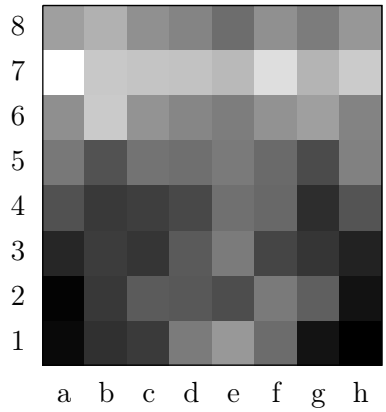
6.2.1: Bauer (Min/Max: 0,92/2,86)



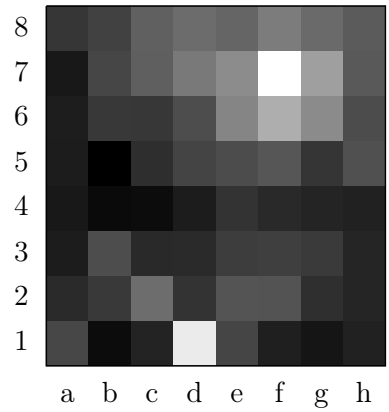
6.2.2: Springer (Min/Max: 2,61/3,13)



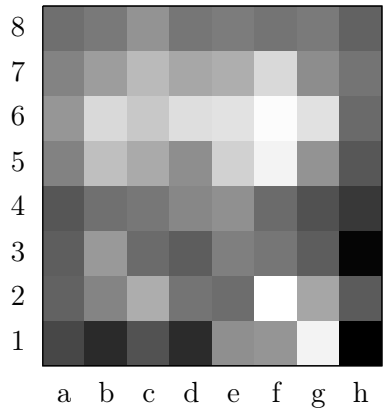
6.2.3: Läufer (Min/Max: 2,83/3,12)



6.2.4: Turm (Min/Max: 4,41/4,9)



6.2.5: Dame (Min/Max: 9,1/9,55)



6.2.6: König (Min/Max: 0,60/0,85)

Abbildung 6.2: Piece-Square-Tables in Standardschach

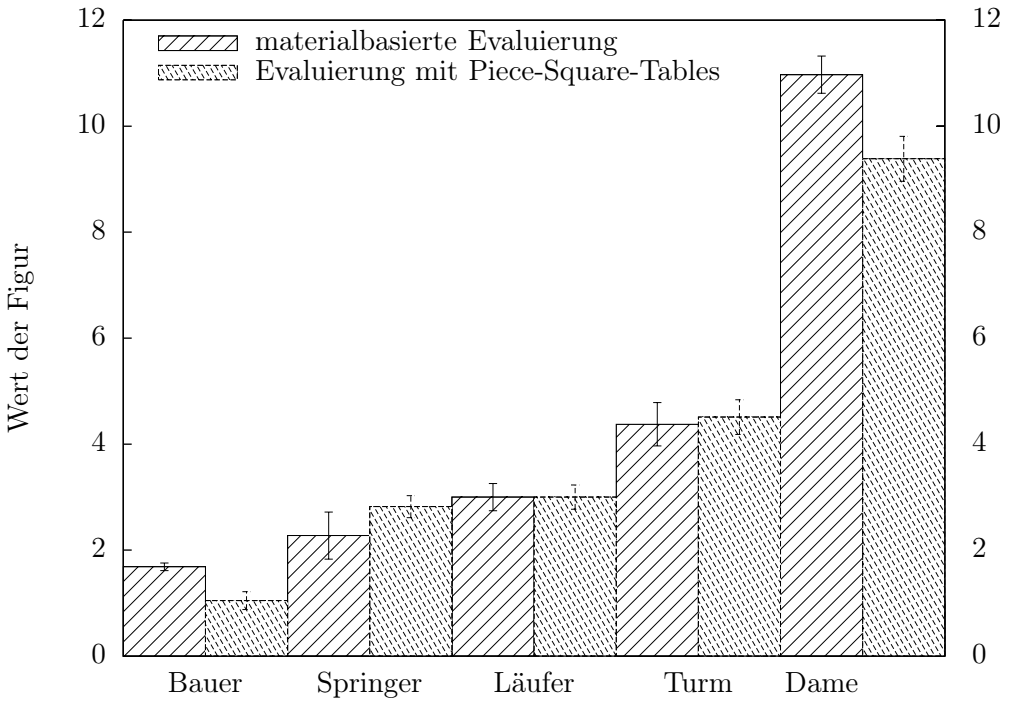


Abbildung 6.3: Finale Materialwerte der einzelnen Figuren

sjeng Das Schachprogramm Sjeng[5] in der Version 11.2

Konfiguration	Gewonnen	Verloren	Unentschieden	Quote
gelernt vs. klassisch	1246	715	39	63,28%
gelernt vs. original	1037	906	57	53,28%
gelernt vs. errechnet	971	993	36	49,45%
piece-square vs. gelernt	1735	245	20	87,25%
piece-square vs. sjeng	2	1997	1	0,13%

Tabelle 6.2: Ergebnisse der Turniere

Die Ergebnisse zeigen, dass die gelernten Werte tatsächlich zu einem stärkeren Spiel führen als anderweitig erhaltene Werte und dass das Programm bei Verwendung der Piece-Square-Tables deutlich stärker spielt als mit einer rein materialbasierten Evaluierung. Überraschend ist hier die Gewinnquote von leicht weniger als 50% beim Spiel der gelernten Werte gegen die errechneten. Hier wäre zu erwarten gewesen, dass das Programm mit den Werten am besten spielt, die gelernt wurden für die gleiche Evaluierungsfunktion die auch im Turnier verwendet wurde. Die letzte Zeile zeigt, dass das

[5] `http://www.sjeng.org/indexold.html`

Lernen der Evaluierungsfunktion alleine noch keinen Schachmeister ausmacht. Einem normalen Computergegner ist auch die Variante mit Piece-Square-Tables hoffnungslos unterlegen.

6.2 Crazyhouse

6.2.1 Kleines Experiment

Der durchschnittliche Verlauf der gelernten Werte für Figuren auf dem Spielbrett ist in Abbildung 6.4 zu sehen. Hier wurde wieder genauso normiert wie bei Standardschach, d.h. ein Läufer auf dem Spielbrett hat einen Wert von 3. Die Kurven sehen denen aus

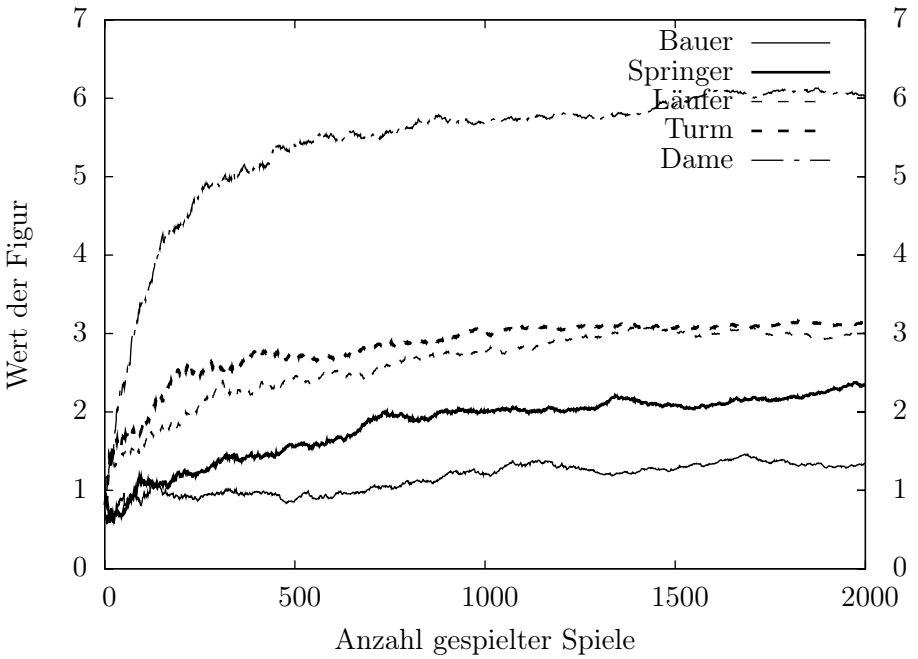


Abbildung 6.4: Entwicklung der Figurenwerte auf dem Spielbrett (Crazyhouse)

dem Standardschach sehr ähnlich. Die Form der Kurven entspricht der in Abbildung 6.1, und die Werte für Bauern und Springer sind auch fast die gleichen. Die beiden starken Figuren, Turm und Dame, haben allerdings deutlich an Wert verloren. Der Wert der Dame hat sich fast halbiert und der Wert eines Turms entspricht jetzt etwa dem eines Läufers. Der niedrige (relative) Wert der Dame ergibt sich vermutlich durch die erhöhte allgemeine Mobilität auf dem Spielbrett, die durch die Platzierungen der Handfiguren entsteht. Durch diese teleportationsartigen Züge verlieren die vielen Zugmöglichkeiten der Dame an Wert.
Abbildung 6.5 zeigt die Verläufe für Figuren auf der Hand. Hier fällt zunächst auf, dass die Kurven (besonders die der Dame) langsamer ihren Zielwert erreichen. Dies lässt sich darüber erklären, dass die entsprechenden Merkmale nicht so häufig ausgeprägt sind, da beispielsweise eine Dame, die auf die Hand genommen wurde, relativ schnell wieder auf das Spielbrett gesetzt wird. Das bedeutet natürlich nicht, dass der Algorithmus nicht funktioniert, aber es zeigt, dass bereits eine kleine Regeländerung einen deutlichen Einfluss haben kann auf das Konvergenzverhalten. Auffällig ist auch, dass drei Figuren

