Techniken für die Bewertung von Spielsituationen anhand von Beispielen

Vom Fachbereich 17 – Mathematik / Informatik –
der Universität–GH–Paderborn
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
genehmigte Dissertation

von
Michael Buro

Techniken für die Bewertung von Spielsituationen anhand von Beispielen

Vom Fachbereich 17 – Mathematik / Informatik –
der Universität–GH–Paderborn
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
genehmigte Dissertation

von
Michael Buro

Referent:	Prof. Dr. Hans Kleine Büning
Korreferenten:	Priv. Doz. Dr. Ingo Althöfer
	Prof. Dr. Burkhard Monien

Tag der mündlichen Prüfung: 2. Dezember 1994

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Angestellter im Fachgebiet Praktische Informatik (Wissensbasierte Systeme) der Universität–GH–Paderborn. Hier wurde mir von Prof. Dr. Hans Kleine Büning die Gelegenheit gegeben, Aspekte eines faszinierenden Bereichs der Künstlichen Intelligenz — der Spielbaumsuche — zu untersuchen.

Es gibt zwei Personen, die wesentlich zum Entstehen dieser Arbeit beigetragen haben und denen ich zu tiefem Dank verpflichtet bin. Ohne die Bereitschaft meiner Ehefrau Karen, auf die von einem Informatiker zuweilen mit einer quälenden Frequenz gestellten Fragen statistischen Inhalts geduldig und kompetent einzugehen, hätten manche der vorgestellten Verfahren nur schwerlich verwirklicht werden können. Mit seinem starken Othello–Programm übte mein Kollege Igor Đurđanović stets Druck auf mich aus. Dieses mußte notgedrungen nach unzähligen Diskussionen zu wesentlichen Verbesserungen unserer Programme führen.

Darüberhinaus gilt mein Dank Dr. Rainer Feldmann und Dr. Peter Mysliwietz, die eine frühe Version dieser Arbeit kritisch durchsahen. Außerdem danke ich Priv. Doz. Dr. Ingo Alhöfer und Prof. Dr. Burkhard Monien für die Übernahme der Korreferate.

Paderborn, im Dezember 1994

– M.B.

Inhalt

1 Einführung **1**

2 Konstruktion von Merkmalen **5**
2.1 Aktuelle Mobilität . . . 7
2.2 Potentielle Mobilität . . . 8
2.3 Muster . . . 10
2.4 Diskussion . . . 14

3 Gewichtung der Merkmale **21**
3.1 Diskriminanzfunktionen für normalverteilte Merkmale . . . 24
3.2 Logistische Regression . . . 26
3.3 Wahl der Beispiele . . . 28
3.4 Vergleich . . . 30
3.5 Diskussion . . . 39
3.6 Theoretische Details . . . 41

4 Selektive Suche **45**
4.1 Selektive heuristische Suche . . . 45
4.2 Selektive Entscheidungssuche . . . 51
4.3 Diskussion . . . 55

5 Strategien für eine Partienfolge **57**
5.1 Strategien . . . 57
5.2 Realisation . . . 59
5.3 Diskussion . . . 63

6 Schlußbemerkungen **64**

Literatur 67

A Othello 71

B LOGISTELLO 73

C Startpositionen 74

D Endspielpositionen 78

E Turnierergebnisse 83

Kapitel 1

Einführung

Im Bereich der Künstlichen Intelligenz sind heuristische Suchverfahren von großer Bedeutung. Sie werden beispielsweise in Robotersteuerungen, Expertensystemen und Spielprogrammen eingesetzt. Dabei werden üblicherweise große Räume gesteuert von heuristischen Funktionen nach Lösungen des vorgelegten Problems durchsucht. Oft haben die Suchräume die Struktur eines Baums: Ausgehend von einer Startsituation entstehen durch Handlungsalternativen neue Situationen, in denen u.U. wiederum verschiedene Handlungen möglich sind. Mit der Suche in speziellen Bäumen solcher Art — den Spielbäumen — beschäftigt sich diese Arbeit.

C.E. Shannon schlug schon 1950 vor, daß Computer Spiele wie beispielsweise Schach oder Dame derart spielen könnten, daß in der zur Verfügung stehenden Zeit der Spielbaum so tief wie möglich untersucht wird, die Blattstellungen im Hinblick auf die Gewinnchancen heuristisch bewertet werden, diese Werte gemäß der MiniMax–Strategie zur Wurzel propagiert werden und schließlich der Zug gewählt wird, der zu einer Stellung mit dem höchsten Wert führt. Diese Idee ist bis heute aktuell, und fast alle Programme im Bereich der Zweipersonen–Nullsummenspiele mit perfekter Information[1] benutzen eine Verbesserung dieses Ansatzes — den $\alpha\beta$–Algorithmus, der schon Brudno (1963) bekannt war und von Knuth & Moore (1975) erstmals eingehend analysiert wurde. Hierbei wird die

[1] Hierbei spielen zwei Spieler, denen zu jedem Zeitpunkt die aktuellen Regeln des Spiels und die Spielsituation vollständig bekannt sind. Darüberhinaus addieren sich beide Gewinnauszahlungen zu Null. Beispiele hierfür sind Othello, Dame, Schach und Go, aber nicht Skat oder Backgammon.

Eigenschaft ausgenutzt, daß zur Bestimmung des MiniMax–Wertes der Wurzelstellung nicht unbedingt alle Blätter besucht werden müssen. Wird in jedem Knoten zuerst der beste Nachfolger untersucht, so benötigt der $\alpha\beta$–Algorithmus nur ungefähr das Doppelte der Quadratwurzel des ursprünglichen Aufwands und kann somit bei gleicher Zeit eine ca. doppelt so große Suchtiefe erreichen.

Um in der Praxis in den Genuß dieser Effizienzsteigerung zu kommen, wurden im Laufe der Zeit Techniken entwickelt, die es ohne großen Zeitaufwand gestatten, den vermeintlich besten Zug in einer Stellung zuerst zu untersuchen. Diese umfassen u.a. das Sortieren der Züge mittels flacher Baumsuchen, die Benutzung sogenannter „Response–Killer"–Listen, bei denen sich zu jedem gegnerischen Zug ein Folge von guten Erwiderungen gemerkt wird und das Aufbewahren von bisher besten Zügen in einer großen Hashtabelle. Die letztgenannte Methode wird in Verbindung mit einer iterativen Vertiefung eingesetzt, bei der der Spielbaum nicht direkt bis zur maximalen Tiefe untersucht wird, sondern die Tiefe schrittweise erhöht wird [vgl. Slate & Atkin (1983)]. Dieses Vorgehen mutet zunächst wie eine Zeitverschwendung an, weil hierbei Knoten mehrmals besucht werden. Tatsächlich zeigt sich aber, daß die aus früheren Iterationen in der Hashtabelle und den Killer–Listen gesammelten Informationen eine gute Zugvorsortierung ermöglichen und damit zu einer insgesamt deutlichen Verringerung der Suchzeit führen. Durch die Verwendung sogenannter Nullfenster–Suchverfahren, bei denen das vom $\alpha\beta$–Algorithmus verwaltete Suchfenster konsequent klein gehalten wird, um möglichst oft Äste im Suchbaum abschneiden zu können, kann die Effizienz noch gesteigert werden. Eine umfassende Darstellung dieses Themenkomplexes gibt Reinefeld (1989). Dort werden auch alternative Verfahren zur Bestimmung des MiniMax–Wertes diskutiert, die zwar im günstigen Fall weniger Blätter als die $\alpha\beta$–Varianten besuchen, aber wegen ihres großen Verwaltungsaufwands und in der Suchtiefe exponentiellen Speicherplatzbedarfs bisher noch keine breite Anwendung fanden. Weitere Verbesserungsmöglichkeiten der $\alpha\beta$–Suche liegen in der selektiven Vertiefung relevanter Zugfolgen zur ökonomischeren Nutzung der Bedenkzeit oder der Parallelisierung zur tieferen Erkundung des Spielbaums bei gleicher Zeitvorgabe, wie beispielsweise Anantharaman et al. (1990) bzw. Feldmann (1993) zeigen.

Bevor die Spielwerte gemäß der MiniMax–Strategie propagiert werden können, müssen sie als Maß für die Gewinnaussichten eines Spielers an den zu untersuchenden Blattstellungen des Spielbaums bestimmt werden. Dieses geschieht durch eine heuristische Bewertungsfunktion, die im Normalfall aus der Verknüpfung mehrerer Funktionen entsteht, welche ihrerseits verschiedene Aspekte der Stellung hinsichtlich deren Beitrags zum Gewinn bewerten. Da sich an eine derartige Bewertung keine Baumsuche mehr anschließt, muß sichergestellt werden, daß die (statische) Bewertungsfunktion nicht auf unruhige, taktische Stellungen angewandt wird. Dieses kann im Rahmen einer selektiven Ruhesuche geschehen, bei der nur jeweils ein Teil aller möglichen Züge untersucht wird und die erst bei Erreichen ruhiger Stellungen endet. Wie in der Praxis beobachtet werden kann, wächst im Normalfall die Spielstärke eines Programms mit dessen Suchtiefe. Daher muß bei der Konstruktion einer Bewertungsfunktion ein Mittelweg gefunden werden: Einerseits soll sie eine hohe Güte besitzen, also mit hoher Sicherheit gewonnene von verlorenen Positionen trennen können, was oft zeitaufwendige Berechnungen zu erfordern scheint. Andererseits erzielt u.U. eine schneller auszuwertende Bewertungsfunktion geringerer Güte in Verbindung mit einer tieferen Suche eine gößere Spielstärke.

Trotz vielversprechender Arbeiten, die alternative Bewertungs-, Propagierungs- und auch Suchtechniken vorschlugen, werden in der Praxis vorwiegend Varianten des $\alpha\beta$–Algorithmus' aufgrund deren Zeit– und Platz–Effizienz und den von ihnen erreichten großen Suchtiefen benutzt.

In der vorliegenden Arbeit werden Techniken vorgestellt, mit deren Hilfe die Güte der Bewertung von Spielsituationen im Vergleich zu herkömmlichen Ansätzen gesteigert werden kann, und die ausnahmslos auf der Untersuchung von Beispiel–Positionen beruhen. Anwendung fanden die neuen Methoden in einem der derzeit stärksten Othello–Programme — LOGISTELLO, das hier auch als Testobjekt für empirische Untersuchungen eingesetzt wurde. Im einzelnen wurden Fortschritte in den Bereichen

- Merkmalskonstruktion
- Merkmalskombination
- Selektive Suche
- Spielfolgen–Strategien

erzielt, denen je ein Kapitel gewidmet ist:

Kapitel 2 beschreibt eine Methode zur schnellen Approximation bekannter Mobilitätsmerkmale bei Othello und zeigt eine neue, auf statistischen Schätzungen basierende Möglichkeit zur Konstruktion tabellenorientierter Merkmale hoher Güte auf.

Kapitel 3 stellt statistische Verfahren zur Gewichtung von Merkmalen bei linearen und quadratischen Kombinationsfunktionen vor, die empirisch verglichen werden.

Kapitel 4 führt ein neues probabilistisches Verfahren zur Steuerung einer selektiven Suche ein, das als einfache Erweiterungen des $\alpha\beta$–Algorithmus zu einer erheblichen Steigerung der Spielstärke führte.

Kapitel 5 präsentiert Strategien zum erfolgreichen Bestreiten einer Partienfolge. Es wird ein Algorithmus entwickelt, der das direkte Lernen aus Spielen in dem Sinne ermöglicht, daß sich Partien gemerkt werden und diese spätere Zugentscheidungen bei bekannten Stellungen unmittelbar beeinflussen.

Die zentralen Kapitel werden jeweils durch Abschnitte beschlossen, in denen die vorgestellten Methoden zusammengefaßt und diskutiert werden. Im **Anhang** wird das Spiel Othello kurz eingeführt und Stellungen aufgelistet, die zu Spielstärkevergleichen und Gütebestimmungen benutzt wurden. Ein Steckbrief LOGISTELLOs und eine Aufstellung der Resultate aller bisherigen Turniere mit Beteiligung des Programms beschließt die Arbeit.

Kapitel 2

Konstruktion von Merkmalen

Bei der Spielbaumsuche werden zur Beurteilung von Spielsituationen Funktionen — im folgenden Merkmale genannt — benutzt, welche gewisse Eigenschaften einer Stellung quantisieren, die mit dem Gewinn korreliert sind. Beispielsweise ist es beim Schachspiel sinnvoll, die Materialdifferenz als Merkmal zu betrachten, denn ein Ungleichgewicht der Kräfte entscheidet häufig eine Partie. Ein naheliegendes Merkmal beim Othellospiel ist die Eckendifferenz, denn Steine auf den Eckfeldern können vom Gegner nicht herumgedreht werden — sie sind stabil. Ausgehend von den Ecken können oft weitere Steine stabilisiert werden, was häufig den Sieg garantiert. Jedem, der schon einmal Schach oder Othello gespielt hat, ist klar, daß die obigen Merkmale nur einen groben Anhaltspunkt für die Einschätzung einer Stellung geben können. Die Betrachtung weiterer Merkmale ist notwendig. Falls das Spiel schon Gegenstand eingehender Untersuchungen war, so wird man leicht Merkmale aus Expertenwissen extrahieren können. Hinsichtlich der Spielbaumsuche gibt es hier einen Zielkonflikt: Einerseits soll ein Merkmal eine hohe Güte besitzen, also mit einer kleinen Fehlerwahrscheinlichkeit gewonnene von verlorenen Stellungen trennen können. Andererseits soll es auch schnell zu berechnen sein, denn längere Berechnungsdauern führen zu einer geringeren Suchtiefe und damit zu einer schwächeren taktischen Spielführung. Unter dem Aspekt der Spielstärkeoptimierung muß demnach ein Mittelweg gefunden werden. Ist man schließlich im Besitz einer Menge von Merkmalen, so stellt sich die Frage, wie und welche Funktionen hieraus zu einer Bewertungsfunktion kombiniert werden sollten. Hierauf gibt das nächste Kapitel Antworten.

Die folgenden Abschnitte beschäftigen sich mit Merkmalen für das Spiel Othello, die von LOGISTELLO benutzt werden. Diese sind das Resultat einer langen Reihe von Experimenten, bei denen eine Vielzahl weiterer Merkmale ersonnen, aber wegen mangelnder Güte, starker Korrelation mit anderen Merkmalen oder Ineffizienz verworfen wurden. Es werden bekannte Ansätze verfeinert und neue, auf statistischen Untersuchungen basierende vorgestellt. Die hierbei benutzten Methoden zur Konstruktion, zur schnellen Approximation und zum Gütevergleich von Merkmalen sind darüberhinaus allgemeiner Natur und damit auf andere Spiele übertragbar.

Für den Vergleich von Merkmalen wird im folgenden die *Diskordanz* herangezogen. Diese bewertet, in welchem Maße ein Merkmal die folgende, für die Spielbaumsuche wichtige Forderung erfüllt: Ist die Merkmalsausprägung einer Position mindestens so groß wie die einer anderen, so soll die erste Position hinsichtlich des Spielausgangs nicht schlechter sein als die zweite. Für ein Merkmal X ist die Diskordanz bezüglich einer Menge von Positionen P und Klassifizierung $K : P \to \{\mathcal{V}, \mathcal{U}, \mathcal{G}\}$ nach Verlust, Unentschieden bzw. Gewinn folgendermaßen definiert:

$$d_{P,K}(X) := \frac{|\{(p_1, p_2) \in P^2 \mid X(p_1) \geq X(p_2) \wedge K(p_1) < K(p_2)\}|}{|\{(p_1, p_2) \in P^2 \mid X(p_1) \geq X(p_2)\}|},$$

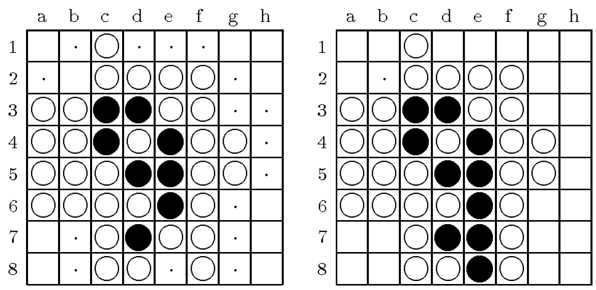
wobei $\mathcal{V} < \mathcal{U} < \mathcal{G}$ sei. Offensichtlich gilt $0 \leq d_{P,K}(X) \leq 1$. Je kleiner die Diskordanz ist, um so besser können zwei gegebene Positionen bezüglich Gewinn und Verlust im Mittel getrennt werden. Bei der MiniMax–Suche ist die lokale Trenngüte bezüglich aller Nachfolger einer Stellung wichtig. Diese kann durch die Diskordanz global und damit approximativ eingeschätzt werden.