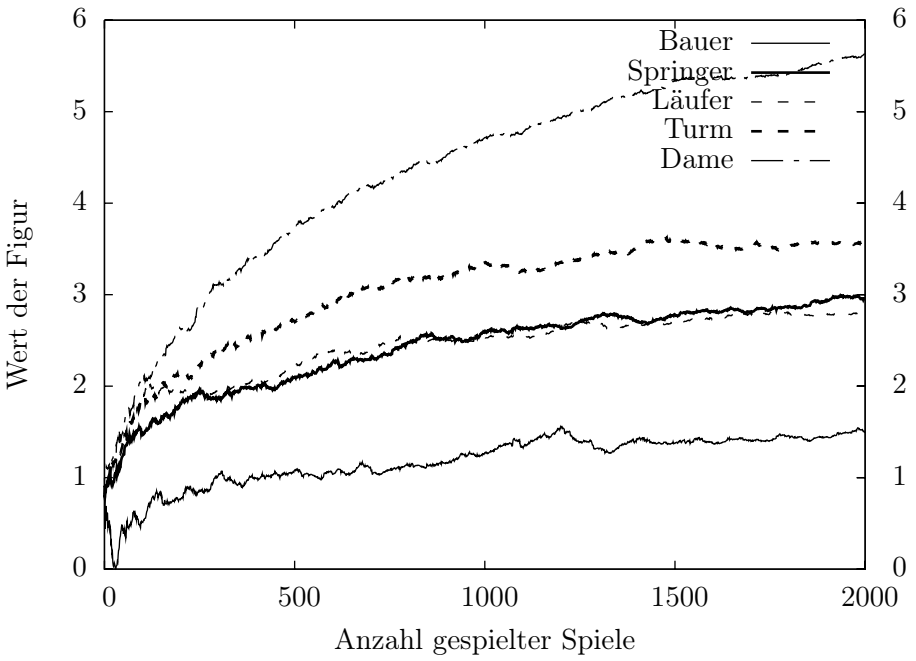
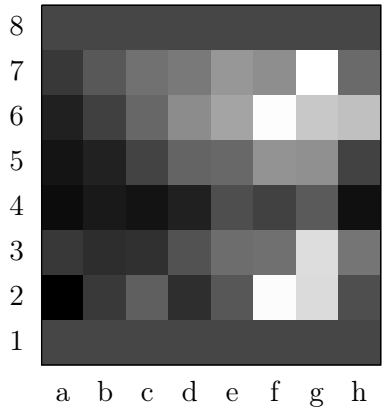


Abbildung 6.5: Entwicklung der Materialwerte für Figuren auf der Hand (Crazyhouse)

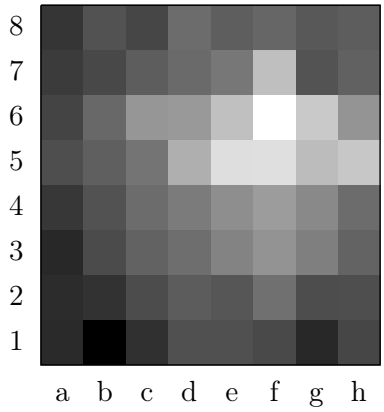
(Bauer, Springer, Turm) auf der Hand einen höheren Wert haben als auf dem Feld. Das liegt vermutlich daran, dass die Stärke dieser Figuren stark abhängig ist von ihrer Position auf dem Spielbrett. Das große Experiment sollte hier zeigen, ob diese Theorie bestätigt werden kann, da dann der Minimal- und der Maximalwert der jeweiligen Figur relativ weit auseinander liegen müssten.

6.2.2 Großes Experiment

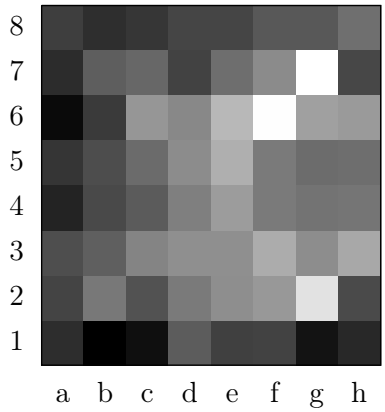
Abbildung 6.6 zeigt die gelernten Piece-Square-Tables für Crazyhouse. Im Groben sehen die Diagramme denen vom Standardschach recht ähnlich, aber es sind doch Unterschiede zu sehen, die sich wiederum recht gut über die Handfiguren bzw. die Platzierzüge (wenn eine Handfigur auf das Spielbrett gesetzt wird) erklären lassen. Besonders stark unterscheidet sich das Diagramm des Bauern von seiner Entsprechung im Standardschach. Da ein Bauer über einen Platzierzug einfach auf die siebte Reihe gesetzt werden kann nimmt sein Wert mit steigendem Rang nicht unbedingt zu. Der hohe Wert auf den vorderen Reihen erklärt sich zwar immernoch durch die bevorstehende Beförderung, der Wert auf den hinteren Reihen ist aber sehr wahrscheinlich darauf zurückzuführen, dass die Bauern hier Bedrohungslinien sehr gut blockieren können. Das spielt bei Crazyhouse eine besondere Rolle, weil solche Bedrohungslinien durch Platzierzüge schlagartig von jedem freien Feld aus aufgebaut werden können.
Im Diagramm des Springers ist das wertvollste Feld ein wenig näher zum Rand gewandert gegenüber Standardschach. Da das weiße Feld in Abbildung 6.6.2 durch die Evaluierungs-



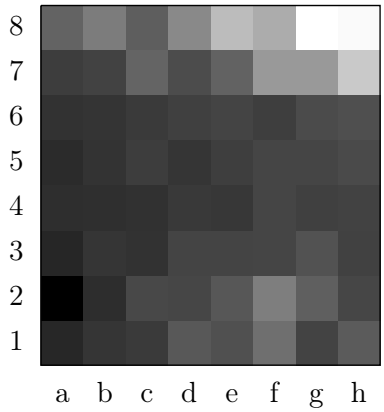
6.6.1: Bauer (Min/Max: 0,66/2,14)



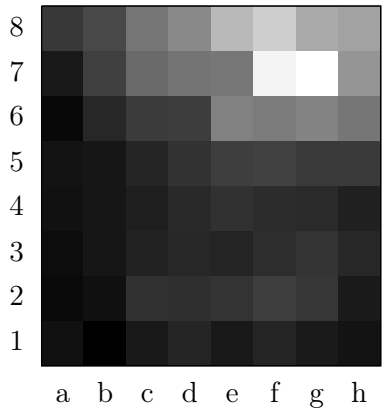
6.6.2: Springer (Min/Max: 1,53/3,97)



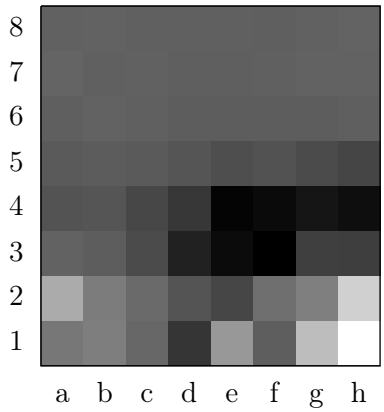
6.6.3: Läufer (Min/Max: 2,38/3,87)



6.6.4: Turm (Min/Max: 2,3/4,8)



6.6.5: Dame (Min/Max: 4,38/6,91)



6.6.6: König (Min/Max: -0,97/4,25)