Auf die Generierung der zur Schätzung der Diskordanz benutzten Beispielstellungen geht Abschnitt 3.3 genauer ein. Um Aussagen über das Verhalten der Merkmale in verschiedenen Spielphasen machen zu können, wurden die klassifizierten Positionen nach Steinanzahlen gruppiert. Die Gruppen bestanden jeweils aus ca. 50% für den anziehenden Spieler gewonnenen bzw. verlorenen Stellungen.

2.1 Aktuelle Mobilität

Neben dem Besitz von Ecken spielt bei Othello die Mobilität eine wesentliche Rolle. Hat ein Spieler im Mittelspiel wenige Züge, so sind dies meistens auch noch schlechte. Ein Beispiel zeigt Abbildung 2.1. Nach e8 hat Weiß nur noch die Zugmöglichkeit b2, die Ecke geht verloren und damit dieses Spiel. In der Literatur [z.B. ROSENBLOOM (1982), MITCHELL (1984)] wurden etliche Mobilitätsmerkmale diskutiert. Der einfachste Ansatz beschränkt sich hierbei auf die Bestimmung der Differenz der Zuganzahlen beider Parteien. Darüberhinaus wurden auch nichtlineare Funktionen benutzt, um zu modellieren, daß bei gleicher Zuganzahldifferenz für den Spieler mit mehr Zügen meistens die Situation vorteilhaft ist, bei denen es insgesamt weniger Zugmöglichkeiten gibt. Andere Varianten gewichten die einzelnen Zugmöglichkeiten, um Feldunterschieden gerecht zu werden (beispielsweise sind Eckenzüge meistens mehr wert als alle anderen).

Die Bestimmung der exakten Anzahl möglicher Züge in einer Stellung war bei den ersten Versionen von LOGISTELLO zeitraubend — ca. 50% der Zeit wurde darauf verwandt. Bei der Einführung der Mustermerkmale, die im übernächsten Abschnitt besprochen werden, ergab sich die Möglichkeit, die Mobilität sehr schnell zu approximieren. Die Idee ist hierbei, die globale Mobilität durch die Summe der lokalen Mobilitäten auf allen Strahlen des Spielfeldes — also auf Horizontalen, Vertikalen und Diagonalen — anzunähern. Zur Berechnung des neuen Merkmals sind dann nur 38 Tabellenzugriffe notwendig (16 für die Horizontalen und Vertikalen und 22 für die Diagonalen). Die benötigten Tabellenindizes sind bei jedem Zug



Schwarz am Zug   Weiß am Zug

Abbildung 2.1: Beispiele für hohe bzw. niedrige Mobilität leicht inkrementell anzupassen und werden auch für die Berechnung der Mustermerkmale benötigt. Somit ergibt sich insgesamt eine deutliche Beschleunigung der Auswertung der Mobilitätsmerkmale.

Eine schnelle Approximation nutzt wenig, wenn sie schlecht ist. Im schlimmsten Fall zählt das neue Merkmal eine Zugmöglichkeit vierfach, wie Abbildung 2.2 verdeutlicht. Andere Möglichkeiten werden wiederum nur einfach gezählt. Man muß sich also fragen, ob die Approximation eine mit dem Original vergleichbare Güte besitzt. Zur Beantwortung dieser Frage wurde bezüglich der in der Einleitung des Kapitels erwähnten Spielesammlung die Diskordanz der Merkmale geschätzt. Auf S. 20 befinden sich die Graphen für die Zuganzahldifferenz und deren Approximation in Abhängigkeit von der Steinanzahl. Es wird deutlich, daß die Näherung nur eine unwesentlich niedrigere Güte besitzt als das exakte Merkmal. Diese Erkenntnis und eine etwa um den Faktor 3 erhöhte Auswertungsgeschwindigkeit rechtfertigen den Einsatz der Mobilitätsapproximation in der Praxis.

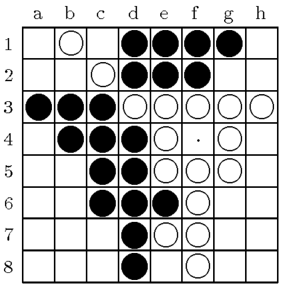


Abbildung 2.2: Ein Extremfall

2.2 Potentielle Mobilität

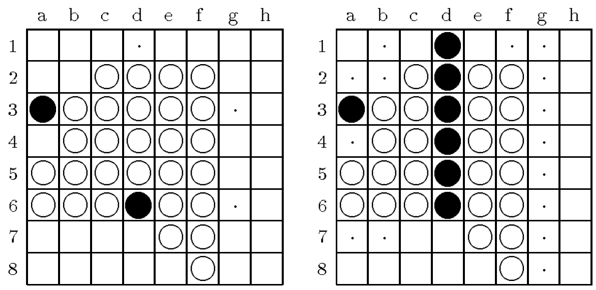
Die aktuelle Mobilität stellt ein taktisches Merkmal dar, da eine im Vergleich zum Gegner geringe Anzahl von Zügen meistens schnell zu einer verlorenen Position führt. Ein weiteres Merkmal — die potentielle Mobilität — soll Auskunft darüber geben, ob in Zukunft Zugmöglichkeiten in Aussicht stehen. Hiermit wird gewissermaßen ein Zwischenziel auf dem

Weg zur Erlangung aktueller Mobilität modelliert. ROSENBLOOM (1982) beschreibt drei Maße für die potentielle Mobilität:

1. Die Anzahl freier, an gegnerische Steine angrenzender Felder.
2. Die Anzahl gegnerischer Steine, die an freie Felder angrenzen.
3. Die Summe aller Anzahlen freier Felder um gegnerische Steine.

Die dahinter stehende Idee illustriert Abbildung 2.3. Zieht Schwarz auf d1, so erhöht sich hierdurch seine aktuelle Mobilität drastisch. Somit liegt es nahe, die Beziehung von freien Positionen zu gegnerischen Steinen zu betrachten.

Auch obige Merkmale können durch die Benutzung von Tabellen schnell approximiert werden. Wiederum wird hierzu die globale Bewertung durch lokale Strahlwerte angenähert. Betrachtet man die drei Merkmale, so kann man unter ihnen eine große Korrelationen vermuten, was anhand der Beispielpositionen auch leicht zu bestätigen ist. Beim Vergleich der Diskordanzen stellte sich außerdem beim 3. Merkmal die höchste Güte heraus. Dieses Merkmal wurde approximiert, indem lokal zu einem Strahl für jede Partei die Summe der Anzahl freier Positionen, die an gegnerische Steine angrenzen und auf die der Gegner lokal nicht setzen kann, ermittelt wird, um danach die Differenz daraus zu bilden. Zu beachten ist, daß auf Kantenstrahlen die Betrachtung potentieller Mobilität nicht sinnvoll ist, da lange Ketten von Steinen gleicher Farbe dort nicht durch Setzen auf einen anderen Strahl zerstört werden können. Betrachtet man die Diskordanz dieses Merkmals auf S. 20, so fällt auf, daß es bis zum späten Mittelspiel — bei ca. 36 Steinen auf dem Brett — eine höhere Güte als die aktuelle Mobilität besitzt, aber danach schnell an Bedeutung verliert.



Schwarz am Zug Züge für Schwarz nach d1

Abbildung 2.3: Beispiel für hohe potentielle Mobilität

2.3 Muster

Im Zusammenhang mit der Eckenrelevanz bei Othello ist es wichtig, Konstellationen zu erkennen, die in Zukunft zum Eckenbesitz bzw. Eckenverlust führen können. Hierbei steht die Taktik auf den Spielfeldkanten im Vordergrund unter Beachtung der X–Felder (s. Anhang A). Rosenbloom (1982) und Lee & Mahajan (1986,1990) benutzen hierzu Tabellen, mit denen jede Kantenkonstellation bewertet wird. Dieses ermöglicht eine sehr schnelle Bewertung, denn es sind nur vier Tabellenzugriffe und drei Additionen notwendig, um selbst kompliziertere Kantensituationen zu bewerten, bei denen erst eine Baumsuche den Gewinn oder Verlust einer Ecke anzeigen würde. In den Arbeiten wurden die Tabelleneinträge heuristisch mittels vorgegebener, plausibler Bewertungen und (probabilistischen) MiniMax–Suchen belegt.

Im folgenden wird obiger Tabellenansatz verallgemeinert. Hierzu werden nun nicht mehr nur die Kanten betrachtet, sondern potentiell alle Mengen von Feldern — von nun an Muster genannt. Eine in obigem Sinne heuristische Belegung der Tabellen kommt bei dieser Allgemeinheit nicht mehr in Betracht, da hierbei eine Vielzahl von Bewertungen intuitiv vorgegeben werden müßte. Als wirkungsvoll hat sich folgender Ansatz herausgestellt:

> *Der Wert einer Musterinstanz ist die bedingte Wahrscheinlichkeit des Gewinns für Schwarz unter Auftreten der Instanz in der zu bewertenden Position.*

Das Ziel ist hierbei, langfristige Auswirkungen von lokalen Brettkonstellationen schnell mittels Tabellenzugriffen zu bewerten. Die Auswertung geschieht wie auch schon bei den Kantentabellen durch Addition der Werte aller Instanzen eines Musters auf dem Brett.

Die Wahrscheinlichkeiten können mittels Beispielpositionen geschätzt werden. Der Grundraum sollte dabei nicht die Gesamtheit aller möglichen Stellungen umfassen, sondern nur relevante Stellungen, die in einem Spiel — oder genauer in den zu bewertenden Blattstellungen — zu erwarten sind. Eine einfache Möglichkeit, die Tabelleneinträge näherungsweise zu schätzen, liegt in der Bestimmung der bedingten relativen Häufigkeiten bezüglich der vorliegenden Spielesammlung. Dabei kommt es ganz wesentlich auf die Zusammensetzung der Beispiele an. Hier sei nur ein Szenario vorgestellt: Betrachtet man das Hauptdiagonal–Muster und dort speziell Instanzen mit unbesetzten Ecken aber einem besetzten X–Feld, bei denen also die angrenzende Ecke bedroht ist, so wird man zu unterschiedlichen Ergebnissen gelangen, wenn man einerseits nur Meisterpartien und andererseits nur Spiele von Anfängern gegen gute Spieler zur Schätzung der Einträge heranzieht. Setzt ein Experte nämlich derartige X–Züge ein, so führt dieses häufig zum Gewinn, da hierdurch die Mobilität des Gegners im Bereich der benachbarten Ecke des Spielfelds eingeschränkt wird. Werden sie dagegen von Anfängern gespielt, dann widerlegt sie ein guter Spieler fast ausnahmslos, indem er ohne Gefahr die Ecke besetzt. Da sich an die Stellungsbewertung üblicherweise keine weitere Baumsuche anschließt, kann man mit hoher Wahrscheinlichkeit davon ausgehen, daß ein vorheriger X–Zug zum Spielverlust führt. Aus diesem Grund sollte man nicht nur aus Meisterpartien lernen. Ähnliche Situationen kommen auch im Schachspiel vor: Ein guter Spieler opfert zuweilen Material, um den Druck auf den Gegner verstärken zu können — ein Anfänger läßt sich hingegen oft Material ohne Kompensation entwenden. Auf die Problematik der Beispielauswahl wird im nächsten Kapitel noch näher eingegangen, da sie auch bei Bestimmung von Merkmalsgewichten auftritt.

Wichtig für die Güte der Schätzungen ist eine große Beispielanzahl, da die Anzahl der Einträge — bei Mustern mit 8 Feldern sind dies schon $3^8 = 6561$ — sehr groß ist. Will man die Bewertung noch abhängig von der Spielphase machen, die bei Othello gut in Abhängigkeit von der Steinanzahl modelliert werden kann, so steigt der Bedarf an Beispielen weiter an. In diesem Falle ist es aber möglich, Tabellenwerte über die Phasen zu glätten, so daß der Effekt gedämpft wird. Zum zweiten ist eine möglichst fehlerfreie Klassifizierung der Beispiele nach Gewinn und Verlust wünschenswert, denn anderenfalls werden die Schätzungen verzerrt.

Ein weiterer beachtenswerter Aspekt ist das lokale Vorkommen von Musterinstanzen. Ecken werden beispielsweise zumeist erst zum Ende des Spiels besetzt. Wenn dieses aber schon in der Eröffnung passiert, soll auch dort eine adäquate Bewertung zur Verfügung stehen. Demnach ist im Falle einer Phasen–abhängigen Bewertung eine Extrapolation der Schätzwerte notwendig.

Um einen Eindruck von obigen Mustermerkmalen zu erhalten, sind in Abbildung 2.4 die Werte ausgewählter Kanteninstanzen über der Steinanzahl nach Extrapolation und Glättung dargestellt. Aus Speicherplatzgründen wurden 12 Spielphasen betrachtet, die durch Gruppierung der Stellungen gemäß vier aufeinanderfolgender Steinanzahlen beginnend bei 10 definiert wurden. Für jeden Merkmalswert wurde nur ein Byte spen-