Abbildung 6.6: Piece-Square-Tables in Crazyhouse

funktion das favorite Feld für einen Platzierungszug ist, könnte sich diese Verschiebung dadurch erklären, dass ein Springer von diesem neuen wertvollsten Feld (F6) die Anfangsposition des Königs bedroht. Im Falle eines zur Verfügung stehenden Handspringers lässt sich so blitzartig eine Bedrohung auf den gegnerischen König aufbauen.
Die Erkennung eines prägnanten Musters in Abbildung 6.6.3 ist schwer, aber es fällt auf, dass die Spanne zwischen dem minimalen und dem maximalen Wert deutlich größer ist als im Standardschach. Das lässt vermuten, dass die absolute Position des Läufers auf dem Spielbrett wichtiger ist als im Standardschach.
Die Stärke des Turms liegt jetzt nicht mehr allgemein im vorderen Spielfeldbereich, sondern konzentiert sich auf die vordere rechte Ecke (H8). Dies korreliert mit dem wertvollsten Springerfeld, dessen Verschiebung ebenfalls auf die rechte vordere Spielbrettecke zielt.
Auch bei der Dame ist ein ähnlicher Trend zu beobachten. Das Diagramm der Dame hatte zwar bereits im Standardschach einen hellen Bereich in der rechten vorderen Zone des Spielbretts, das Feld D1 (das Ausgansfeld der Dame), welches im Standardschach noch sehr hell gefärbt war, verschwindet nun aber in einem allgemein dunklen Bereich. Auch die Wertespanne der Dame ist erheblich größer als im Standardschach.
Der große graue Bereich in Abbildung 6.6.6 entsteht durch den Wegfall des Endspiels. Da keine Figur das Spiel endgültig verlässt, entsteht effektiv nie eine Situation, die dem üblichen Endspiel entspricht. Der helle Bereich aus Abbildung 6.2.6 findet sich hier daher nicht wieder und die oberen Ränge (Reihe 5–8) werden vom König fast nie betreten. Daher konnten innerhalb der durchgeführten 10000 Spiele auch keine aussagefähigen Werte für diese Felder gelernt werden.
Die durchschnittlichen Materialwerte für Figuren auf dem Feld, die aus den Piece-Square-Tables errechnet wurden, sind mit ihrer jeweiligen Standardabweichung in Abbildung 6.7 dargestellt[6]. Die Normierung entspricht der im Standardschach in Abschnitt 6.1.
Zum Vergleich sind in Tabelle 6.3 die Werte aus dem kleinen Experiment, die errechneten Werte aus dem großen Experiment, die Werte aus Sunsetter 7e und die Werte, die Sjeng verwendet, aufgelistet.

	Großes Exp.	Kleines Exp.	Sunsetter	Sjeng
Bauer	1,26	1,34	1,54	1,30
Springer	2,54	2,35	2,95	2,74
Läufer	**3,00**	**3,00**	**3,00**	**3,00**
Turm	3,02	3,14	3,08	3,26
Dame	4,83	6,04	6,00	5,87

Tabelle 6.3: Vergleich verschiedener Sätze von Materialwerten (Crazyhouse)

[6] Die Werte der Handfiguren entsprechen (normiert) den gelernten. Eine Verrechnung mit Piece-Square-Tables in irgendeiner Form ist nicht erfolgt

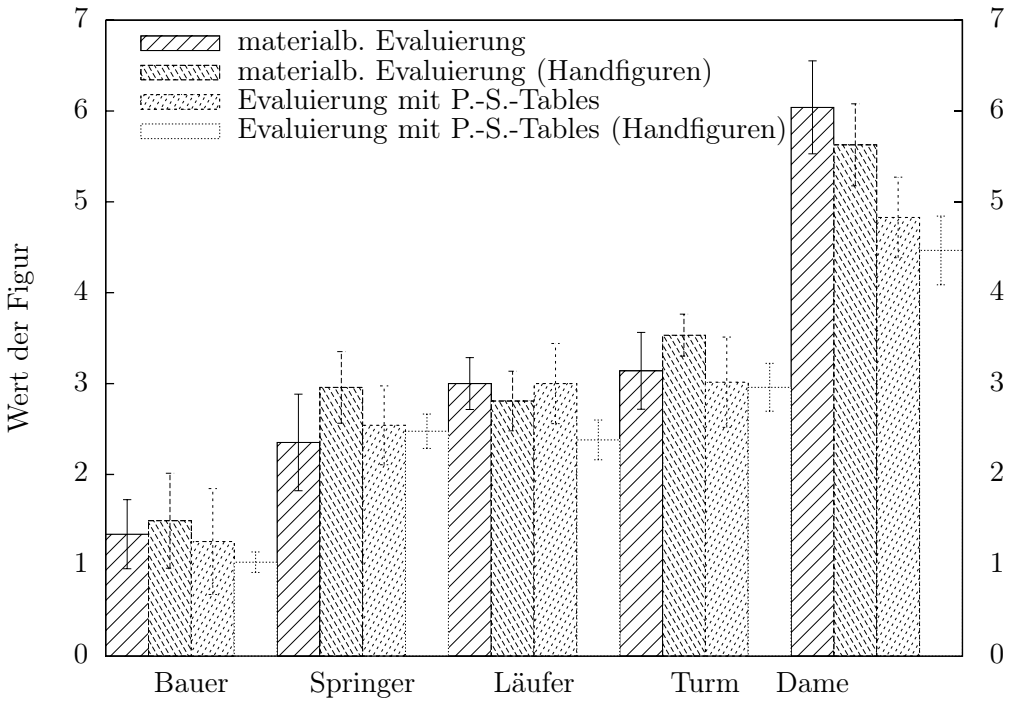


Abbildung 6.7: Finale Materialwerte der einzelnen Figuren

6.2.3 Turnier

Die Ergebnisse des Turniers sind in Tabelle 6.4 angegeben. Die Einträge in der Spalte Konfiguration haben hier folgende Bedeutungen:

gelernt DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den Materialwerten, die im kleinen Experiment gelernt wurden

klassisch DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und klassischen Materialwerten; Figuren auf der Hand haben die gleiche Wertigkeit wie die auf dem Brett.

original DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den in Sunsetter 7e verwendeten Werten

Sjeng (Werte) DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den in Sjeng in der Datei eval.c fest eingestellten Werten

errechnet DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den errechneten Materialwerten aus Tabelle 6.3

piece-square DaSunsetter mit der gleichen Evaluierungsfunktion wie im großen Experiment und den dort durchschnittlich gelernten Werten

Konfiguration	Gewonnen	Verloren	Unentschieden	Quote
gelernt vs. klassisch	1352	646	2	67,65%
gelernt vs. original	1317	680	3	65,93%
gelernt vs. Sjeng (Werte)	1322	674	4	66,20%
gelernt vs. errechnet	813	1184	3	40,73%
piece-square vs. gelernt	1928	71	1	96,43%

Tabelle 6.4: Ergebnisse der Turniere (Crazyhouse)

In den ersten drei Zeilen zeigt eine Gewinnquote von etwa 66% wieder, dass eine leichte Optimierung der Figurenwerte durchaus zu einem stärkeren Spiel führt. Das Symptom aus dem Standardschach, dass das Programm mit gelernten Werten scheinbar schlechter spielt als mit errechneten verstärkt sich hier allerdings auf ein Maß, das nicht mehr mit bloßer Statistik zu erklären ist. Hier wäre wieder zu erwarten gewesen, dass das Programm mit den Werten am besten spielt, die gelernt wurden für die gleiche Evaluierungsfunktion die auch im Turnier verwendet wurde.
Die Gewinnquote des Programmes mit Piece-Square-Tables übertrifft noch die Quote aus der gleichen Spielserie im Standardschach. Dies passt gut zu den vergrößerten Differenzen zwischen den Minimal- und Maximalwerten in Diagramm 6.6, die bereits ein Hinweis darauf waren, dass Piece-Square-Tables sich in Crazyhouse besser als Evaluierungsterm eignen als in Standardschach.

6.3 Räuberschach

6.3.1 Kleines Experiment

Die Tatsache, dass alle Werte in Abbildung 6.8 negativ sind lässt sich einfach dadurch erklären, dass die Evaluierungsfunktion in DaSunsetter nicht unterscheidet zwischen Standardschach und Räuberschach. Da normalerweise eine Materialüberzahl günstig ist für einen Spieler werden die Figuren positiv bewertet. Das führt bei einem Figurenvorsprung zu einer positiven Evaluierung und somit einer hohen errechneten Gewinnwahrscheinlichkeit. In Räuberschach ist es allerdings schlecht viele Figuren bzw. einen Figurenvorsprung zu haben. Daher werden die Figuren negativ bewertet. So ergibt sich aus einem Figuren*rückstand* eine hohe Evaluierung und somit eine hohe errechnete Gewinnwahrscheinlichkeit.

Ein stark negativer Wert, wie der des Turms oder der Dame, ist hier so zu interpretie-

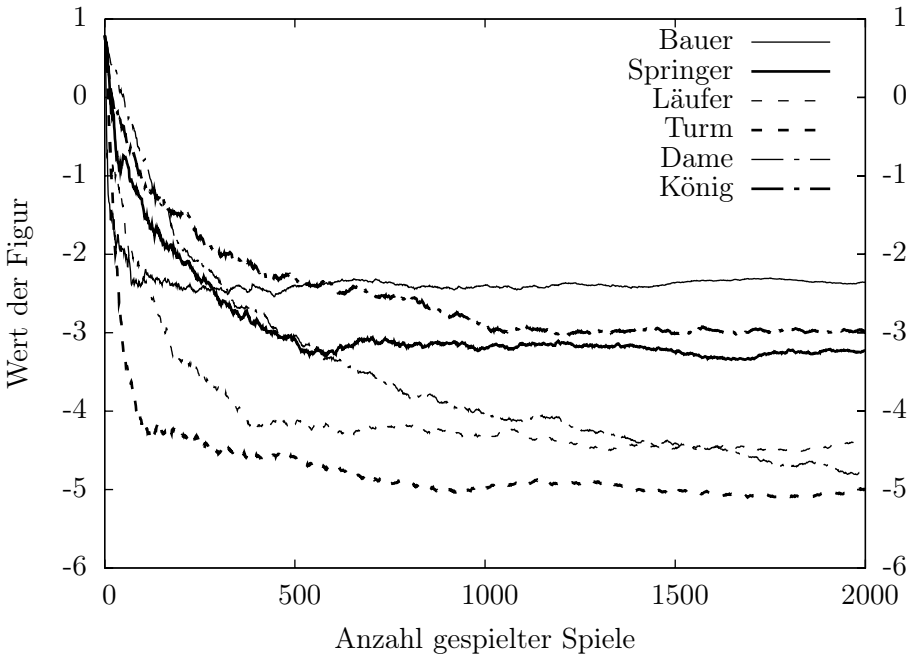


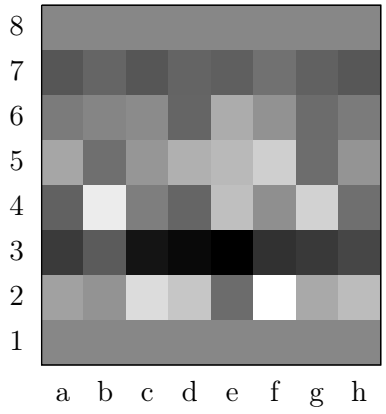
Abbildung 6.8: Entwicklung der Materialwerte (Räuberschach)

ren, dass es besonders schlecht ist von dieser Figur viele bzw. mehr als der Gegner zu haben. Die Werte von Turm und Dame erklären sich beim Räuberschach einfach durch die Fülle an Zugmöglichkeiten. Wer einmal selber Räuberschach spielt stellt schnell fest, dass es sehr schwierig ist, die Dame so zu platzieren, dass sie keine gegnerischen Figuren schlagen kann. Dennoch ist sie nicht ganz so gefährlich wie der Turm, da sie häufig mehrere gegnerische Figuren bedroht und der entsprechende Spieler trotz Schlagzwang noch ein wenig taktieren kann und mit der Dame so schlägt, dass sie anschließend selbst geschlagen wird. Für den gegnerischen Spieler ist es wiederum häufig ein leichtes seine

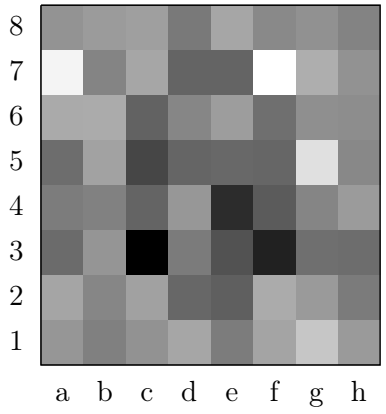
Figuren so zu platzieren, dass der Turm immer genau *eine* Figur schlagen kann. So kann ein Spieler eine häufig spielentscheidende Kette von Schlagzwängen konstruieren.
Der Springer ist kaum gefährlicher als der König, da man ihn leicht seiner Zugmöglichkeiten berauben kann indem man ihn am Rand platziert. Auch der (betragsmäßig) geringe Wert der Bauern ergibt sich aus deren stark eingeschränkten Schlagmöglichkeiten. Selbst wenn ein Bauer mal eine Figur schlägt ist es eher schwierig daraus eine Schlagzwangkette zu konstruieren. Auch die Beförderung macht einen Bauern kaum gefährlicher, da hier als Beförderungsfigur einfach ein König gewählt werden kann.
Die finalen Werte der Figuren sind mit Standardabweichung im Balkendiagramm auf Seite 44 dargestellt. Die Normierung der Figurenwerte für Räuberschach wurde so vorgenommen, dass ein Turm einen Wert von 5 bzw. -5 hat. Der Zweck dieser (von Standardschach unterschiedlichen) Normierung besteht in einer verbesserten Vergleichbarkeit der Werte in Tabelle 6.5.

6.3.2 Großes Experiment

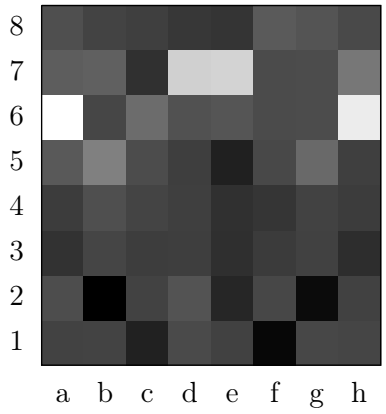
Abbildung 6.9 zeigt die gelernten Piece-Square-Tables für Räuberschach. Im Vergleich zu den Diagrammen der anderen Varianten wirken die Abbildungen 6.9.1 bis 6.9.6 eher chaotisch. Die Diagramme für den Läufer und den Turm sind beide recht symmetrisch aufgebaut. Da es aber isolierte Felder sind, die besonders hell oder besonders dunkel sind, und sich kaum Helligkeitsübergänge zeigen ist die Interpretation dieser Ergebnisse eher müßig; die teilweise recht hohen Differenzen zwischen den minimalen und den maximalen Werten, legt allerdings nahe, dass es sich bei den Helligkeitsunterschieden zwischen den Feldern nicht nur um Rauschen handelt. Ein Blick auf die Standardabweichungen in Abbildung 6.10 und das Phänomen der oben genannten Schlagzwangketten legt für Räuberschach die Erklärung nahe, dass Piece-Square-Tables sich als Merkmale zur Stellungsevaluierung nicht eignen. Ein Merkmal, dass die Möglichkeit des Gegners die eigenen Figuren zu schlagen berücksichtigt, wäre hier vermutlich erheblich effektiver. Die mangelnde Eignung der Piece-Square-Tables wird auch dadurch bestätigt, dass die Standardabweichungen beim großen Experiment deutlich größer sind als beim kleinen. Diese Analyse legt nahe, dass ein Optimieren der Materialwerte kaum zusätzliche Spielstärke bringt solange keine zusätzlichen Merkmale zur Evaluierung herangezogen werden.
Aus den Piece-Square-Tables wurden wieder durchschnittliche Werte der Figuren berechnet. Die errechneten Werte sind zusammen mit den Werten aus dem kleinen Experiment in Tabelle 6.5 aufgeführt. Zum Vergleich enthält die Tabelle auch die Werte, die in Sjeng eingestellt sind. Suicide entspricht der Variation von Räuberschach, die auch DaSunsetter spielt. In Losers hat der König eine Sonderstellung, und ein Spieler hat gewonnen, sobald er matt gesetzt wird. Warum dort dennoch ein Materialwert für den König festgelegt wurde ist nicht ersichtlich. Zu beachten ist auch, dass Sjeng durchaus auf das Spiel von Räuberschach optimiert ist. Der starke Unterschied der Materialwerte zu den hier gelernten erklärt sich vermutlich aus einer völlig anderen, auf Räuberschach optimierten, Evaluierungsfunktion



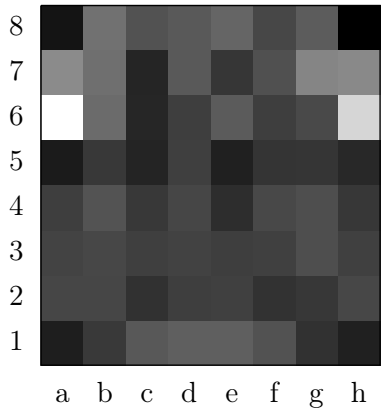
6.9.1: Bauer (Min/Max: -3,76/-2,45)



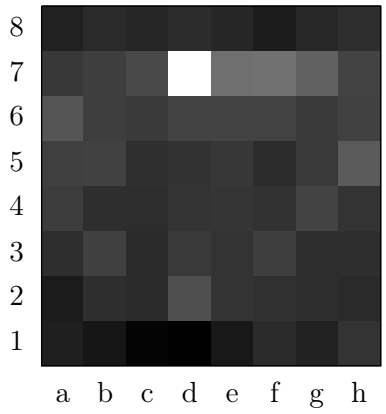
6.9.2: Springer (Min/Max: -4,09/-2,99)



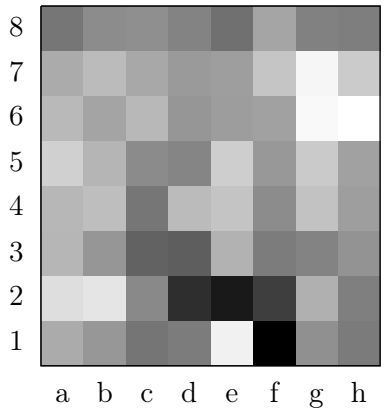
6.9.3: Läufer (Min/Max: -4,13/-1,64)



6.9.4: Turm (Min/Max: -5,43/-3,07)



6.9.5: Dame (Min/Max: -4,20/-1,64)



6.9.6: König (Min/Max: -3,15/-2,66)