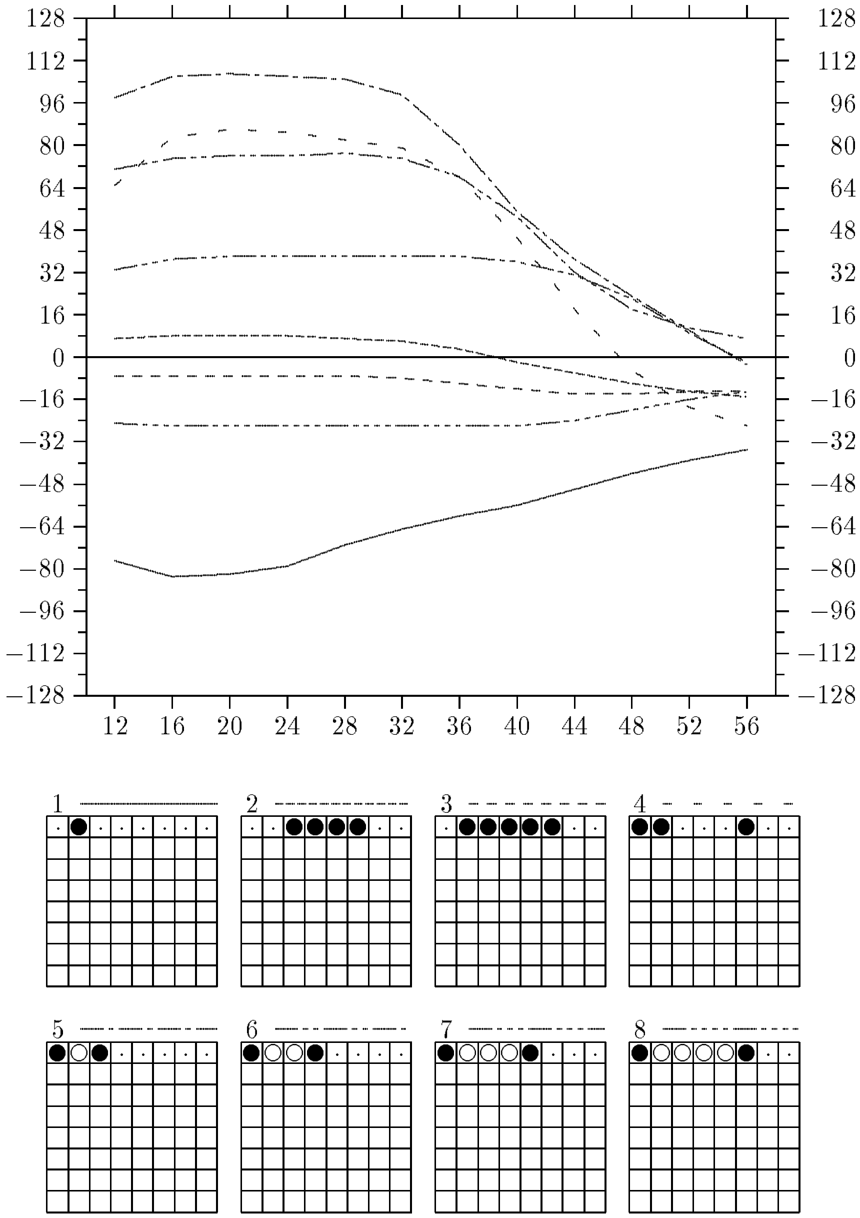


Abbildung 2.4: Bewertungen ausgewählter Kanteninstanzen

iert, so daß -127 einen sicheren Verlust bzw. $+127$ einen sicheren Gewinn aus der Sicht von Schwarz kodiert. Wie man sieht, sind die Wahrscheinlichkeiten von der Spielphase abhängig. Die besondere Bedeutung der Ecken schlägt sich in den Kantenbewertungen nieder. So ist Situtation 1 sehr gefährlich für Schwarz, da die linke Ecke leicht angegriffen werden kann. Je früher diese Konstellation vorkommt, um so riskanter ist sie. Auch die dritte Situation — eine unbalancierte Kante — ist im allgemeinen, wenn sie nicht balanciert werden kann, zu vermeiden, da oft die rechte Ecke von Weiß geopfert wird, um die gesamte restliche Kante zu erhalten. Die Instanzen 5 bis 8 verdeutlichen, daß es bei Kanten nicht nur um Ecken geht. Kann sich eine Partei innere Steine sichern, so ist dies manchmal sogar von größerem Wert als der Eckenbesitz selbst. Beachtenswert und für einen mäßigen Othello–Spieler schwerer zu motivieren sind die Vorzeichenwechsel bei Instanz 2 und 4. Eine mögliche Schwäche der Konstellation 4 ist die ungerade Anzahl freier Felder zwischen den schwarzen Steinen. Wenn es Schwarz nicht gelingt, zuerst dorthin zu setzen, so hat Weiß einen sogenannten Paritätsvorteil: Der Spieler kann zuerst in ein ungerades Gebiet setzen und somit Schwarz zwingen, später als erstes in ein anderes Gebiet zu setzen. Insgesamt läßt sich feststellen, daß Musterinstanzen durch Schätzung der bedingten Wahrscheinlichkeiten ohne intuitive Vorgabe von Parametern adäquat bewertet werden können.

Welche Muster sollen nun gewählt werden? Eine Beschränkung der Feldanzahl der Muster ergibt sich direkt aus einer Speicherplatz– bzw. Schätzgütebetrachtung: So werden für ein 10–Felder–Muster bei 12 Spielphasen schon $59049 \cdot 12 = 708.588$ Bytes benötigt, wenn keine Symmetrie vorliegt. Weiterhin soll die Schätzung der Tabellenwerte auch eine Generalisierung ermöglichen. Ist aber die Anzahl zur Verfügung stehender Spiele in der Größenordnung der Anzahl zu schätzender Tabelleneinträge, so ist eine Verallgemeinerbarkeit nicht gewährleistet, denn pro Beispielposition gibt es höchstens 8 Beiträge zur Schätzung. Da „nur“ ca. 50.000 Spiele zur Verfügung standen, kamen für die zu wählenden Muster nur solche mit höchstens 8 Feldern in Betracht. In Verbindung mit den Mobilitätsmerkmalen liegt die Benutzung aller Horizontalen, Vertikalen und Diagonalen nahe, da die Indizes dort auch verwendet werden können. Darüberhinaus wurden eine Vielzahl weiterer Muster untersucht. Die Abbildungen 2.5 bis 2.9 geben einen Überblick. Zum Gütevergleich der Muster wurde die Diskordanz bzgl. der Trainingsmenge bestimmt und in Abhängigkeit von der Spielphase dargestellt. Hierbei wurden auch Muster mit 9 bzw. 10 Feldern betrachtet, bei denen die Schätzungen wie oben erläutert zwar nicht generalisierungsfähig sind, deren Güte aber untereinander verglichen werden kann. Häufig werden in Othello–Programmen Tabellen für Muster 31 eingesetzt, da die Kantenbewertung nur in Verbindung mit den X–Feldern sinnvoll erscheint. Wie jedoch auf S. 20 zu sehen ist, hat das (2×5)–Eckenmuster 33 eine deutlich höheren Güte. Der Grund mag hierfür in der größeren Gesamtanzahl beteiligter Felder liegen oder in der Möglichkeit, durch Muster 33 auch den Zugriff auf Kantenfelder einschätzen zu können. Gleiches gilt auch für die 8–Felder–Muster 7 und 11.

Insgesamt ist auffällig, daß die Diskordanz vieler Muster mit der Steinanzahl sinkt. Bei Eckenmustern ist dieses nicht verwunderlich, da Ecken sehr bedeutend sind, aber normalerweise erst zum Ende des Spiels hin besetzt werden. Bei anderen Mustern ist zu erkennen, daß sich der Verlauf bei 44 Steinen ändert. Auch dieses verblüfft nicht — sind doch die Beispielpartien ab gerade dieser Steinanzahl optimal bzgl. Gewinn gespielt. Bedingt durch die mit fallender Steinanzahl abnehmende Klassifizierungsgüte ergibt sich eine größere Diskordanz in früheren Phasen.

Ausgehend von einer Fülle plausibel erscheinender Muster kann nun durch obige Gütebetrachtung eine Vorauswahl getroffen werden. Das folgende Kapitel geht auf die engere Auswahl der Merkmale und deren Verknüpfung ein. In Abbildung 2.9 ist die Diskordanz von mit dortigen Methoden bestimmten, von der Spielphase abhängigen Linearkombinationen der wichtigsten Strahlmerkmale, Mustermerkmal 11 und der beiden Mobilitätsmerkmale dargestellt. Wie zu erwarten war, ist hierbei im Vergleich zu jedem Einzelmerkmal die Diskordanz deutlich kleiner.

2.4 Diskussion

In diesem Kapitel wurde eine Möglichkeit aufgezeigt, bekannte Merkmale schnell mit großer Güte zu approximieren. Die Idee war hierbei, globale Aspekte der Position durch lokale anzunähern, um hierdurch eine sehr viel schnellere, auf Tabellenzugriffen basierende Auswertung zu ermöglichen. Die Baumsuche erfuhr hierdurch insgesamt eine wesentliche Beschleunigung bei vergleichbarer Bewertungsgüte.

Desweiteren ist mit den Mustern eine neue Art von Merkmalen vorgestellt worden, bei der taktische wie auch strategische Aspekte der Stellung wiederum mittels Tabellenzugriffen schnell bewertet werden können. Grundlage der Bewertung bildet hierbei die bedingte Gewinnwahrscheinlichkeit unter Auftreten einer Musterinstanz, wie beispielsweise einer bestimmten Kantenkonstellation bei Othello. Die Wahrscheinlichkeiten wurden mittels einer großen Menge von Beispielpositionen geschätzt und in Tabellen abgelegt. Damit ist die Bewertung völlig unabhängig von menschlicher Intuition, die bei der numerischen Einschätzungen von Positionen oft fehlerbehaftet ist, aber z.B. bei LEE & MAHAJAN (1986,1990) eine große Rolle bei der Bewertung spielte.

Durch den Vergleich verschiedener Muster hinsichtlich ihrer Trenngüte wurde es möglich, eine Vorauswahl aus einer Vielzahl von plausiblen Alternativen zu treffen. Letzteres kann als ein Schritt hin zu einer automatischen Merkmalsgenerierung angesehen werden. Hierzu muß der Musterraum speziell bei Othello aber noch weiter geeignet eingeschränkt werden, denn die Anzahl der Muster schon höchstens der Größe 8 ist mindestens $\binom{64}{8}/8 \geq 4 \cdot 10^9$. Aussichtsreich erscheinen unter diesem Aspekt Ansätze, bei denen neue Muster aus bekannten im Hinblick auf das vorgestellte Gütemaß zusammensetzt werden. Weiterhin kann der Ansatz dahingehend verallgemeinert werden, daß auf die Tabellen nicht direkt mittels Musterindizes zugegriffen wird, sondern zunächst Musterinstanzen auf Indizes geeignet (nicht injektiv) abgebildet werden. Hierdurch kann die Anzahl zu schätzender Tabelleneinträge wesentlich reduziert werden, um so mit weniger Beispielen auszukommen und sogar die Modellierung von Musterinteraktionen zu ermöglichen.

In dieser Allgemeinheit ist die Anwendung des Musteransatzes zur Stellungsbewertung auch bei anderen Spielen denkbar — beispielsweise für die Einschätzung der Güte von Bauernkonstellationen beim Schach — und erinnert an die von SAMUEL (1967) eingeführten Signaturtabellen zur Modellierung von Merkmalsinteraktionen. Hierbei dienen eine Vielzahl von Merkmalen in quantisierter Form als Indizes auf hierarchisch angeordnete Tabellen, deren Einträge mit Hilfe von Zügen aus Expertenspielen besetzt werden und wiederum quantisiert als Indizes für die nächste Tabellenstufe dienen. Aus dieser Sicht sind die hier vorgestellten Mustermerkmale Signaturtabellen der Tiefe 1, bei denen Spielfeldausschnitte direkt als unquantisierte Eingabe dienen und die Bewertung freilich auf eine andere Art erfolgt.