Abbildung 6.9: Piece-Square-Tables in Räuberschach

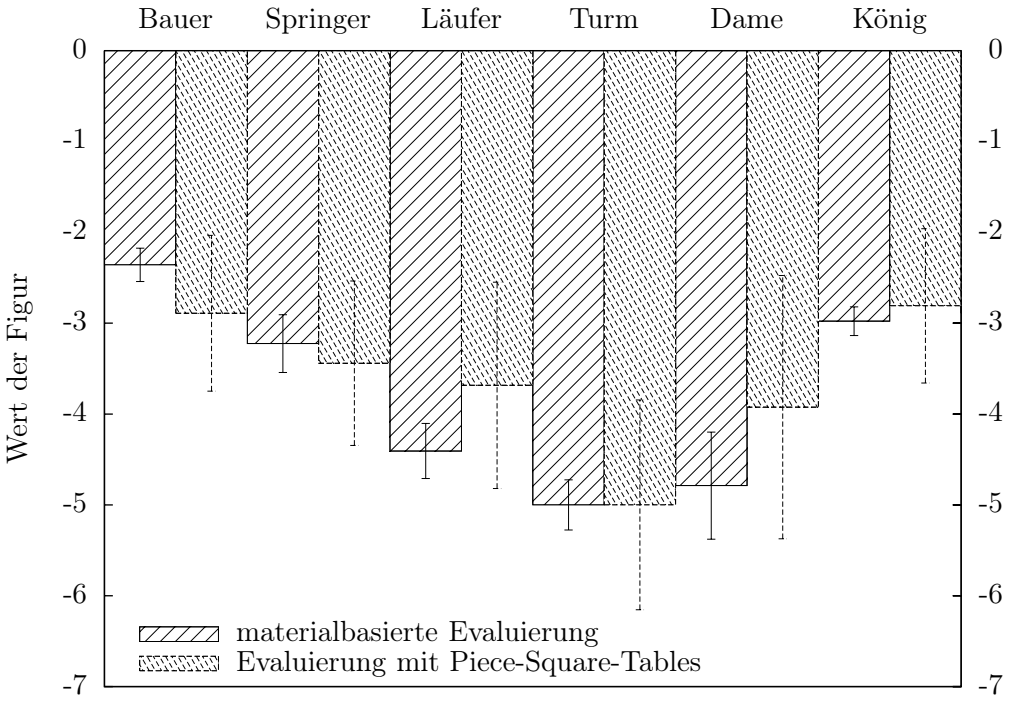


Abbildung 6.10: Finale Materialwerte der einzelnen Figuren

6.3.3 Turnier

Die Ergebnisse des Turniers sind in Tabelle 6.6 angegeben. Die Einträge in der Spalte Konfiguration haben hier folgende Bedeutungen:

gelernt DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den Materialwerten, die im kleinen Experiment gelernt wurden

klassisch DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und invertierten klassischen Materialwerten, d.h. die Werte wurden mit -1 multipliziert. Der König wurde mit -2 bewertet, weil er sich ähnlich bewegt und schlägt wie ein Bauer,

	Großes Exp.	Kleines Exp.	Sjeng (Losers)	Sjeng (Suicide)
Bauer	-2,89	-2,36	1,14	0,50
Springer	-3,44	-3,22	4,57	5,00
Läufer	-3,68	-4,41	3,86	0,00
Turm	**-5,00**	**-5,00**	**5,00**	**5,00**
Dame	-3,92	-4,79	5,71	1,67
König	-2,81	-2,98	14,29	16,67

Tabelle 6.5: Vergleich verschiedener Sätze von Materialwerten (Räuberschach)

aber etwas agiler ist.

Sjeng (Suicide) DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den in Sjeng in der Datei eval.c eingestellten Werten für die Schachvariante Suicide. Zur Anpassung an die in DaSunsetter verwendete Evaluierungsfunktion wurden alle Werte mit -1 multipliziert.

Sjeng (Losers) DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den in Sjeng in der Datei eval.c eingestellten Werten für die Schachvariante Losers. Zur Anpassung an die in DaSunsetter verwendete Evaluierungsfunktion wurden alle Werte wieder mit -1 multipliziert.

errechnet DaSunsetter mit einer rein materialbasierten Evaluierungsfunktion und den errechneten Materialwerten aus Tabelle 6.5

piece-square DaSunsetter mit der gleichen Evaluierungsfunktion wie im großen Experiment und den dort durchschnittlich gelernten Werten

Konfiguration	Gewonnen	Verloren	Unentschieden	Quote
gelernt vs. klassisch	1018	934	48	52,10%
gelernt vs. Sjeng (Suicide)	1790	185	25	90,13%
gelernt vs. Sjeng (Losers)	984	910	106	51,85%
gelernt vs. errechnet	986	948	66	50,95%
piece-square vs. gelernt	1742	243	15	87,48%

Tabelle 6.6: Ergebnisse der Turniere (Räuberschach)

Die einfache Invertierung klassischer Figurenwerte scheint für Räuberschach bei dieser Evaluierung fast genauso gut zu sein wie gelernte Werte. Die Gewinnquote von nur wenig mehr als 50% lässt sich zum Teil durch statistische Effekte "wegerklären".
Das recht schwache Abschneiden der Suicide Werte bestätigt den Verdacht, dass Sjeng zur Evaluierung im Räuberschach (Suicide) eine auf die Variante stark optimierte Evaluierungsfunktion verwendet. Das erheblich bessere Abschneiden der Werte von Losers erklärt sich aus der Ähnlichkeit der Werte zu den gelernten, zeigt aber auch, dass Schwankungen der Figurenwerte in einem nicht ganz unerheblichen Rahmen (siehe Tabelle 6.5) bei dieser Evaluierung das Spielergebnis nur sehr leicht beinflussen.
Die Variante mit Piece-Square-Tables spielt trotz der Beobachtungen im vergangenen Abschnitt deutlich stärker als die ohne Piece-Square-Tables. Die Gewinnquote entspricht mit 87,48% sogar fast genau der Quote von 87,25% aus Tabelle 6.2 im Standardschach. Daraus lässt sich schließen, dass die Muster in Abbildung 6.9 doch nicht so zufällig sind, und die absoluten Positionen der Figuren in Räuberschach ähnlich wichtig sind wie im Standardschach.

6.4 Atomschach

6.4.1 Kleines Experiment

In Atomschach wurden die Werte so normiert, dass ein Springer genau einen Wert von 2,5 aufweist. Die Kurven in Abbildung 6.11 weisen alle etwa ab dem 1000. Spiel kaum noch Veränderungen auf, das Verfahren konvergiert also auch für Atomschach gut, und die durchgeführten 2000 Spiele sind völlig ausreichend.
Auffällig ist in der Abbildung vor allem der extrem hohe Endwert der Dame, der taktisch

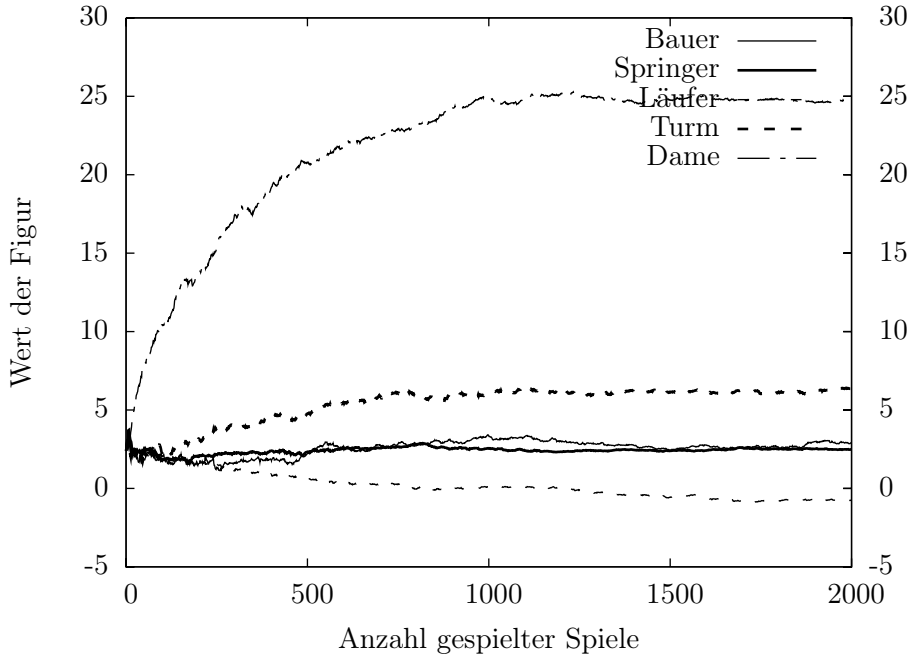
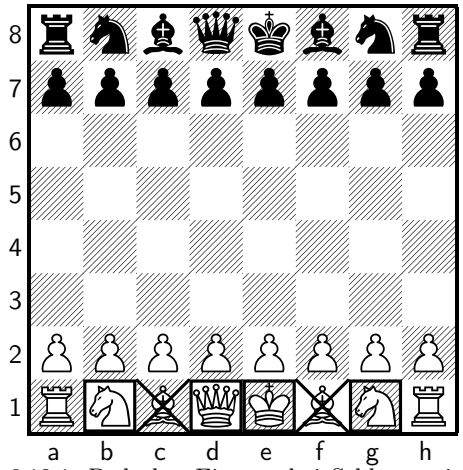


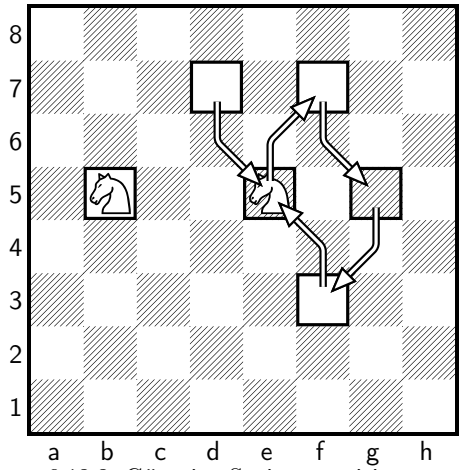
Abbildung 6.11: Entwicklung der Materialwerte (Atomschach)

nur schwer nachvollziehbar ist. Da (aufgrund der Explosion) jede Figur pro Spiel nur einmal schlagen kann liegt die Vermutung nahe, eine Figur müsse bei einem geschickten Zug mindestens so viele gegnerische Figuren schlagen, dass deren Wert den Verlust der eigenen Figur(en) aufwiegt. Bei einem derartig hohen Wert der Dame in Relation zu den übrigen Figuren ist das allerdings kaum zu schaffen. Selbst wenn die Dame (≈25 Punkte) dem Gegner beide Türme (≈12 Punkte), beide Springer (5 Punkte), und einen Bauern[7] (≈2,5 Punkte) nehmen würde, läge der größere Verlust immernoch beim schlagenden Spieler. Der Wert der Dame ergibt sich daher vermutlich eher aus ihrem Bedrohungspotenzial bezüglich des Königs. Durch ihre hohe Beweglichkeit und die Möglichkeit des indirekten Schachs kann sie in manchen Situationen den König über zwei Züge gleichzeitig bedrohen und somit auch (leichter als im Standardschach) den König alleine Matt setzen.
Bei etwas genauerem Hinsehen fällt auch der leicht negative Wert des Läufers auf, der

[7]Da Bauern von der Explosion nicht betroffen sind, kann höchstens einer geschlagen werden.



6.12.1: Bedrohte Figuren bei Schlagen eines Läufers



6.12.2: Günstige Springerpositionen

Abbildung 6.12: Darstellung Anhand der Figuren auf dem Schachbrett

evtl. erklärt werden kann über seine ungünstige Anfangsposition. Eine Bedrohung des Läufers auf seinem Ursprungsfeld gefärdet üblicherweise auch die Dame oder den König (siehe Abbildung 6.12.1).

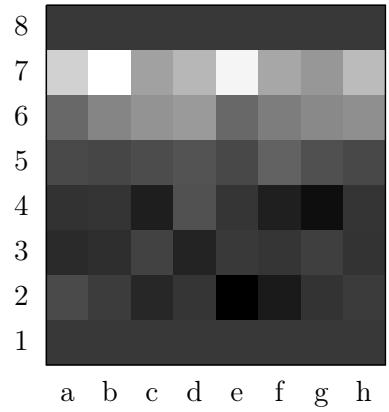
6.4.2 Großes Experiment

Abbildung 6.13.1 sieht ihrer Entsprechung im Standardschach recht ähnlich: Es ist ein Übergang von Dunkel zu Hell von Unten nach Oben zu sehen und vermutlich besteht auch hier die Erklärung in der Beförderung der Bauern bei Erreichen des achten Ranges. Das Diagramm für den Springer wirkt zunächst eher zufällig, offenbart bei genauem Hinsehen jedoch einige Muster. Viele der etwas helleren Stellen liegen gerade einen Springerzug auseinander (siehe Abbildung 6.12.2). Besonders ungünstig sind die Ausgangspositionen des Springers. Das ist nachvollziehbar, da hier eine ähnliche Bedrohung enstehen kann wie die im vergangenen Abschnitt in Abbildung 6.12.1 dargestellte.

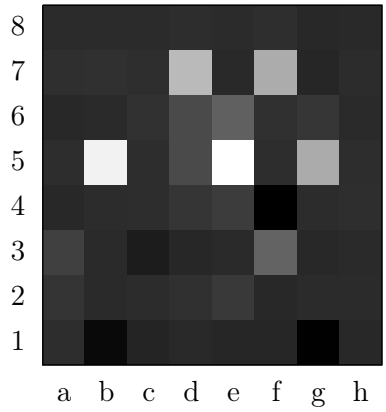
In diesem Kontext erscheint das Diagramm des Läufers verwunderlich. Die Ausgangspositionen (C1 und F1) haben keine erkennbare negative Ausprägung. Das ganze Bild wirkt eher zufällig und mit der geringen Differenz zwischen dem minimalen (2,07) und dem maximalen (3,33) Wert lässt dies den Schluss zu, dass in dem Bild zwischen Muster und Rauschen kein klarer Unterschied gemacht werden kann.

Im Gegensatz dazu steht das Diagramm für den Turm. Das Feld E8 ist klar die beste Position und die umliegenden Felder werden mit steigender Distanz langsam dunkler. In stark abgeschwächter Form findet sich dies auf E1 wieder, von wo aus der Turm mit einem einzigen Zug nach E8 ziehen kann.

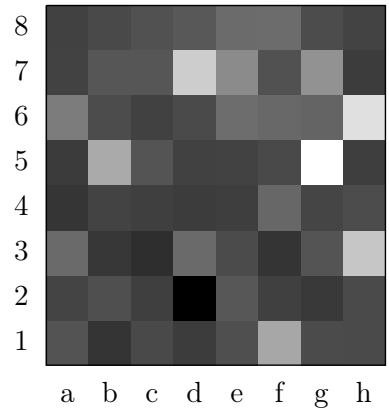
Das Diagramm für die Dame zeigt eine etwas hellere und eine dunklere Zone. Die hellere Zone ergibt sich aus den Bedrohungsmöglichkeiten auf den gegnersichen König, die dunklere Zone aus der Gefährdung der eigenen Figuren im Falle einer Explosion (wenn die



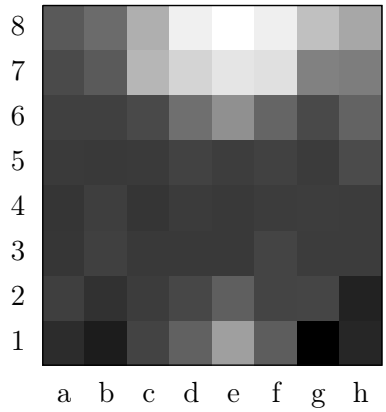
6.13.1: Bauer (Min/Max: 0,54/6,81)



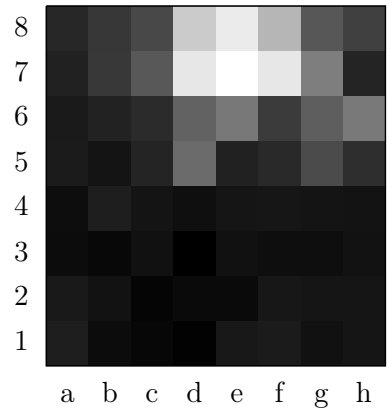
6.13.2: Springer (Min/Max: 1,74/5,67)



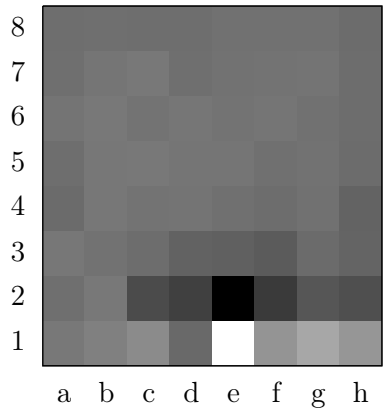
6.13.3: Läufer (Min/Max: 2,07/3,33)



6.13.4: Turm (Min/Max: 4,48/7,73)



6.13.5: Dame (Min/Max: 13,25/22,90)



6.13.6: König (Min/Max: -2,5/5,54)

Abbildung 6.13: Piece-Square-Tables in Atomschach

Dame selbst geschlagen wird). Selbst in diesem dunklen Bereich ist die Dame allerdings immernoch mehr wert als jede andere Figur.
Das gleichmäßige Grau im oberen Bereich von Abbildung 6.13.6 ist sehr wahrscheinlich dadurch entstanden, dass der König diese Felder fast nie betreten hat und der Algorithmus deshalb die Werte nur schlecht bzw. garnicht lernen konnte. Etwas merkwürdig wirkt die direkte Nachbarschaft des weißen und des schwarzen Feldes, der große Unterschied zwischen ihren Werten zeigt aber, dass es irgendeinen taktischen Grund dafür geben muss. Eventuell ist diese Konstellation aber auch über einen anderen Effekt zu erklären, der sich offenbart wenn man Atomschach einmal selber gespielt hat. Gewinnt man ein Spiel, so bewegt man den König üblicherweise garnicht. So kommt im Experiment ein hoher Wert für das Feld E1 zustande. Läuft das Spiel wiederum so schlecht, dass man anfangen muss mit dem König zu fliehen, verliert man im Normalfall. Dadurch entstehen niedrige bzw. negative Werte für die häufigsten Fluchtfelder D2, E2, F2.
In Abbildung 6.14 finden sich wieder errechnete Durchschnittsmaterialwerte mit Standardabweichung aus dem großen Experiment und gelernte Materialwerte aus dem kleinen Experiment. Auf die Vergleichstabelle wird in diesem Abschnitt verzichtet, weil für Atom-