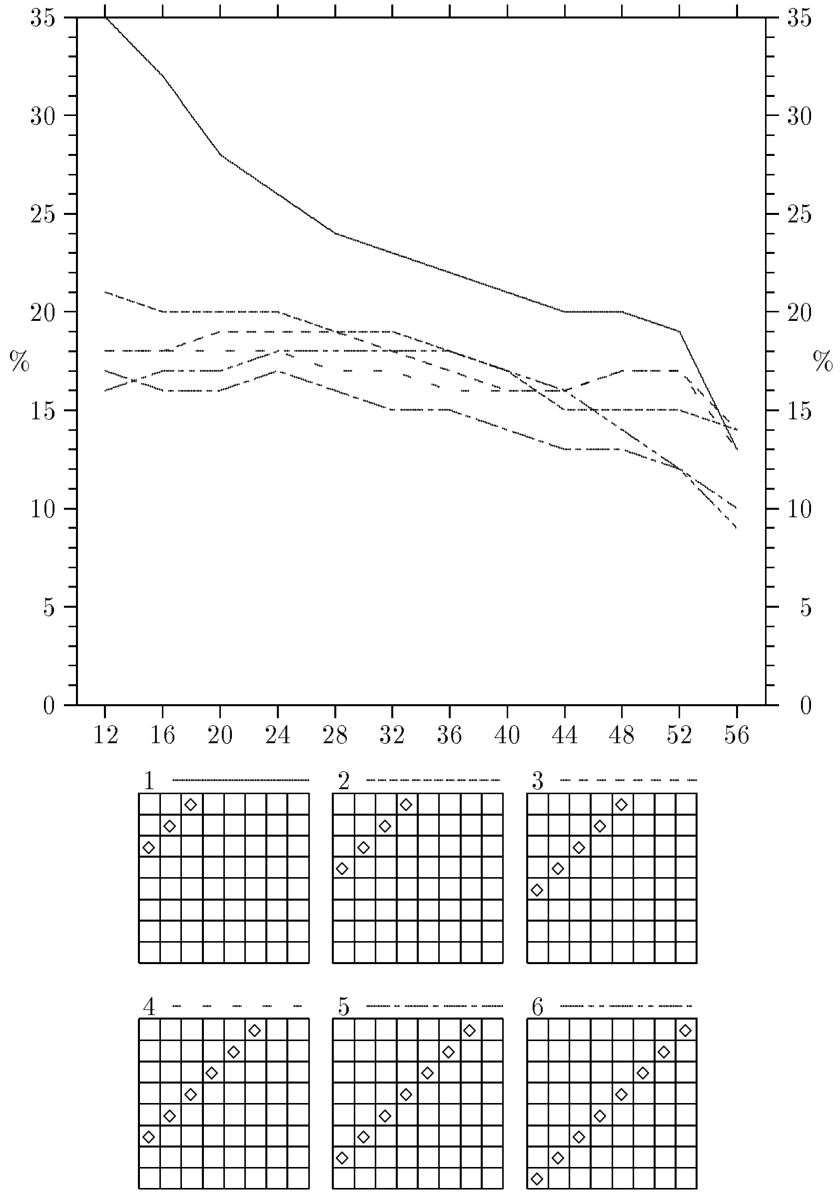


Abbildung 2.5: Diskordanzen I

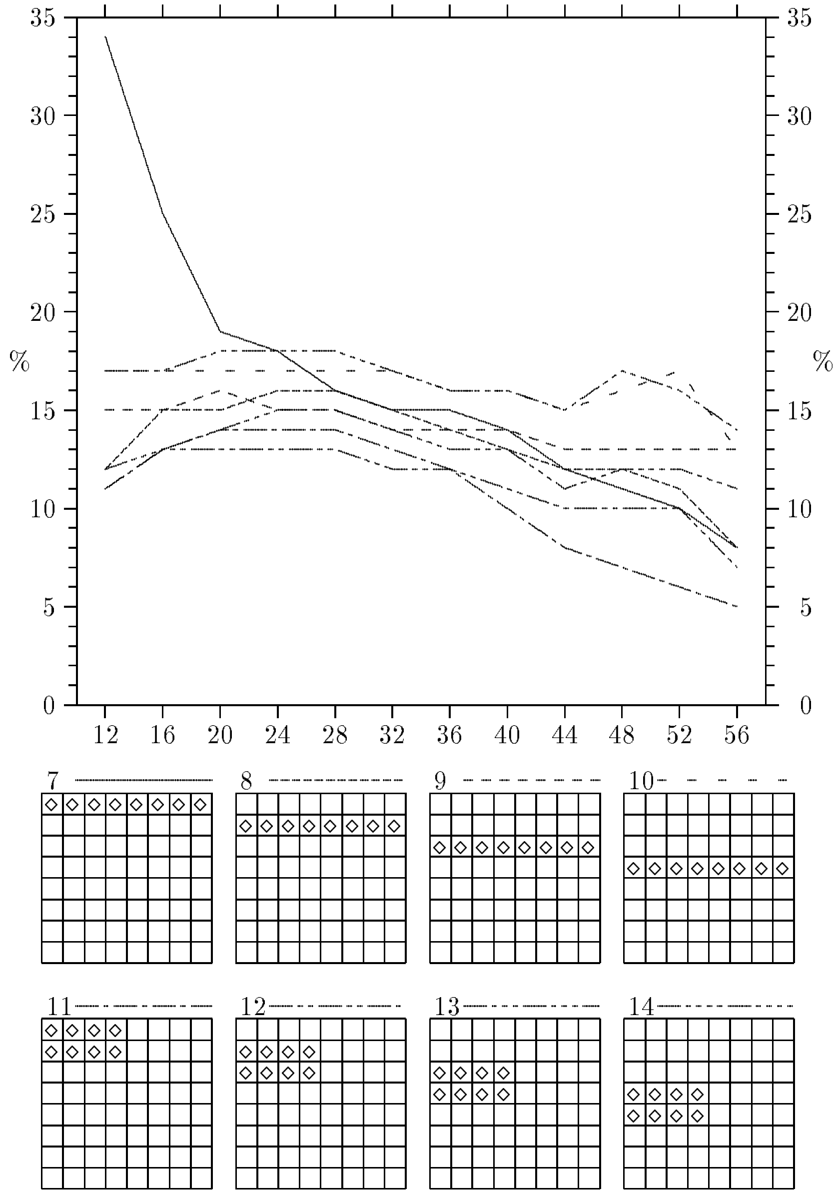


Abbildung 2.6: Diskordanzen II

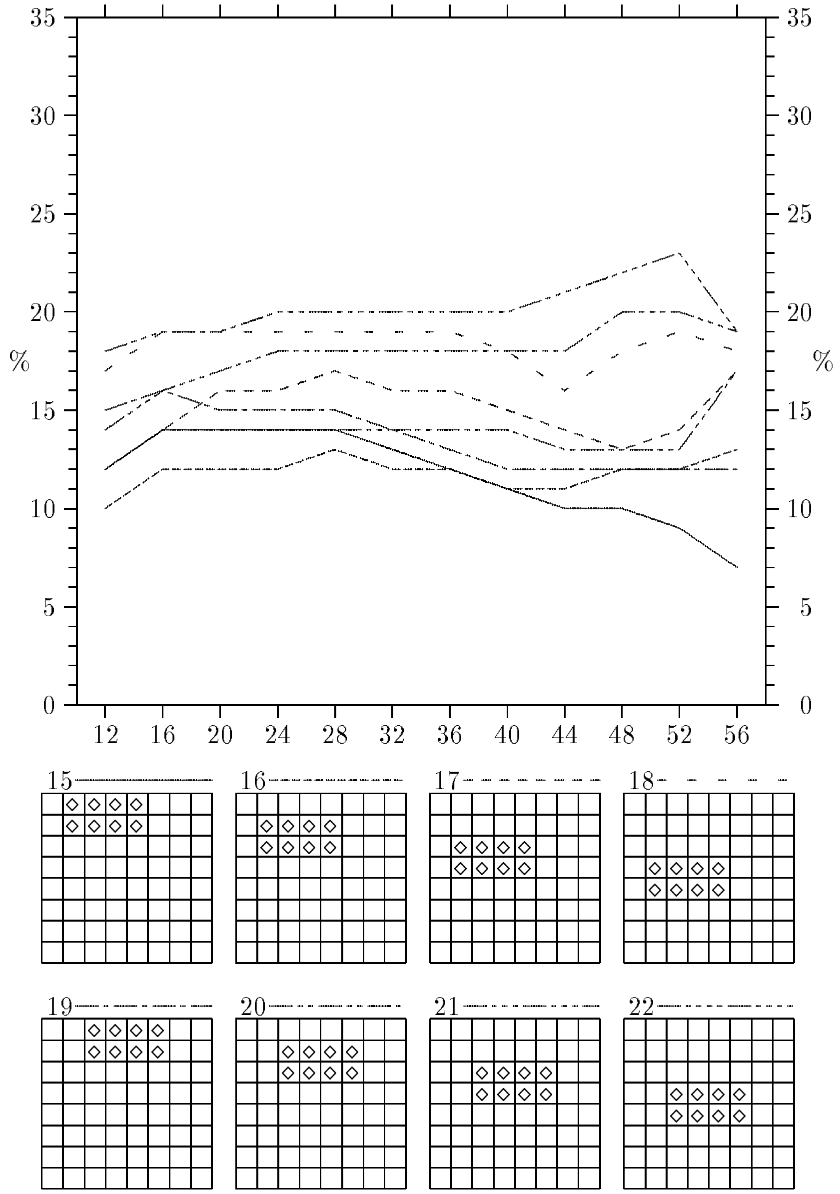


Abbildung 2.7: Diskordanzen III

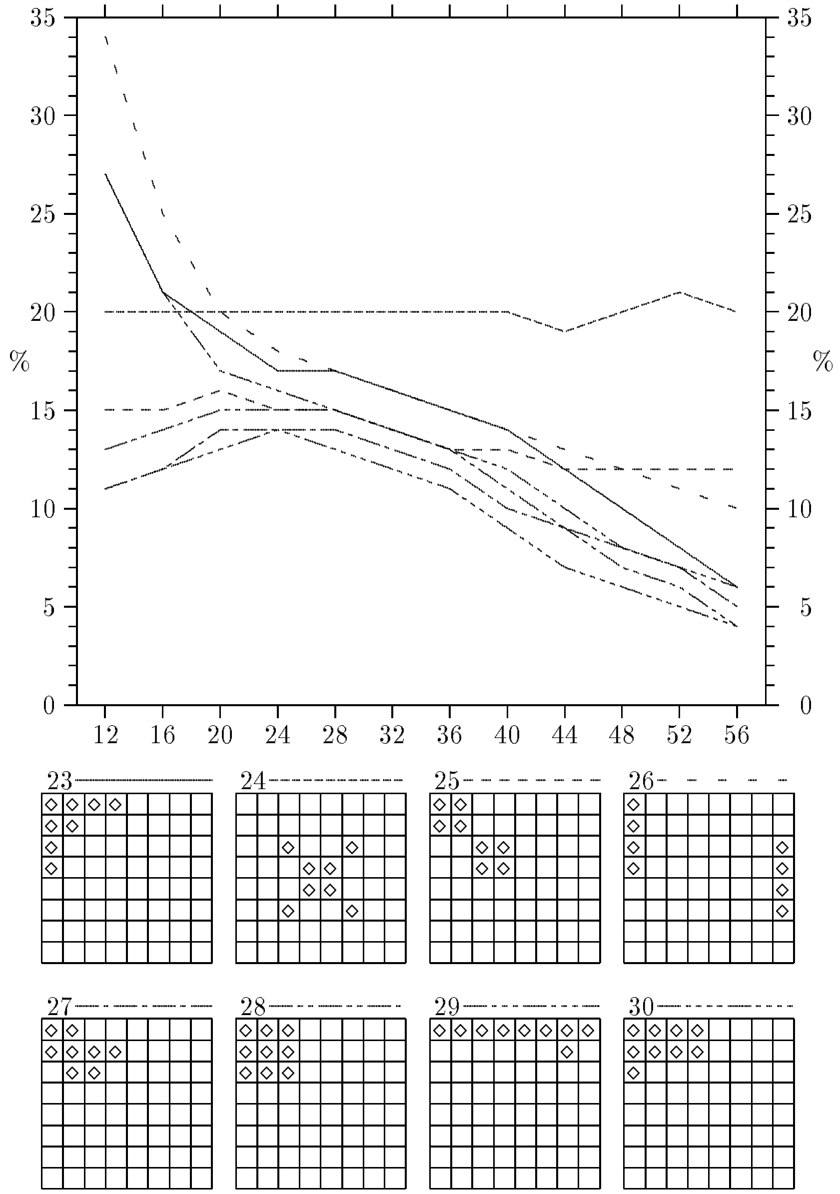
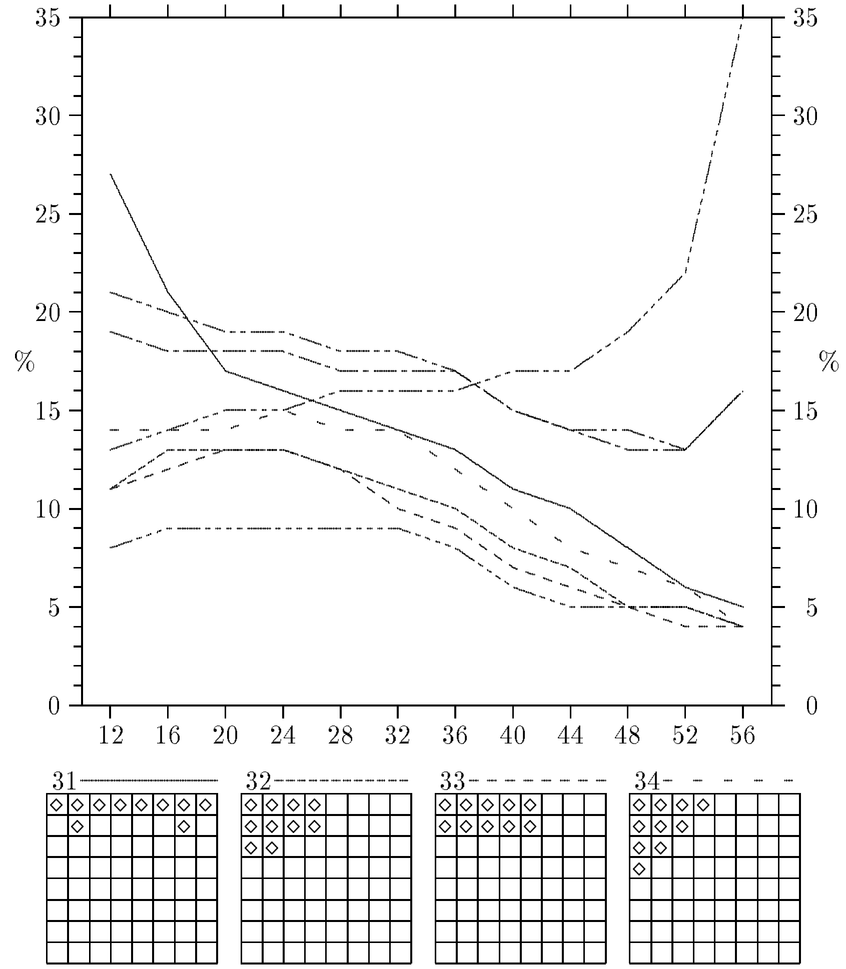


Abbildung 2.8: Diskordanzen IV



35		exakte Mobilität
36		approximierte Mobilität
37		potentielle Mobilität
38		Linearkombination der Merkmale 3,...,11,36,37

Abbildung 2.9: Diskordanzen V

Kapitel 3

Gewichtung der Merkmale

Die Aufgabe einer Bewertungsfunktion in der Spielbaumsuche ist es, eine vorgelegte Position im Hinblick auf die Gewinnchance für den Spieler am Zug zu bewerten. Da eine in diesem Sinne optimale Funktion für ein nicht–triviales Spiel aufgrund des aktuellen Erkenntnisstandes meist nur mit enormem Zeitaufwand berechenbar ist, muß in der Praxis auf schnellere Approximationen zurückgegriffen werden. Im Hinblick auf deren Einsatz in einer klassischen MiniMax–Suche genügt es hierbei, vielversprechendere Stellungen auf höhere Werte abzubilden — die absoluten Werte sind irrelevant. Die Situation ändert sich, wenn man die Bewertungsfunktion abhängig von der Spielphase machen will und z.B. im Rahmen einer selektiven Vertiefung die Bewertung von Stellungen aus unterschiedlichen Spielphasen zugelassen wird. In diesem Fall muß nämlich die Vergleichbarkeit der Bewertungen gewährleistet sein. Hierfür ist die Existenz einer phasenunabhängigen Wertinterpretation wie z.B. die Gewinnwahrscheinlichkeit oder speziell beim Othello die erwartete Steindifferenz hinreichend.

Üblicherweise werden zur Bildung einer Bewertungsfunktion Merkmale verknüpft. Die wohl einfachste Form hierfür ist eine Linearkombination, bei der die Merkmalsgewichte geeignet gewählt werden müssen. In der Pionierzeit der Spieleprogrammierung wurden die Gewichte „sinnvoll“ gewählt und in einem iterativen Hill–Climbing–Prozeß manuell verbessert, bis die Spielstärke dadurch nicht mehr gesteigert werden konnte. Dieser Ansatz ist unintuitiv, mühsam und zeitaufwendig. Samuel (1959,1967) beschreibt als erster Möglichkeiten der automatischen Verbesserung von Bewertungsfunktionen. Seither wurden eine Vielzahl von Ansätzen zur

Optimierung parametrisierter Bewertungsfunktionen untersucht. Hierbei können zwei Vorgehensweisen unterschieden werden:

Zuganpassung Eine Menge von Trainingspositionen mit in diesen Stellungen guten Zügen ist gegeben. Es wird versucht, durch Wahl der Parameter die Trefferquote, die mittels einer Baumsuche erreicht wird, zu maximieren.

Wertanpassung Auch hier liegt eine Menge von Positionen vor. Diese sind mit einem Wert markiert wie z.B. der MiniMax–Bewertung, die sich aus einer Baumsuche unter Verwendung eines älteren Parametersatzes ergibt, oder mit dem Resultat eines sich anschließenden Spiels aus der Sicht des anziehenden Spielers. Es wird versucht, die Parameter so zu wählen, daß die Werte möglichst gut approximiert werden.

Bei Verfahren der Zuganpassung — wie beispielsweise von Marsland (1985), v.d.Meulen (1989) oder Mysliwietz (1994) beschrieben — gibt es bei der Kombinationsfunktion zwei Freiheitsgrade, da sie mit einer Konstanten größer als Null multipliziert werden oder zu ihr eine beliebige Konstante addiert werden kann, ohne daß sich der bei einer MiniMax–Suche gefundene Zug ändert. Möchte man mittels Zuganpassung Spielphasen–abhängige Bewertungsfunktionen bestimmen und in der Suche auch Stellungen aus unterschiedlichen Phasen z.B. im Rahmen einer selektiven Vertiefung bewerten, so müssen die Konstanten geeignet gewählt werden. Dieses Problem wurde bisher noch nicht eingehend untersucht und erscheint — wenn überhaupt — nicht einfach lösbar zu sein, da die erhaltenen Bewertungen zunächst keine globale Interpretation besitzen. Beim Schachspiel wird üblicherweise die additive Konstante und das Gewicht der Materialdifferenz festgelegt. Dieses Vorgehen ist aber nur in dem Fall vertretbar, bei dem wie hier ein Merkmal in allen Spielphasen dominant ist. Nimmt hingegen der Wert eines Merkmals in Abhängigkeit von der Spielphase ab und der eines anderen zu, so würde die Bewertung durch Festhalten eines Gewichts verzerrt und eine Vergleichbarkeit der Werte aus verschiedenen Phasen wäre nicht mehr gegeben.

In dieser Hinsicht erfolgversprechender sind Verfahren zur Wertanpassung. Die hierbei erhaltenen Bewertungen sind über Spielphasen hinweg vergleichbar, wenn die Beispielpositionen mit Werten markiert werden, die eine Phasen–unabhängige Bedeutung haben. Erstmals wurden hierzu von Mitchell (1984) Othello–Stellungen mit dem Spielresultat in Form der Steindifferenz markiert und versucht, diesen Wert mittels einer in den Merkmalen linearen Funktion zu approximieren. Durch Verwendung einer multiplen linearen Regression zur Bestimmung der Gewichte wurde es hierbei auch möglich, die statistische Relevanz der Merkmale zu untersuchen. Ein weiteres statistisches Verfahren zur Wertanpassung benutzten Lee & Mahajan (1988). Wiederum wurden Stellungen mit den jeweiligen Spielresultaten markiert, wobei aber nur nach Gewinn bzw. Verlust unterschieden wurde. Unter der Annahme, daß die Merkmale in den beiden Klassen multivariat normalverteilt sind, konnte so eine quadratische Funktion angegeben werden als ein Maß für die Wahrscheinlichkeit, daß die zu bewertende Stellung bzgl. der vorliegenden Merkmalsausprägungen zur Klasse der gewonnenen Stellungen gehört. Somit ist auch hier die Vergleichbarkeit gegeben. Darüberhinaus ist die Methode im Gegensatz zu Mitchells auch auf Spiele übertragbar, bei denen es keine Abstufung des Gewinns wie bei Othello gibt.