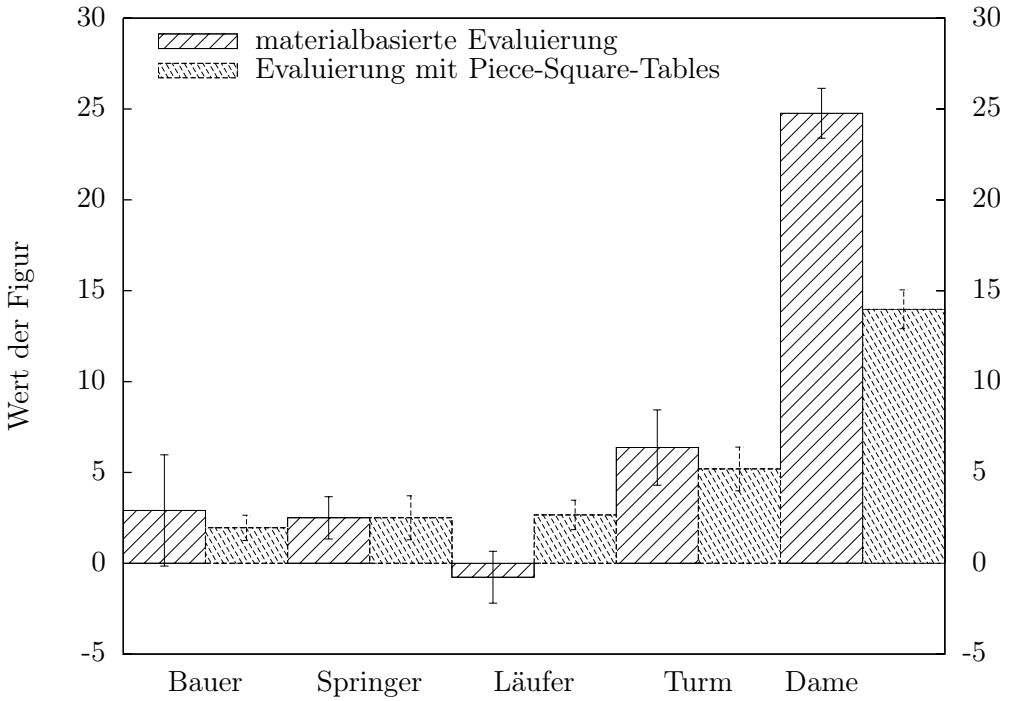


Abbildung 6.14: Finale normierte Materialwerte der einzelnen Figuren (Atomschach)

schach keine Vergleichswerte zu finden waren. Der Vollständigkeit halber findet sich aber Tabelle B.1 mit den finalen Werten aus dem Balkendiagramm 6.14 im Anhang.

6.4.3 Turnier

Die Ergebnisse des Turniers in Tabelle 6.7 bestätigen, was die vergangenen zwei Abschnitte bereits erahnen lassen. Die gelernten Werte führen mit einer Gewinnquote von 61,95% eindeutig zu einem stärkeren Spiel und im Rückschluss auch zu einer besseren Evaluierungsfunktion bzw. einer besseren Abschätzung der Gewinnchancen in einer gegebenen Situation. Das sehr gute Abschneiden der Variante mit Piece-Square-Tables (vergleiche

Konfiguration	Gewonnen	Verloren	Unentschieden	Quote
gelernt vs. klassische	1231	753	16	61,95%
gelernt vs. errechnet	1186	797	17	59,73%
piece-square vs. gelernt	1924	75	1	96,23%

Tabelle 6.7: Ergebnisse der Turniere (Atomschach)

hierzu auch Tabelle 6.2) bestätigt die relativ hohe Bedeutung der absoluten Positionen der Figuren auf dem Spielbrett.

7 Zusammenfassung

Nachdem in der Einleitung die Aufgabe beschrieben wurde haben die Kapitel 2 bis 4 eine Einführung gegeben in das Grundwissen, das zum Verständnis dieser Arbeit nötig ist. Es wurden die Spielregeln für Standardschach und die hier betrachteten drei Varianten erklärt sowie die Funktionsweise eines Schachprogrammes. Dazu gehörte auch eine Darstellung des Alpha-Beta-Verfahrens (Seite 8), mit dessen Hilfe der Spielbaum durchsucht wird und welches somit einen zentralen Bestandteil des künstlichen Spielers darstellt. In diesem Rahmen wurde auch die Rolle der Evaluierungsfunktion beschrieben, die in den Experimenten dieser Arbeit gelernt wurde.
Anschließend wurde das Verfahren TD(λ) erklärt, unter dessen Einsatz diese Funktion bzw. Gewichte in dieser Funktion trainiert wurden. Kapitel 3 enthielt eine Beschreibung des Ablaufs von TD(λ) und der dabei verwendeten Gleichungen. Auch die Stauchungsfunktion (Seite 12) wurde hier vorgestellt, die die Ausgabe der Evaluierungsfunktion in einen Wertebereich abbildet, mit dem TD(λ) arbeiten kann.
Kapitel 4 hat einen Überblick gegeben über wichtige Arbeiten zur Thematik dieser Arbeit und damit die Themen aus den Kapiteln 2 und 3 zusammengeführt. Zum überwiegenden Teil handelte es sich dabei um Arbeiten von Donald F. Beal und Martin C. Smith, die sich bereits früher (1997) damit beschäftigt haben, wie Temporal Difference Learning verwendet werden kann, um die Gewichte einer Evaluierungsfunktion für Standardschach zu lernen.
In Kapitel 5 wurde die Herangehensweise an die durchzuführenden Experimente dargestellt. Dazu gehörte eine Beschreibung zweier Experimente (klein und groß; Seite 23), die für jede Variante durchgeführt wurden, sowie eine Vorstellung der verwendeten Programme. Zum einen war das das verwendete Schachprogramm DaSunsetter und zum anderen die Experimentierumgebung, die die einzelnen Spiele gestartet hat.
Die Resultate der Experimente wurden in Kapitel 6 aufgeführt, zusammen mit Interpretationen einzelner Ergebnisse. Der Stil der Diagramme in diesem Kapitel war stark an die Diagramme in den Arbeiten von Beal und Smith angelehnt. So wurde zusammen mit den Experimenten im Standardschach eine Vergleichbarkeit der Experimente *dieser* Arbeit mit denen in [BS97] und [BS99a] hergestellt. Zu den besonders interessanten Ergebnissen gehört die Feststellung im Crazyhouse-Abschnitt (Seite 35), dass bereits durch eine relativ kleine Regeländerung neue Merkmale zur Evaluierungsfunktion und damit neue zu lernende Gewichte hinzukommen können. Wenn die neuen Merkmale seltener eine Ausprägung ungleich 0 haben als die bisher bekannten (was leicht mal übersehen werden kann), dann hat dies auch einen entsprechenden Einfluss auf die Konvergenz des Verfahrens, bzw. auf die Anzahl von Spielen, die notwendig sind um alle Werte korrekt zu lernen.
In den Experimenten mit Räuberschach wurde deutlich, dass die gelernten Werte auch

bei anschaulicher Präsentation nicht unbedingt ein für den Menschen erkenn- oder erklärbares Muster aufweisen müssen (siehe Abbildung 6.9). Dennoch erweisen sich die gelernten Werte als sinnvoll und legen damit den Schluss nahe, dass in den Werten eben doch ein System verborgen liegt (siehe Tabelle 6.6 letzte Zeile).
Weiterhin zeigte sich das Phänomen, dass manche Ergebnisse sich eher aus dem Ablauf des Lernverfahrens heraus erklären lassen als über die tatsächlichen Werte einer Figur oder Position, wie es der Fall war mit der Piece-Square-Table des Königs in Atomschach (siehe Abschnitt 6.4.2). Ein weiteres Beispiel eines etwas kontraintuitiven Ergebnisses ist der Materialwert der Dame in Abschnitt 6.4.1. Die naive Annahme, der Wert einer Figur entspräche in etwa dem summierten Wert aller Figuren die sie schlägt, erwies sich dort als völlig falsch.
So haben sich aus den gemachten Versuchen mit Schachvarianten tatsächlich Aspekte ergeben, die im Standardschach so nicht aufgetreten oder zumindest nicht aufgefallen sind. Es hat sich auch gezeigt, dass das Prinzip der Piece-Square-Tables nicht bei allen Varianten gleich gut funktioniert (Vergleiche dazu die Einträge Piece-Square vs. gelernt in den Tabellen 6.2, 6.4, 6.6 und 6.7), was wiederum Rückschlüsse auf die Natur des Spiels ermöglicht. Die absolute Position einer Figur scheint beispielsweise in Atomschach eine größere Rolle zu spielen als in Räuberschach.
Für künftige Beschäftigungen mit dem Themenbereich dieser Arbeit böten sich Evaluierungsfunktionen, deren Gestalt für eine ganz bestimmte Schachvariante optimiert ist. Beispielsweise wurde am Rande dieser Arbeit ein kleiner Versuch durchgeführt, in dem für Räuberschach eine Evaluierungsfunktion mit insgesamt 14 Gewichten trainiert. Die Idee hinter dieser Evaluierung bestand in einer sehr einfachen Erkennung von Situationen, die das Potential von Schlagzwangketten enthalten. Das Programm mit dieser Spezialevaluierung gewann gegen eines mit gelernten Piece-Square-Tables von 2000 Spielen knapp 90%. Wurden Gewichte für die gleiche Evaluierung für Standardschach gelernt, so gewann gewann das Programm (in Standardschach) gegen eine Evaluierung mit gelernten Piece-Square-Tables kaum mehr als 1% der 2000 Turnierspiele. Eventuell könnte auch ein System nach dem Prinzip von Morph ([LHHS$^+$91]) dabei helfen, neue Evaluierungsmerkmale für einzelne Schachvarianten zu finden.