In den folgenden Abschnitten wird zum einen der von Lee & Mahajan (1988) benutzte Ansatz auch in einer spezielleren Form und zum anderen ein weiteres, bisher im Bereich der Spielbaumsuche noch nicht eingesetztes statistisches Verfahren zur Gewichtsbestimmung in einem solchen Umfang vorgestellt, wie es zum praktischen Einsatz ausreichend ist. Als weiterführende Literatur sei hier z.B. auf Duda & Hart (1973), Hand (1981), Agresti (1990) und McCullagh & Nelder (1989) verwiesen. Im Anschluß hieran werden die Methoden empirisch bzgl. der resultierenden Spielstärke verglichen.

Ihnen gemeinsam ist die folgende formale Grundlage:

- Ω ist die Menge der betrachteten Positionen.

- $Y : \Omega \to \{\mathcal{G}, \mathcal{V}\}$ klassifiziert Positionen hinsichtlich Gewinn oder Verlust aus der Sicht des anziehenden Spielers. Möchte man auch unentschiedene Spielausgänge betrachten, so bietet sich an, solche Spiele jeweils als verloren und gewonnen in die unten erwähnte Beispielfolge aufzunehmen.

- $X_1, \ldots, X_n : \Omega \to \mathbb{R}$ sind die Merkmale.

- Für eine Position $\omega \in \Omega$ mit $\boldsymbol{x} = (X_1, \ldots, X_n)(\omega)$ ist deren Bewertung die unter den Merkmalsausprägungen bedingte Gewinnwahrscheinlichkeit: $W(\omega) = P(Y = \mathcal{G} \mid (X_1, \ldots, X_n) = \boldsymbol{x}) =: P(\mathcal{G} \mid \boldsymbol{x})$.

- Zur Parameterschätzung liegen N klassifizierte Beispielpositionen $\omega_1, \ldots, \omega_N \in \Omega$ vor mit $\boldsymbol{x}_i = (X_1, \ldots, X_n)(\omega_i)$ und $y_i = Y(\omega_i)$.

3.1 Diskriminanzfunktionen für normalverteilte Merkmale

Unter der Annahme, daß die bedingte Dichtefunktion $p(\boldsymbol{x} \mid K)$ und die apriori–Wahrscheinlichkeit $P(K)$ für $K \in \{\mathcal{G}, \mathcal{V}\}$ bekannt sind, liefert die Bayes–Regel die folgende Darstellung für die unter $\boldsymbol{x}$ bedingte Gewinnwahrscheinlichkeit:

$$\begin{aligned} P(\mathcal{G} \mid \boldsymbol{x}) &= \frac{p(\boldsymbol{x} \mid \mathcal{G})P(\mathcal{G})}{p(\boldsymbol{x})} \\ &= \frac{p(\boldsymbol{x} \mid \mathcal{G})P(\mathcal{G})}{p(\boldsymbol{x} \mid \mathcal{G})P(\mathcal{G}) + p(\boldsymbol{x} \mid \mathcal{V})P(\mathcal{V})} \\ &= \frac{1}{1 + \left[p(\boldsymbol{x} \mid \mathcal{V})P(\mathcal{V})\right] / \left[p(\boldsymbol{x} \mid \mathcal{G})P(\mathcal{G})\right]}, \end{aligned}$$

falls nicht durch Null geteilt wird. Geht man weiter davon aus, daß die Merkmale bedingt unter der Klasse multivariat normalverteilt sind, also

$$p(\boldsymbol{x} \mid K) = (2\pi)^{-n/2} |\boldsymbol{\Sigma}_K|^{-1/2} \exp\left\{-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_K)\boldsymbol{\Sigma}_K^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_K)'\right\}$$

mit Erwartungswertvektor $\boldsymbol{\mu}_K$ und Kovarianzmatrix $\boldsymbol{\Sigma}_K$ für $K \in \{\mathcal{G}, \mathcal{V}\}$ gilt, so folgt im Falle gleicher apriori–Wahrscheinlichkeiten

$$P(\mathcal{G} \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-f(\boldsymbol{x}))}$$

für die Diskriminanzfunktion f gegeben durch

$$\begin{aligned} f(\boldsymbol{x}) &= -\tfrac{1}{2}\Big\{(\boldsymbol{x} - \boldsymbol{\mu}_\mathcal{G})\boldsymbol{\Sigma}_\mathcal{G}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_\mathcal{G})' - (\boldsymbol{x} - \boldsymbol{\mu}_\mathcal{V})\boldsymbol{\Sigma}_\mathcal{V}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_\mathcal{V})' + \\ &\qquad \log|\boldsymbol{\Sigma}_\mathcal{G}| - \log|\boldsymbol{\Sigma}_\mathcal{V}|\Big\} \\ &= -\Big\{\tfrac{1}{2}\boldsymbol{x}\left(\boldsymbol{\Sigma}_\mathcal{G}^{-1} - \boldsymbol{\Sigma}_\mathcal{V}^{-1}\right)\boldsymbol{x}' + \left(\boldsymbol{\mu}_\mathcal{V}\boldsymbol{\Sigma}_\mathcal{V}^{-1} - \boldsymbol{\mu}_\mathcal{G}\boldsymbol{\Sigma}_\mathcal{G}^{-1}\right)\boldsymbol{x}' + \\ &\qquad \tfrac{1}{2}\left(\boldsymbol{\mu}_\mathcal{G}\boldsymbol{\Sigma}_\mathcal{G}^{-1}\boldsymbol{\mu}_\mathcal{G}' - \boldsymbol{\mu}_\mathcal{V}\boldsymbol{\Sigma}_\mathcal{V}^{-1}\boldsymbol{\mu}_\mathcal{V}' + \log|\boldsymbol{\Sigma}_\mathcal{G}| - \log|\boldsymbol{\Sigma}_\mathcal{V}|\right)\Big\}. \end{aligned}$$

Im Fall $\boldsymbol{\Sigma}_{\mathcal{G}} = \boldsymbol{\Sigma}_{\mathcal{V}} =: \boldsymbol{\Sigma}$ kann der Ausdruck stark vereinfacht werden — man erhält anstelle einer quadratischen die folgende lineare Diskriminanzfunktion:

$$f(\boldsymbol{x}) \quad = \quad (\boldsymbol{\mu}_{\mathcal{G}} - \boldsymbol{\mu}_{\mathcal{V}})\boldsymbol{\Sigma}^{-1}\{\boldsymbol{x} - (\boldsymbol{\mu}_{\mathcal{V}} + \boldsymbol{\mu}_{\mathcal{G}})/2\}'.$$

Interessant ist hier, daß diese Funktion auch eine Lösung des Problems darstellt, eine lineare Abbildung zu finden, die das Verhältnis der Varianzen außerhalb zu innerhalb der Klassen maximiert. Sie hat demnach, auch ohne daß die Verteilungsvoraussetzung erfüllt sind, eine bzgl. der Trennungsgüte vorteilhafte Eigenschaft und ist in diesem Kontext als Fishersche Diskriminanzfunktion bekannt. Abbildung 3.1 zeigt beispielhaft zwei bedingte Dichten samt der dazugehörigen Gewinnwahrscheinlichkeit in Abhängigkeit eines Merkmals.

Die gesuchten Parameter können mittels der Beispiele gemäß des Maximum–Likelihood (ML) Ansatzes auf folgende Art geschätzt werden:

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_K &= \frac{1}{|I_K|}\sum_{i \in I_K} \boldsymbol{x}_i \\
\hat{\boldsymbol{\Sigma}}_K &= \frac{1}{|I_K|}\sum_{i \in I_K} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_K)'(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_K)
\end{aligned}$$

mit $I_K = \{i \mid y_i = K\}$ und bei gleichen Kovarianzen

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{|I_{\mathcal{G}}| + |I_{\mathcal{V}}|}\sum_{K \in \{\mathcal{V},\mathcal{G}\}}\sum_{i \in I_K} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_K)'(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_K).$$

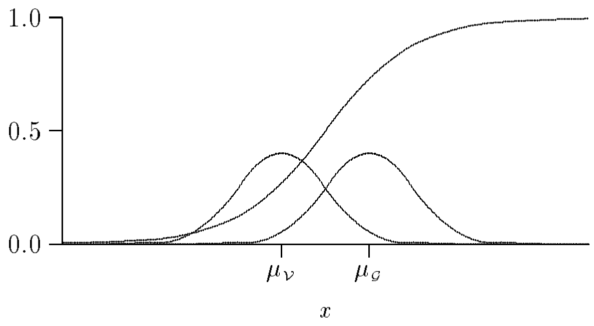


Abbildung 3.1: Bedingte Dichten und Gewinnwahrscheinlichkeit

3.2 Logistische Regression

Die bedingte Gewinnwahrscheinlichkeit $P(\mathcal{G} \mid \boldsymbol{x})$ soll im hier betrachteten Modell von einer Linearkombination der x_i abhängen. Im folgenden sei $X_1 = 1$, um auch konstante Verschiebungen modellieren zu können. Der einfache Ansatz $P(\mathcal{G} \mid \boldsymbol{x}) = \boldsymbol{x\beta}$ für einen Parameter–Spaltenvektor $\boldsymbol{\beta}$ ist unbrauchbar, da $\boldsymbol{x\beta} \in [0,1]$ nicht gewährleistet ist. Durch die Benutzung einer sogenannten Link–Funktion $g : (0,1) \to \mathbb{R}$ kann obige Forderung gemäß $g(P(\mathcal{G} \mid \boldsymbol{x})) = \boldsymbol{x\beta}$ erfüllt werden. Trägt man die empirisch geschätzte bedingte Gewinnwahrscheinlichkeit in Abhängigkeit eines Merkmals ab, so ergibt sich typischerweise der in Abbildung 3.2 gezeigte Verlauf. Da die Wahrscheinlichkeit gewöhnlich monoton in den Merkmalen ist, wird man $\lim_{x\to 0+} g(x) = -\infty$, $\lim_{x \to 1-} g(x) = +\infty$ und die Monotonie von g verlangen. Die Link–Funktion $g(t) = \operatorname{logit}(t) := \log(t/(1-t))$ hat diese Eigenschaften und auch eine andere, die sie auszeichnet: Sind die Merkmale (ohne Betrachtung von $X_1 = 1$) in den Klassen multivariat normalverteilt mit gleicher Kovarianzmatrix, so kann bei der im vorherigen Abschnitt besprochenen Diskriminanzanalyse $\operatorname{logit}(P(\mathcal{G} \mid \boldsymbol{x}))$ als $\boldsymbol{x\beta}$ dargestellt werden [vgl. AGRESTI (1990) S. 127].

Mit der Wahl $g = \operatorname{logit}$ gilt für die bedingte Gewinnwahrscheinlichkeit

$$P(\mathcal{G} \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{x\beta})}{1 + \exp(\boldsymbol{x\beta})} = \frac{1}{1 + \exp(-\boldsymbol{x\beta})}.$$

Sie hat demnach die gleiche Gestalt wie im letzten Abschnitt. Im Unterschied zu dortigen Modellannahmen wird aber bei der hier vorgestellten logistischen Regression nicht gefordert, daß die Merkmale in den Klassen

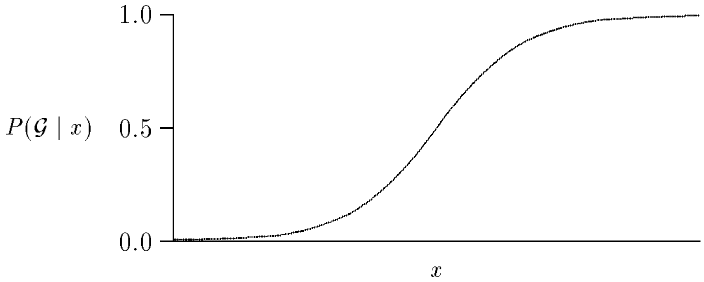


Abbildung 3.2: Typischer Verlauf der Gewinnwahrscheinlichkeit

multivariat normalverteilt sind. So sind sogar Merkmale mit nur wenigen Ausprägungen zulässig. Der Parametervektor $\boldsymbol{\beta}$ kann wiederum mit Hilfe des ML–Ansatzes geschätzt werden. Hierbei ergibt sich die Notwendigkeit zur Lösung eines nichtlinearen Gleichungssystems. Im folgenden wird ein bekannter Lösungsansatz vorgestellt, der auf dem Newton–Raphson Verfahren basiert.

Die Likelihood–Funktion $L(\boldsymbol{\beta})$ gibt die Wahrscheinlichkeitsdichte an, die in der Beispielmenge vorliegenden Realisation $\boldsymbol{y}$ der als stochastisch unabhängig angenommenen Zufallsvariablen Y_i unter Vorliegen des Parametervektors $\boldsymbol{\beta}$ zu erhalten. Um L zu maximieren, genügt die Betrachtung von $\log(L)$. Mit $\pi_i = 1/(1+\exp(-\boldsymbol{x}_i\boldsymbol{\beta}))$ und $\tilde{y}_i = 1$ für $y_i = \mathcal{G}$ und 0 sonst gilt dann

$$\begin{aligned}
\log(L(\boldsymbol{\beta})) &= \log\Big(\prod_{i=1}^{N} \pi_i^{\tilde{y}_i}(1-\pi_i)^{1-\tilde{y}_i}\Big) \\
&= \sum_{i=1}^{N} \tilde{y}_i \log \pi_i + (1-\tilde{y}_i)\log(1-\pi_i) \\
&= \sum_{i=1}^{N} \tilde{y}_i \log\Big(\frac{\pi_i}{1-\pi_i}\Big) + \log(1-\pi_i) \\
&= \sum_{j=1}^{n} \Big(\sum_{i=1}^{N} \tilde{y}_i x_{ij}\Big)\beta_j - \sum_{i=1}^{N} \log\Big[1+\exp\Big(\sum_{j=1}^{n} x_{ij}\beta_j\Big)\Big]
\end{aligned}$$

Diese Funktion ist zweimal differenzierbar, ist laut Wedderburn (1976) bis auf gewisse Grenzfälle strikt konkav und besitzt eine eindeutige Maximalstelle. Diese kann daher mit Hilfe der Newton–Raphson Methode folgendermaßen iterativ gefunden werden:

$$\hat{\boldsymbol{\beta}}^{(t+1)} = \hat{\boldsymbol{\beta}}^{(t)} - (\boldsymbol{H}^{(t)})^{-1}\boldsymbol{q}^{(t)},$$

wobei

$$\begin{aligned}
q_j^{(t)} &= \frac{\partial L}{\partial \beta_j}(\hat{\boldsymbol{\beta}}^{(t)}) &&= \sum_{i=1}^{N}(\tilde{y}_i - \hat{\pi}_i^{(t)})x_{ij} \\
h_{jk}^{(t)} &= \frac{\partial^2 L}{\partial \beta_j\ \partial \beta_k}(\hat{\boldsymbol{\beta}}^{(t)}) &&= \sum_{i=1}^{N} -x_{ij}x_{ik}\hat{\pi}_i^{(t)}(1-\hat{\pi}_i^{(t)})
\end{aligned}$$

und

$$\hat{\pi}_i^{(t)} = \frac{1}{1 + \exp\Big(-\sum_{j=1}^{n} x_{ij} \hat{\beta}_j^{(t)} \Big)}$$

ist. Mit diesen Größen, $\boldsymbol{\Delta} = \mathbf{diag}\big[\hat{\pi}_i^{(t)}(1 - \hat{\pi}_i^{(t)})\big]$ und der aus den $\boldsymbol{x}_i$ gebildeten $(N \times n)$–Matrix $\boldsymbol{X}$ erhält man

$$\hat{\boldsymbol{\beta}}^{(t+1)} = \hat{\boldsymbol{\beta}}^{(t)} + (\boldsymbol{X}'\boldsymbol{\Delta}\boldsymbol{X})^{-1}\boldsymbol{X}'(\tilde{\boldsymbol{y}} - \hat{\boldsymbol{\pi}}^{(t)})$$

und hieraus

$$\hat{\boldsymbol{\beta}}^{(t+1)} = (\boldsymbol{X}'\boldsymbol{\Delta}\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\Delta}\,\boldsymbol{z}^{(t)}$$

mit

$$z_i^{(t)} = \log\Big(\frac{\hat{\pi}_i^{(t)}}{1 - \hat{\pi}_i^{(t)}}\Big) + \frac{\tilde{y}_i - \hat{\pi}_i^{(t)}}{\hat{\pi}_i^{(t)}(1 - \hat{\pi}_i^{(t)})}.$$

Die letzte Darstellung zeigt nun eine naheliegende Möglichkeit für die Wahl des Startvektors auf. $\boldsymbol{z}^{(0)}$ kann beispielsweise aus der Setzung

$$\hat{\pi}_i^{(0)} = \begin{cases} 0.01 & \tilde{y}_i = 0 \\ 0.99 & \tilde{y}_i = 1 \end{cases}$$

berechnet werden. Typischerweise liegt bereits nach wenigen Schritten die ML–Schätzung $\hat{\boldsymbol{\beta}}$ in einer hohen Genauigkeit vor, da das Verfahren quadratisch konvergent ist und in dem hier betrachteten Fall auch robust gegenüber der Wahl des Startvektors.

3.3 Wahl der Beispiele

Zunächst muß geklärt werden, welche Stellungen zur Parameterschätzung herangezogen und wie sie klassifiziert werden. Schon bei der Schätzung der Tabelleneinträge wurde im vorherigen Kapitel darauf hingewiesen, möglichst nicht nur Stellungen aus guten Partien zu betrachten. Idealerweise sollten zur Schätzung die Positionen benutzt werden, die auch im späteren Einsatz zu bewerten sind. In Verbindung mit den statistischen Ansätzen blieben dahingehende Experimente aber ohne Erfolg: Als Beispielpositionen wurden hierbei Blattstellungen gewählt, die in Baumsuchen in Turniertiefe bewertet wurden. Diese wurden ihrerseits mittels einer Baumsuche heuristisch klassifiziert und die so erzeugten Beispiele zur

Schätzung benutzt. Hierbei trat das Problem auf, daß derartig gewählte Positionen bzgl. Gewinn und Verlust ungleichmäßig verteilt sind. Wie in Abschnitt 3.6 ausgeführt, läßt sich unter vereinfachenden Modellannahmen zeigen, daß im Normalfall ab einer gewissen Tiefe für den dort anziehenden Spieler deutlich mehr gewonnene als verlorenen Stellungen existieren. Diese Eigenschaft kann man leicht motivieren: „Wer den letzten Zug in einer beliebigen Zugfolge macht, begeht häufig den letzten spielentscheidenden Fehler“. Da hier ML–Schätzer benutzt werden, reproduziert sich diese Asymmetrie und es kommt zu starken Oszillationen der Stellungsbewertung in Abhängigkeit von der Tiefe. Durch Zusammenfassen von Beispielen jeweils zweier aufeinanderfolgender Steinanzahlen konnte zwar die Symmetrie erzwungen werden, die Spielstärke der mit den geschätzten Parametern ausgestatteten Programme war aber enttäuschend. Weitere Untersuchungen ergaben, daß sich die Dichtefunktionen der Merkmale bei Betrachtung der Blattspielfelder stark von denen bei Betrachtung der Partiestellungen unterscheiden: Die Erwartungswerte haben im ersten Fall einen deutlich kleineren Abstand als im zweiten. Es hat den Anschein, daß durch das überwiegende Vorkommen von Stellungen, die aus beiderseits mittelmäßigen Zügen entstehen, die scharfe Trennung zwischen den Klassen erschwert wird.

Eine pragmatische Möglichkeit zur Wahl der Beispiele für die Schätzung von Parametern liegt in der Betrachtung vieler Partien zwischen starken Gegnern. Notwendig für eine gute Trennbarkeit der Beispielpositionen bzgl. Gewinn und Verlust anhand von Merkmalen ist eine hohe Klassifikationsgüte der Beispiele und eine große klassenübergreifende Merkmals–Varianz. Im Abschnitt 2.3 wurde angesprochen, daß die Schätzung der Werte der Musterinstanzen nur anhand von Positionen aus Expertenspielen problematisch ist. Auch hier gibt es diesbezüglich Schwierigkeiten, denn die Varianz wichtiger Merkmale ist klein, da Experten versuchen, sie während eines Spiels zu maximieren. Speziell bei Othello hat sich weiterhin herausgestellt, daß sogenannte Meisterspiele nicht halten, was sie versprechen. Eine Computeranalyse von Turnierpartien zwischen menschlichen Gegnern ergab eine derart hohe Fehlerrate, daß die Spiele für das Schätzen von Parametern selbst nach einer Endspielkorrektur ungeeignet sind.

Von Lee & Mahajan (1988) wurden daher Spiele von älteren Versionen ihres spielstarken Programms BILL erzeugt, indem jeweils zunächst 20 zufällige Halbzüge gemacht und die Spiele danach innerhalb der normalen Turnierspielzeit beendet wurden. Die auftretenden Positionen wurden gemäß des jeweiligen Spielausgangs klassifiziert und zum Schätzen der Parameter benutzt. Möchte man auch für die frühe Eröffnungsphase Parameter bestimmen, so bietet sich an, alle Eröffnungen einer bestimmten Länge zu erzeugen und sie durch zwei starke Programme zuende spielen zu lassen. Dieses Vorgehen führt darüberhinaus, wie auch die von Lee & Mahajan (1988) eingesetzte Methode, auf eine natürliche Weise zu einer großen klassenübergreifenden Varianz der Merkmale. In einem Zeitraum von zwei Jahren wurden derart über 50.000 Spiele von verschiedenen Versionen von Igor Đurđanovićs Programm REV und LOGISTELLO generiert.

Die Klassifikation von Stellungen durch Rückpropagierung der Spielresultate ist problematisch, da die Güte zum Beginn der Partien hin aufgrund von Spielfehlern abnimmt. Bei Othello kann dieser Effekt durch eine frühe Endspielsuche abgeschwächt werden. So sind alle Beispielpartien ab 44 Steinen auf dem Brett bezüglich Gewinn korrekt gespielt. Bedenkt man, daß im späten Mittelspiel unter Turnierbedingungen regelmäßig die Brute–Force–Suchtiefe 11 erreicht wird, so werden schon Stellungen mit 33 Steinen mittels Parametern bewertet, die auf korrekt klassifizierten Positionen beruhen. Darüberhinaus wurde die Spielesammlung in Abständen mit den jeweils aktuellen Versionen der Programme untersucht, um „offensichtliche" Fehler älterer Programme zu korrigieren. Durch die Betrachtung aller Eröffnungen der Länge 7 und die Korrektur von Partien wiederholten sich viele Spielanfänge. Darum wurden Positionen nicht aufgrund des jeweiligen Spielausgangs, sondern gemäß des NegaMax–Spielwertes in dem aus allen Partien gebildeten Spielbaum klassifiziert, um die Klassifzierungsgüte für Stellungen zu erhöhen, die in mehreren Spielen auftreten.

3.4 Vergleich

Ein Verfahren zur Bestimmung von Merkmalsgewichten wird man nicht schon im Vorfeld verwerfen, wenn nicht alle Voraussetzungen zur Anwendung exakt erfüllt sind — im praktischen Einsatz kann es sich nämlich trotzdem als wirkungsvoll erweisen. Dennoch soll nicht ganz auf die Betrachtung der Voraussetzungen verzichtet werden, da hierdurch vielleicht später erklärt werden kann, warum die eine Methode zu besseren Resultaten führt als eine andere, oder wo Verbesserungsmöglichkeiten bestehen.

Notwendig für das Vorliegen einer multivariaten Normalverteilung in den Klassen ist die univariate Normalverteilung der einzelnen Merkmale. Da der Grundraum endlich ist, kann diese Forderung natürlich nur approximativ erfüllt werden. Graphische Darstellungen können helfen, sich schnell einen Überblick hinsichtlich der Dichtefunktionen zu verschaffen. Ergeben sich hierbei Anzeichen, daß sich Merkmalsverteilungen deutlich von einer geforderten Normalverteilung unterscheiden, so wird man nach Alternativen für diese Merkmale suchen müssen. Die Abbildung 3.3 und 3.4 zeigen exemplarisch die empirischen Dichten für das (2×4)–Eckenmuster und die approximierte aktuelle Mobilitätsdifferenz bei 40 Steinen. Man erkennt, daß die Merkmale näherungsweise normalverteilt sind. Bei näherer Betrachtung fällt jedoch auch auf, daß die Dichten für das Eckenmuster unsymmetrisch sind und daß die Mobilitätsdifferenz fast dreiecksverteilt ist. Allgemein wird man bei Merkmalen, die sich additiv aus mehreren Untermerkmalen zusammensetzen, wegen des zentralen Grenzwertsatzes eine gute Approximation einer Normalverteilung erwarten können. Dieses ist letztlich auch der Grund dafür, daß in der Praxis die Normalverteilungsannahme oft gerechtfertigt ist.

Bei der logistischen Regression wird vorausgesetzt, daß die bedingte Gewinnwahrscheinlichkeit gemäß der logit–Funktion von einer Linearkombination der Merkmale abhängt. Hierfür kann man auch eine leicht zu visualisierende notwendige Bedingung angeben: Die Einzelmerkmale müssen bezüglich der Wahrscheinlichkeit ein logit–Verhalten zeigen. Die beiden Abbildungen verdeutlichen, daß auch diese Bedingung näherungsweise für die Beispielmerkmale erfüllt ist. Hierbei ist zu bemerken, daß der fast lineare Verlauf der Wahrscheinlichkeit in der Umgebung von Null im Fall der Mobilitätsdifferenz mittels eines betragsmäßig kleinen Koeffizienten modelliert werden kann, da die Umkehrfunktion von logit um den Nullpunkt einen fast linearen Verlauf hat.

Neben den Musterwerten sind auch die Gewichte der Merkmale abhängig von der Spielphase. Deshalb wurden auch hier die Beispiele gemäß der Steinanzahl gruppiert und für die Schätzung eines Parametersatzes jeweils zwei benachbarte Gruppen zusammengefaßt, um die Anzahl gewonnener und verlorener Stellungen im Hinblick auf die ML–Schätzung bei der logistischen Regression und die Annahme gleicher apriori–Wahrscheinlichkeiten bei der Diskriminanzanalyse auszugleichen.

Ausgehend von einer Menge von Merkmalen stellt sich die Frage, welche daraus in das jeweilige Modell aufzunehmen sind. Im vorherigen Kapitel wurden Kriterien besprochen, die eine Vorauswahl in Frage kommender Merkmale ermöglichen. Hinsichtlich der späteren Auswertung in einer Baumsuche spielt hier neben der Güte auch die Auswertungsgeschwindig-

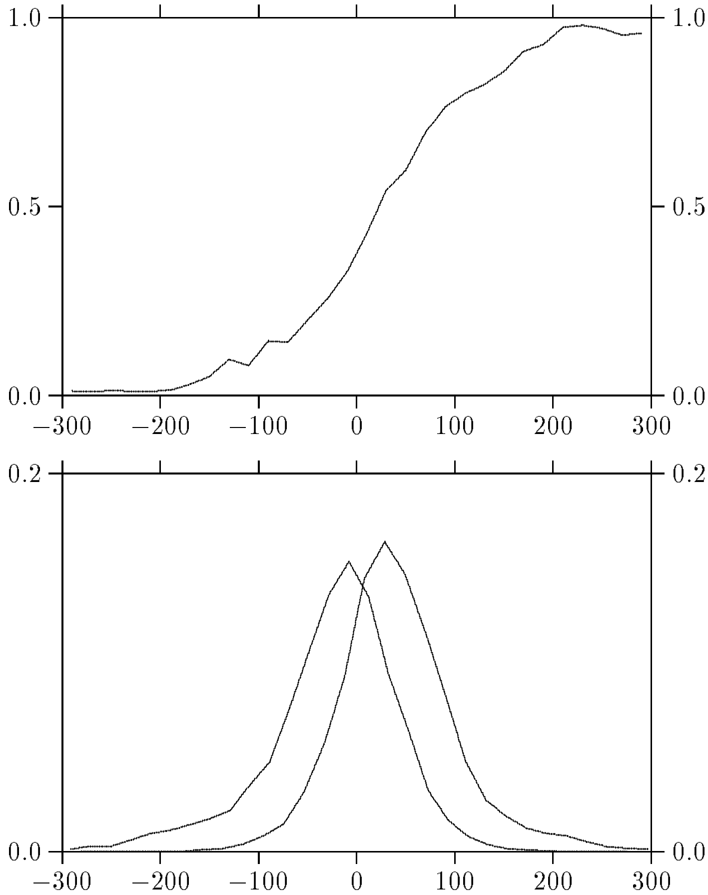


Abbildung 3.3:

Empirische Gewinnwahrscheinlichkeit und Dichten
für das (2×4)–Eckenmuster (30 Intervalle)

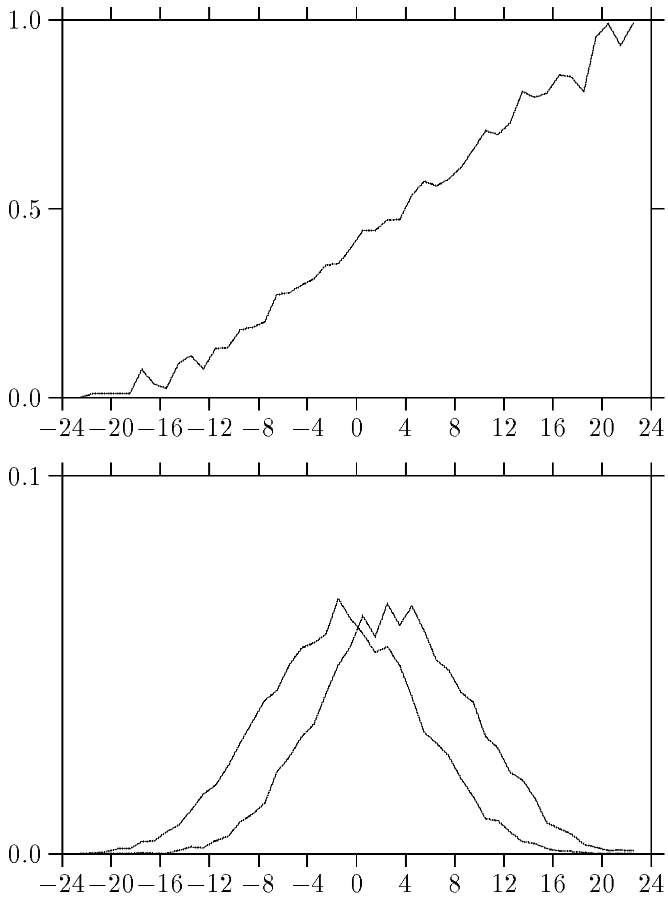


Abbildung 3.4:

Empirische Gewinnwahrscheinlichkeit und Dichten
für die approximierte aktuelle Mobilitätsdifferenz
(46 Intervalle)

keit der Merkmale eine Rolle. Unter diesem Aspekt wurden als Merkmale die Horizontal– und die Diagonalmuster, das (2×4)–Eckenmuster und die beiden Mobilitätsapproximationen gewählt. Um die Relevanz der Merkmale im Gesamtmodell vergleichen zu können, wurden die standardisierten Parameterschätzungen betrachtet, die sich aus den originalen Parameterschätzungen gemäß einer Normierung der jeweiligen Merkmalsvarianz auf Eins ergeben. Abbildung 3.5 zeigt die Verläufe für die logistische Regression. Über alle Spielphasen hinweg stellte sich bei dieser Untersuchung heraus, daß die Diagonalmuster der Länge 3 und 4 im Gesamtmodell vernachlässigbar sind. Deutlich zu erkennen ist die Relevanz der Mobilitäten und Muster, in denen ein Eckfeld vorkommt, im Vergleich zu den restlichen Merkmalen. Es zeigt sich aber auch, daß aus der Sicht des anziehenden Spielers positiv definierte Merkmale (wie die Muster) negative Parameter erhalten können. Diese Beobachtung konnte auch bei der Diskriminanzanalyse gemacht werden und wird in Kürze ausführlicher diskutiert.