A Das XBoard Protokoll

Im Bereich der Schachsoftware werden die grafische Oberfläche und das eigentliche Schachprogramm häufig als zwei getrennte Programme entwickelt. Die grafische Oberfläche (z.B. XBoard oder WinBoard) startet ein (vom Benutzer bestimmbares) Schachprogramm und kommuniziert dann mit diesem über ein vorher definiertes Protokoll. Ein wichtiges Protokoll, das von vielen Programmen verwendet wird heißt XBoard (genau wie die dazugehörige grafische Oberfläche). In XBoard sind Befehle definiert, über die das Schachprogramm mit der Oberfläche Informationen über gemachte Züge und ähnliches austauschen kann. Die Beschreibung des Protokolls findet sich im Internet unter `http://www.tim-mann.org/xboard/engine-intf.html`. Im folgenden ist eine kurze Beispielsitzung (mit einem etwas älteren DaSunsetter) gegeben[1]:

```
xboard                                          # verwende XBoard-Protokoll
protover 2                                      # ...Version 2
-> feature setboard=1                           # setboard Feature aktiv
-> feature analyze=1
-> feature ping=1
-> feature ics=1
-> feature variants="normal,crazyhouse,suicide" # beherrschte Varianten
-> feature myname="Sunsetter C10sd"
-> feature done=1
new                                             # starte neues Spiel
sd 5                                            # Suchtiefe beträgt 5 Halbzüge
variant suicide                                 # spiele Räuberschach
d2d4                                            # Weißer Zug von d2 nach d4
->
-> Found move: b7b5 +0 fply: 4  searches: 776
-> Time      : Time Alloc: -2147483648 Clock Ticks Used (in Thousands): 0
->
-> move b7b5                                     # Schwarzer Zug von b7 nach b5
[Weiterer Austausch von Zügen]
-> 1-0 {White mates}                            # Weiß hat gewonnen
quit                                            # Beende das Programm
```

[1] Zeilen, die mit einem "->" beginnen, stellen Antworten von DaSunsetter dar. "#" leitet einen Kommentar ein.

B Daten

In diesem Anhang werden der Vollständigkeit halber Daten aufgelistet, die in keinem der vorigen Abschnitte enthalten sind.

	Großes Experiment	Kleines Experiment
Bauer	1,95	2,91
Springer	**2,50**	**2,50**
Läufer	2,66	-0,77
Turm	5,19	6,37
Dame	13,98	24,77

Tabelle B.1: Berechnete und direkt gelernte Materialwerte in Atomschach

	Standard-schach	Crazyhouse (Feld)	Crazyhouse (Hand)	Räuberschach	Atomschach
Bauer	1,919276	0,695343	1,646323	-5,028221	1,332826
Springer	6,782095	2,696487	3,942069	-5,549156	1,947714
Läufer	7,278907	3,462090	3,789443	-5,482632	1,969341
Turm	11,893306	3,519030	4,712312	-7,211810	5,391944
Dame	25,256481	6,498132	7,111896	-5,716622	16,196686
König	k.A.	k.A.	k.A.	-4,770337	k.A.

Tabelle B.2: Unnormierte durchschnittlich gelernte Materialwerte im kleinen Experiment

	Standard-schach	Crazyhouse (Feld)	Crazyhouse (Hand)	Räuberschach	Atomschach
Bauer	1,906929	1,652256	1,490270	-2,978669	1,211811
Springer	2,573399	2,895256	3,641500	-4,074316	1,042676
Läufer	3,395056	3,693724	3,456344	-5,567883	-0,320755
Turm	4,951009	3,866528	4,349853	-6,317475	2,655727
Dame	12,416373	7,437093	6,929007	-6,049365	10,329872
König	k.A.	k.A.	k.A.	-3,762665	k.A.

Tabelle B.3: Unnormierte durchschnittlich gelernte Materialwerte im großen Experiment

	Standard-schach	Crazyhouse (Feld)	Crazyhouse (Hand)	Räuberschach	Atomschach
Bauer	0,006481	0,013479	0,021709	0,008147	0,023706
Springer	0,011386	0,030805	0,040058	0,023301	0,011942
Läufer	0,010038	0,028439	0,041628	0,026534	0,028890
Turm	0,015885	0,030503	0,052202	0,024403	0,039013
Dame	0,029663	0,051864	0,125860	0,058598	0,081926
König	k.A.	k.A.	k.A.	0,040419	k.A.

Tabelle B.4: Durchschnittliche finale Alphas im kleinen Experiment

	Standard-schach	Crazyhouse (Feld)	Crazyhouse (Hand)	Räuberschach	Atomschach
Bauer	0,004996	0,003810	0,004901	0,004151	0,004982
Springer	0,011598	0,003807	0,005434	0,007125	0,009025
Läufer	0,010936	0,005608	0,006218	0,006755	0,011253
Turm	0,018030	0,006223	0,008439	0,007662	0,016296
Dame	0,058367	0,007358	0,013445	0,010144	0,034585
König	k.A.	k.A.	k.A.	0,007662	k.A.

Tabelle B.5: Durchschnittliche finale Alphas (der Figurengewichte) im großen Experiment

Literaturverzeichnis

[BM99] Yngvi Björnsson and Tony Marsland. Multi-cut pruning in alpha-beta search. In H. J. van den Herik and H. Iida, editors, *Proceedings of the First International Conference on Computers and Games (CG-98)*, volume 1558 of *Lecture Notes in Computer Science*, pages 15–24. Springer-Verlag, 1999.

[BS97] D.F. Beal and M.C. Smith. Learning piece values using temporal differences. *International Computer Chess Association Journal*, 20(3):147–151, 1997.

[BS99a] D.F. Beal and M.C. Smith. Learning piece-square values using temporal differences. *International Computer Chess Association Journal*, 22(4):223–235, December 1999.

[BS99b] D.F. Beal and M.C. Smith. Temporal coherence and prediction decay in td learning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 564–569. Morgan Kaufmann, 1999.

[BS99c] Donald F. Beal and Martin C. Smith. First results from using temporal difference learning in Shogi. In H. J. van den Herik and H. Iida, editors, *Proceedings of the First International Conference on Computers and Games (CG-98)*, volume 1558 of *Lecture Notes in Computer Science*, pages 113–125. Springer-Verlag, 1999.

[BU] Dennis M. Breuker and Jos W.H.M. Uiterwijk. Transposition tables in computer chess.

[Cla73] M.R.B. Clarke. Some ideas for a chess compiler. In Alick Elithon and David Jones, editors, *Proceedings of a Nato Symposium entitled Human Thinking: Computer Techniques for its Evaluation*, pages 189–192. Elsevier Scientific Publishing Company, 1973.

[dE] Fédération Internationale des Échecs. Laws of chess. http://www.fide.com/official/handbook/pdf/EE101.pdf.

[JB98] Lex Weaver Jonathan Baxter, Andrew Tridgell. Tdleaf(lambda): Combining temporal difference learning with game-tree search. *Australian Journal of Intelligent Information Processing Systems*, 5(1):39–43, 1998.

[LHHS+91] Robert Levinson, Feng Hsiung-Hsu, Jonathan Schaeffer, T. Anthony Marsland, and David E. Wilkins. The current and future role of chess in artificial intelligence an machine learning. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 547–552, 1991.

[Lug01] George F. Luger. *Künstliche Intelligenz - Strategien zur Lösung komplexer Probleme (4. Auflage)*, chapter 9.7 Reinforcement-Lernen. Pearson Studium, 2001.

[Mit97] Tom M. Mitchell. *Machine Learning*, chapter 13.3 Q Learning, pages 373–380. The McGraw-Hill Companies, Inc., international edition, 1997.

[Sch86] Peter Schefe. *Künstliche Intelligenz - Überblick und Grundlagen*, volume 53 of *Reihe Informatik*, chapter 2.5.2 Durchsuchen von Spielbäumen, pages 96–101. Bibliographisches Institut Wissenschaftsverlag Mannheim/Wien/Zürich, 1986.

[Sut88] Richard S. Sutton. Learning to predict by the methods of temporal differences. In *Machine Learning*, volume 3, pages 9–44, 40 Sylvan Road, Waltham, MA 02254, U.S.A., 1988. GTE Laboratories Incorporated, Kluwer Academic Publishers, Boston.

[TN02] Omid David Tabibi and Nathan S. Netanyahu. Verified null-move pruning. Technical Report CAR-TR-980, CS-TR-4406, UMIACS-TR-2002-39, Center for Automation Research, University of Maryland, 2002. Published in ICGA Journal, Vol. 25, No. 3, pp. 153–161.