Um die Spielstärke der verschiedenen Programme vergleichen zu können, wurden eine Reihe von Turnieren gespielt. Zu insgesamt 35 Spielfeldern aus der Eröffnungsphase wurden jeweils Hin– und Rückspiel unter den üblichen Turnierbedingungen — 30 Minuten pro Spieler — gemacht. Bei der Wahl der Spielfelder wurde auf deren Ausgeglichenheit geachten, da anderenfalls der Gewinn selbst bei kleinen Spielstärkeunterschieden nur von der Spielfarbe abhängt und daher ein Vergleich der Spielstärke nur schwer möglich ist. Konkret suchte hierzu ein Programm in der Menge der Beispielpartien nach (bzgl. der derzeit aktuellen Bewertungsfunktion) in Tiefe 10 relativ ausgeglichenen Positionen mit 12 Steinen. Sie sind im Anhang C aufgeführt. An der ersten Turniersequenz nahmen folgende Spieler teil:

FISHER geht von normalverteilten Merkmalen mit gleichen Kovarianzmatrizen in den Klassen aus und benutzt deshalb die auf S. 25 definierte lineare Diskriminanzfunktion. Die bedingte Wahrscheinlichkeit muß hieraus während der Baumsuche nicht berechnet werden, da sie streng monoton von der Diskriminanzfunktion abhängig ist. Dieses gilt auch für die anderen Bewertungsfunktionen.

QUAD nimmt normalverteilte Merkmale mit unterschiedlichen Kovarianzmatrizen an und benutzt daher die aus S. 24 gegebene quadratische Diskriminanzfunktion. Dieses führt hier zu einer um ca. 25% kleineren Suchgeschwindigkeit gemessen in Knoten pro Sekunde im Vergleich zu FISHER.

QUAD* hat die gleiche Bewertungsfunktion wie QUAD, bekommt aber 1/3 mehr Zeit, um das Turnierergebnis unabhängig von der Suchgeschwindigkeit zu machen. Bei Spielen von QUAD* wird aus offensichtlichen Gründen nicht auf gegnerische Zeit gerechnet.

LOG macht keine Verteilungsannahmen, gibt jedoch die Form der Abhängigkeit der unter der Merkmalsausprägung bedingten Gewinnwahrscheinlichkeit vor. Wie FISHER begnügt er sich mit einer linearen Kombinationsfunktion.

Die Turnierergebnisse werden durch Tabelle 3.1 zusammengefaßt, wobei in den Resultatspalten die Anzahlen gewonnener, unentschiedener und verlorener Spiele vermerkt sind. In Abhängigkeit von der Gewinnquote kann die Hypothese, daß ein Spieler stärker spielt als ein anderer, mittels statistischer Tests untermauert bzw. verworfen werden. Eine genauere Betrachtung, die im Abschnitt 3.6 ausgeführt ist, zeigt, daß bei einer Spielfeldanzahl von 35 bei Gebrauch konservativer statistischer Methoden erst bei Gewinnquoten über 64% von einer signifikant größeren Spielstärke zum Niveau 5% gesprochen werden kann. Selbst bei der Betrachtung von 100 Spielfeldern wäre immer noch eine Gewinnquote von mindestens 58% notwendig. Somit müssen die hier gemachten Bemerkungen aufgrund knapper Turnierergebnisse und Rechenzeitbeschränkungen als Trendanalysen angesehen werden.

Augenfällig ist der Vorteil der linearen Kombinationsfunktionen gegenüber der quadratischen bei gleicher Rechenzeit. In der Tat stellte sich heraus, daß die Kovarianzmatrizen bzgl. der beiden Klassen sich komponentenweise kaum unterscheiden. Somit sind die Koeffizienten vor den quadratischen Termen betragsmäßig sehr klein und die quadratische Funktion unterscheidet sich nur geringfügig von der Fisherschen Linearkombination,

Paarung	Resultat	Gewinnquote
LOG – QUAD	38 – 11 – 21	62.1%
FISHER – QUAD	39 – 5 – 26	59.3%
LOG – FISHER	36 – 8 – 26	57.1%
LOG – QUAD*	35 – 8 – 27	55.7%
QUAD* – FISHER	33 – 5 – 32	50.7%

Tabelle 3.1: Turnierresultate

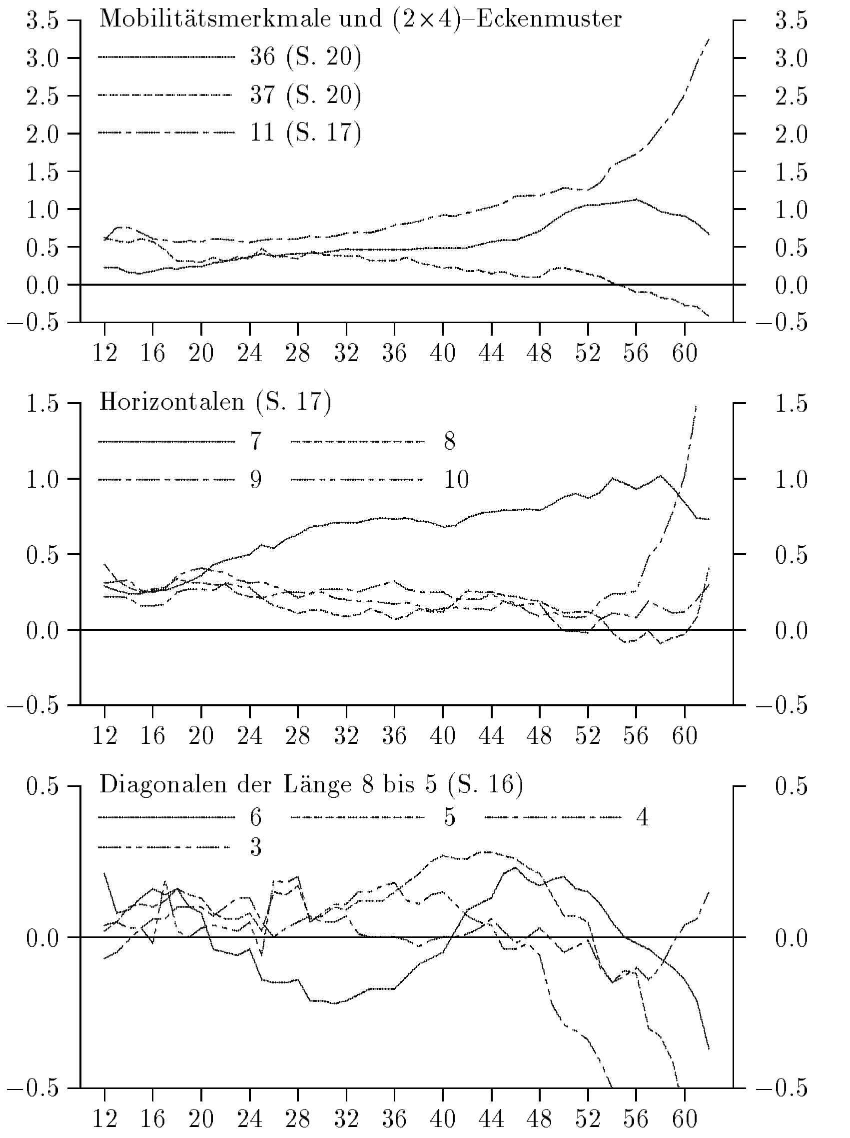


Abbildung 3.5: Standardisierte Parameterschätzungen

führt aber zu einer um ca. 25% kleineren Suchgeschwindigkeit. Kompensiert man diesen Nachteil durch eine größere Rechenzeit, so ergibt sich eine mit FISHER vergleichbare Spielstärke.

Bei der Betrachtung der Programmausgaben fiel bei allen Spielern auf, daß sie sich oft in der letzten Iteration für einen anderen Zug als vorher berechnet entschieden und dieses oft nicht zu ihrem Vorteil, wie anschließende Untersuchungen ergaben. Dieses Verhalten mag einerseits in der Schätzung der Musterwerte begründet liegen, die zu Unstabilitäten der Bewertung führen kann. Andererseits tragen auch negative Parameter vor für den anziehenden Spieler positiven Merkmalen hierzu bei. Dieser Effekt liegt in Merkmalskorrelationen begründet und ist nicht wünschenswert, da er die Bewertungsfunktion unrobust gegenüber etwaig veränderten Korrelationen bei den Blattstellungen macht und darüberhinaus die betreffenden Merkmale bei der großen Vielfalt in einer Baumsuche bewerteter Positionen minimiert statt maximiert werden. Da das Vorkommen solcher negativer Parameter auch nicht nur auf Merkmale kleiner Relevanz beschränkt ist, wurde ein Verfahren entwickelt, daß die Konsistenz der Parameter in dem Sinne sicherstellt, daß die Vorzeichen der Parameter im multivariaten Modell gleich denen in den jeweiligen univariaten Modellen sind. Hierzu kann im Rahmen der logistischen Regression eine ML–Schätzung unter den Nebenbedingungen $\beta_i \geq 0$ oder $\beta_i \leq 0$ für gewisse i berechnet werden. Wie in Abschnitt 3.6 näher begründet wird, kann dieses approximativ einfach dadurch geschehen, daß in einem iterativen Prozeß ML–Schätzungen berechnet und die Merkmale aus dem Modell genommen werden, deren Parameter nicht das gleiche Vorzeichen wie im univariaten Modell erhielten, um anschließend erneut eine ML–Schätzung zu berechnen, bis schließlich alle verbliebenen Merkmale konsistente Parameter besitzen. Auf diese Weise berechnete Parametersätze (des Spielers LOG$^+$) zeigt Abbildung 3.6. Beachtenswert ist hier der Vorzeichenwechsel des Parameters für die potentielle Mobilität, der auch im univariaten Modell zu verzeichnen ist. Außerdem sind im Endspiel ab 54 Steinen nur noch 5 oder 6 von 12 Merkmalen im Modell, und das Hauptdiagonal–Muster spielt nur noch eine untergeordnete Rolle im Gesamtmodell.

Das oben geschilderte Verfahren kann auch bei der Diskriminanzanalyse angewendet werden — dies führte zum Spieler FISHER$^+$. Eine angepaßte Version der quadratischen Kombinationsfunktion wurde aufgrund der weiterhin bestehenden Laufzeitverluste nicht betrachtet. Um einen Eindruck von der Spielstärke der neuen Spieler zu bekommen, wurden

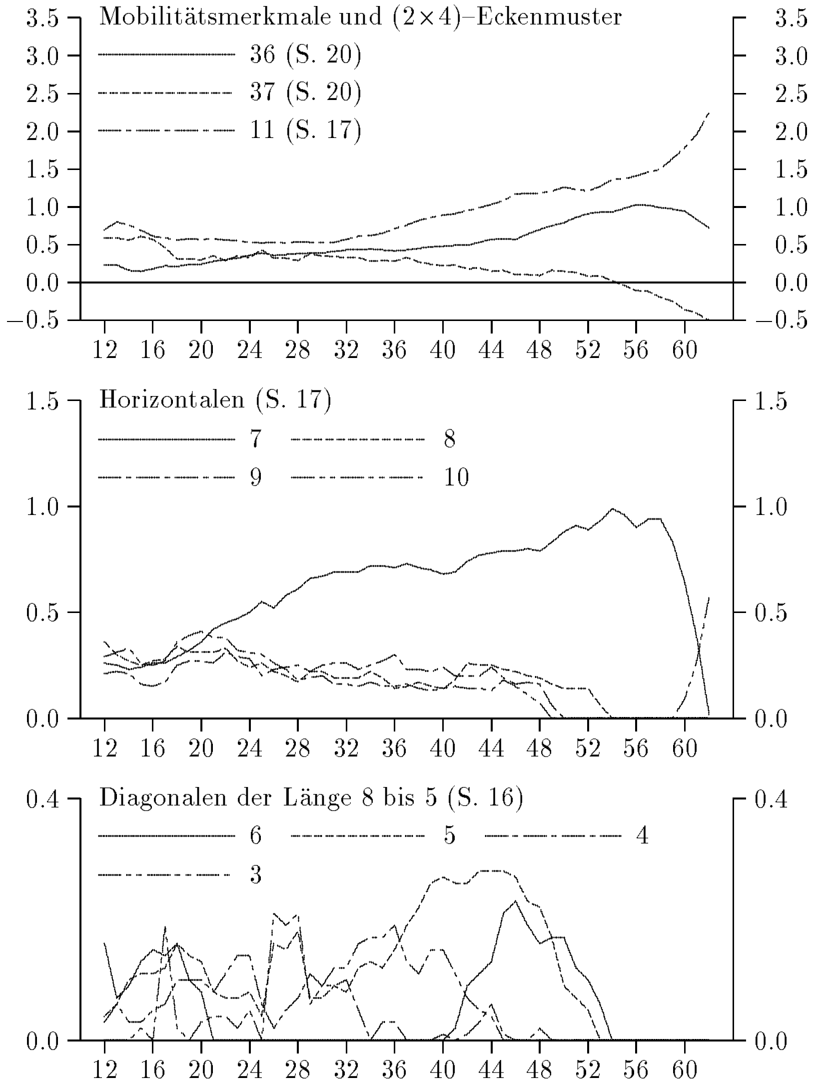


Abbildung 3.6:
Standardisierte Parameterschätzungen unter Nebenbedingungen

wiederum eine Reihe von Turnieren gespielt, deren Resultate die folgende Tabelle zeigt:

Paarung	Resultat	Gewinnquote
LOG$^+$ – QUAD	$39 - 9 - 22$	62.1%
LOG$^+$ – FISHER$^+$	$39 - 5 - 26$	59.3%
LOG$^+$ – QUAD*	$35 - 9 - 26$	56.4%
LOG$^+$ – FISHER	$36 - 5 - 29$	55.0%
LOG$^+$ – LOG	$33 - 5 - 32$	50.7%

Tabelle 3.2: Turnierresultate von LOG$^+$

Aufgrund dieser Ergebnisse benutzt LOGISTELLO die Bewertungsfunktion von LOG$^+$. Diese hat darüberhinaus durch die Beschränkung auf konsistente Parameter in dem Sinne an Stabilität gewonnen, daß sich sehr viel seltener gefundene Züge jeweils bei Erhöhung der Suchtiefe um Eins abwechseln. Hierdurch wird die Suchzeit bei Verwendung der Nullfenstersuche verringert, da weniger Wiederholungssuchen notwendig sind.

3.5 Diskussion

In diesem Kapitel sind im Hinblick auf die Vergleichbarkeit von Bewertungen aus unterschiedlichen Spielphasen drei statistische Ansätze zur Bestimmung von Merkmalsgewichten verglichen worden.

Als Grundlage für die Parameterschätzungen diente eine große Menge von nach Gewinn und Verlust klassifizierten Beispielpositionen, deren Wahl großen Einfluß auf die Güte der Bewertung hat. Wichtig ist hierbei vor allem eine große klassenübergreifende Varianz der Merkmale, die durch geeignete Maßnahmen bei der Erzeugung der Beispiele gewährleistet werden kann.

Bei Vergleichen der Spielstärke mittels einer Folge von Spielen stellte sich heraus, daß die Verwendung der quadratischen Diskriminanzfunktion im Vergleich zur linearen nicht notwendigerweise zu einer größeren Spielstärke führt. Im direkten Vergleich zur Diskriminanzfunktion von Fisher zeigte sich, daß der Nachteil der quadratischen Funktion wohl auf den hier beobachteten Suchgeschwindigkeitsrückgang von ca. 25% zurückzuführen ist. Bei der Benutzung von weniger oder Rechenzeit–intensiveren

Merkmalen könnten die Ergebnisse anders ausfallen. Vor dem Gebrauch der quadratischen Diskriminanzfunktion sollten daher die Kovarianzmatrizen in den Klassen verglichen werden: Sind diese identisch, so kann auf die quadratischen Terme verzichten werden. Die von Lee & Mahajan (1988) angegebene Motivation bedarf daher einer Verfeinerung, denn eine bestehende Korrelation zwischen den Merkmalen begründet alleine noch nicht die Notwendigkeit der Benutzung einer quadratischen Kombinationsfunktion. In derselben Arbeit wurde der große Spielstärkezuwachs nach Einführung der quadratischen Kombinationsfunktion anhand eines Turniers gegen ein Programm mit einer linearen Funktion demonstriert. Leider kam hierbei nur eine lineare Kombinationsfunktion mit handoptimierten Gewichten zum Einsatz und nicht — wie es naheliegend wäre — auch die lineare Diskriminanzfunktion von Fisher.

Neben der linearen Regression und der Diskriminanzanalyse ermöglicht auch die logistische Regression die Schätzung von Merkmalsgewichten für Bewertungsfunktionen mit einer Spielphasen–unabhängigen Interpretation. Sie macht keine Verteilungsannahmen über die Merkmale und ist somit auch geeignet bei Benutzung von Merkmalen mit nur wenigen Ausprägungen. Dem Nachteil der Rechenzeit–intensiven Parameterbestimmung durch Lösen nichtlinearer Gleichungssysteme steht die hohe Güte der Bewertungsfunktion im Vergleich zu den anderen Ansätzen gegenüber: Die Resultate einer Folge von Turnieren zeigten einen Spielstärkevorteil, der wenngleich nicht zum Niveau 5% statistisch signifikant aber relativ deutlich war.

Bemerkenswert ist das beobachtete Auftreten negativer Parameterschätzung für den anziehenden Spieler positiv definierte Merkmale wie beispielsweise die Muster, das mit Merkmalskorrelationen begründet werden kann. Im Rahmen der Spielbaumsuche sind solche negativen Gewichte nicht aktzeptabel, da hierdurch bei der Vielfalt der untersuchten Stellungen konsequent solche Positionen bevorzugt werden, die die jeweiligen Merkmale minimieren anstatt sie zu maximieren. Aus diesem Grunde ist ein Verfahren entwickelt worden, daß bei der Schätzung die Vorgabe der Parametervorzeichen ermöglicht, um somit obigen Effekt auszuschließen. Die Suche profitiert hiervon, da die Bewertung hierdurch stabiler wird.

3.6 Theoretische Details

Anteil der gewonnenen Stellungen auf einer Tiefe

Unter vereinfachenden Modellannahmen wird in diesem Abschnitt der Anteil für den anziehenden Spieler gewonnener Stellungen auf einer Tiefe bestimmt. Hierbei stellt sich heraus, daß dieser im Normalfall unabhängig vom Spielwert der betrachteten Wurzelstellung für wachsende Tiefen konvergiert und größer als $1/2$ ist.

Zur Vereinfachung der Analyse wird im folgenden von uniformen Spielbäumen mit Verzweigungsgrad w ausgegangen. Weiterhin sollen bis zu einer Tiefe d keine Terminalstellungen vorkommen und in gewonnenen Positionen jeweils l von den w möglichen Zügen zum Verlust führen. Für die Anzahl W_d der gewonnenen Stellungen bzw. die Anzahl L_d der verlorenen Stellungen auf Tiefe $d > 0$ gilt dann:

$$\begin{aligned} W_d &= w^d - L_d \\ L_d &= W_{d-1} \cdot (w - l) \end{aligned}$$

Hieraus erhält man durch Einsetzen die folgende Rekursionsgleichung

$$L_d = w^{d-1} \cdot c - c \cdot L_{d-1}$$

für $d > 0$ und $c := w - l$, welche die Lösung

$$L_d = \frac{(-c)^{d+1} - w^{d+1}}{c + w} + w^d + (-c)^d \cdot L_0$$

besitzt. Somit gilt

$$W_d = \frac{w^{d+1} - (-c)^{d+1}}{c + w} - (-c)^d \cdot (1 - W_0),$$

und

$$\frac{W_d}{w^d} = \frac{w}{c + w} + (\frac{w}{c + w} + W_0)(-c/w)^d$$

strebt für wachsende d bei $l \neq 0$ gegen $\dfrac{w}{c + w} = \dfrac{1}{2 - l/w} > 1/2$.

Betrachtet man beispielsweise ein fiktives Spiel, bei dem

- es bis auf Terminalpositionen in jeder Stellung 36 Züge gibt, von denen in gewonnenen Positionen immer 25 zum Verlust führen,
- ausgehend von einer Wurzelstellung erst in großen Tiefen Terminalpositionen existieren und
- Zugumstellungen nicht möglich sind,

so ergeben sich gemäß den obigen Ausführungen folgende rasch konvergierende (gerundete) Anteile der gewonnenen Stellungen in Abhängigkeit von der Tiefe d:

d	W_d/w^d $(W_0 = 0)$	W_d/w^d $(W_0 = 1)$
1	1.000000	0.694444
2	0.694444	0.787809
3	0.787809	0.759281
4	0.759281	0.767998
5	0.767998	0.765334
6	0.765334	0.766148
7	0.766148	0.765899
8	0.765899	0.765975

$$\frac{1}{2 - l/w} \doteq 0.765957$$

Spielstärkevergleich

Anhand der Gewinnquote, die sich aus einem Turnier ergibt, soll im folgenden die Stärke von zwei Spielern verglichen werden. Hierzu kann die Gewinnquote bzgl. eines Spielpaares bestehend aus Hin- und Rückspiel aus der Sicht eines Spielers mittels einer multinomial–verteilten Zufallsvariable X modelliert werden, deren Bildmenge $\{0,\ 1/4,\ 1/2,\ 3/4,\ 1\}$ ist. Mit dieser Definition wird der Spieler, dessen Paar–Gewinnquote X angibt, genau dann stärker als der andere genannt, wenn $E(X) > 1/2$ gilt. Diese Hypothese gilt es nun mittels der vorliegenden Spielpaarresultate $x_1, \ldots, x_N$, die Realisationen einer Folge stochastisch unabhängiger, identisch wie X verteilter Zufallvariablen $X_1, \ldots, X_N$ sind, zu testen. Als Testgröße dient hier $\hat{\mu} = (1/N)\sum_{i=1}^{N} X_i$. Unter den Voraussetzungen $E(X_i) = \mu$ und $V(X_i) = \sigma^2 > 0$ besagt der Grenzwertsatz von Lindeberg–Lévy, daß $\hat{\mu}$ asymptotisch $\mathcal{N}(\mu, \sigma^2/N)$ verteilt ist. Die Hypothese $E(X) > 1/2$ wird angenommen, wenn $\hat{\mu}$ für genügend großes

N einen gewissen Wert c überschreitet, der aus der Forderung berechnet werden kann, daß die Wahrscheinlichkeit für die fälschliche Annahme klein sein soll — also $P_\mu(\hat{\mu} \geq c) = 1 - \Phi_{\mu,\sigma^2/N}(c) \leq \alpha$ für $\mu \leq 1/2$ und ein kleines α gilt. $\Phi_{\mu,\sigma^2/N}$ bezeichnet hierbei die Verteilungsfunktion einer normalverteilten Zufallsvariablen mit Erwartungswert μ und Varianz σ^2/N. Für $\alpha = 0.05$, ein genügend großes N und alle $\mu \leq 1/2$ ist die obige Ungleichung mit $c \geq 1/2 + 1.65 \cdot \sigma/\sqrt{N}$ erfüllt, wie ein paar kleine Umformungen und ein Blick in eine Tafel für $\Phi_{0,1}$ zeigen. Da die Varianz der X_i unbekannt ist, wird hier konservativ die maximal mögliche Varianz $1/4$ angenommen. Für verschiedene N gibt die nachstehende Tabelle die Werte für c an, bei denen die Hypothese bei kleineren Werten für $\hat{\mu}$ abgelehnt wird:

N	c
20	0.685
35	0.640
50	0.617
100	0.583
200	0.558

ML–Schätzung unter Nebenbedingungen

Das auf S. 37 beschriebene Verfahren zur Bestimmung der ML–Schätzung unter Nebenbedingungen wird durch folgende Aussage motiviert:

Lemma

Sei $L : \mathbb{R}^n \to \mathbb{R}$ eine stetige und strikt konkave Funktion mit

$$\lim_{\|\boldsymbol{\beta}\| \to +\infty} L(\boldsymbol{\beta}) = -\infty.$$

Dann existieren eindeutige Vektoren $\boldsymbol{\beta}' \in \mathbb{R}^n$ und $\boldsymbol{\beta}'' \in \mathbb{R}^n_{\geq 0}$ mit

$$L(\boldsymbol{\beta}') = \sup\{L(\boldsymbol{\beta}) \mid \boldsymbol{\beta} \in \mathbb{R}^n\}$$

und

$$L(\boldsymbol{\beta}'') = \sup\{L(\boldsymbol{\beta}) \mid \boldsymbol{\beta} \in \mathbb{R}^n_{\geq 0}\},$$

und es gilt

$$\exists i : \beta'_i < 0 \Rightarrow \left(\exists j : \beta'_j < 0 \land \beta''_j = 0\right).$$

Beweis

Aus der Voraussetzung folgt direkt die Existenz einer abgeschlossenen Kugel $K \subset \mathbb{R}^n$ mit Mittelpunkt $\mathbf{0}$, außerhalb der $L(\boldsymbol{\beta}) < L(\mathbf{0})$ gilt. Somit gibt es ein $\boldsymbol{\beta}' \in K$ mit $L(\boldsymbol{\beta}') \geq L(\boldsymbol{\beta})$ für alle $\boldsymbol{\beta} \in \mathbb{R}^n$ gemäß des Satzes von Weierstraß, denn K ist kompakt und L auf K stetig. Analog ergibt sich die Existenz einer Maximalstelle in $\mathbb{R}^n_{\geq 0}$, wenn man die kompakte Menge $K \cap \mathbb{R}^n_{\geq 0}$ benutzt. Die Eindeutigkeit der Maximalstellen folgt aus der strikten Konkavität von L. Sei nun also $\beta'_i < 0$ für ein i. Betrachtet man die Verbindungsstrecke von $\boldsymbol{\beta}'$ und $\boldsymbol{\beta}''$, deren Punkte $\boldsymbol{\beta}(\lambda)$ die Darstellung $(1-\lambda)\boldsymbol{\beta}' + \lambda\boldsymbol{\beta}''$ für $\lambda \in [0{,}1]$ haben, so existiert wegen der Konvexität von $\mathbb{R}^n_{\geq 0}$ ein $\lambda_0 \in (0{,}1]$ mit $\boldsymbol{\beta}(\lambda) \in \mathbb{R}^n_{\geq 0} \iff \lambda \geq \lambda_0$. Daher gibt es offenbar ein j mit $\beta(\lambda)_j < 0$ für $\lambda < \lambda_0$ und $\beta(\lambda_0)_j = 0$. Weil L konkav ist und $L(\boldsymbol{\beta}') \geq L(\boldsymbol{\beta}'')$ gilt, erhält man $L(\boldsymbol{\beta}(\lambda_0)) \geq L(\boldsymbol{\beta}'')$, womit $\boldsymbol{\beta}'' = \boldsymbol{\beta}(\lambda_0)$ gezeigt ist. Also existiert ein j mit $\beta'_j < 0$ und $\beta''_j = 0$, wie behauptet.

□

Gemäß Wedderburn (1976) erfüllt die log–Likelihoodfunktion für die logistische Regression typischerweise die Voraussetzungen des Lemmas. Das vorgestellte Verfahren wäre korrekt, wenn man aus $\beta'_i < 0$ $\beta''_i = 0$ folgern könnte. Dieses ist ohne weitere Voraussetzungen aber nicht möglich, wie einfache Gegenbeispiele zeigen. Wenn in jedem Iterationsschritt höchstens eine Komponente β'_i der aktuellen ML–Schätzung $\boldsymbol{\beta}'$ kleiner als Null ist, so folgt aus der Hilfsaussage $\beta''_i = 0$, und Merkmal i kann von der weiteren Betrachtung ausgeschlossen werden. In diesem Fall liefert das Verfahren also die restringierte ML–Schätzung. Sind dagegen zu einem Zeitpunkt mindestens zwei Komponenten kleiner als Null, so besteht die Möglichkeit, daß ein Parameter gleich Null gesetzt und somit auf das entsprechende Merkmal verzichtet wird, obwohl dies für die eingeschränkte ML–Schätzung nicht gilt. Da das Vorkommen negativer Parameter durch starke Merkmals–Korrelationen erklärt werden kann, ist der entstehende Güteverlust hinnehmbar, weil die im Modell verbleibenden Merkmale die verworfenen approximieren.

Kapitel 4

Selektive Suche

Von menschlichen Spielern weiß man, daß zum Auffinden eines guten Zugs nicht der Spielbaum in seiner vollen Breite untersucht werden muß. Aus Erfahrung werden viele Varianten von vorneherein nicht betrachtet, weil sie nicht erfolgversprechend sind. Der daraus resultierende Suchbaum ist schmal und kann daher bei gegebener Knotenanzahl sehr tief sein. Im Gegensatz hierzu wird vom ursprünglichen MiniMax–Algorithmus der Spielbaum bis zu einer gewissen Tiefe vollständig untersucht, wobei in der Praxis nur ruhige Blattstellungen bewertet werden und ggf. eine selektive Ruhesuche in den Blättern gestartet wird. Das Ziel dieses Kapitels ist es, Verfahren zu entwickeln, die der menschlichen Vorgehensweise näherkommen als die vollständige Suche, um dadurch die zur Verfügung stehende Rechenzeit besser zu nutzen.

4.1 Selektive heuristische Suche

Im folgenden wird ein Verfahren — ProbCut — vorgestellt, das in Verbindung mit dem $\alpha\beta$–Algorithmus ermöglicht, wahrscheinlich irrelevante Teilsuchbäume von einer tiefen Suche auszuschließen, um somit insgesamt die relevanten Varianten bei gleicher Zeit tiefer zu untersuchen. Da der $\alpha\beta$–Algorithmus um eine selektive Komponente erweitert werden soll, zeigt Abbildung 4.1 eine grobe Implementation einer NegaMax–Version, bei der der Spielwert immer aus der Sicht des anziehenden Spielers ermittelt wird. Aus Gründen der Übersichtlichkeit wird hierbei die globale Datenstruktur `pos` benutzt, die alle Stellungsinformationen enthält. Die Idee der Modifikation besteht nun darin, einer Suche in Tiefe d_b eine flachere Suche in Tiefe $d_s < d_b$ voranzustellen, den Spielwert v_b mit Hilfe von v_s zu schätzen und daraufhin zu entscheiden, ob mit einer vorgegebenen Sicherheit der Wert v_b außerhalb des Fensters `(alpha,beta)` liegt, es also nach der tiefen Suche zu einem Schnitt oder eine Ebene tiefer ausschließlich zu Schnitten kommt. Ist dabei die Antwort positiv, so kann man auf die tiefe Suche verzichten. Im anderen Fall hat man eine flache Suche zuviel investiert — im Vergleich zum Aufwand für die tiefe Suche fällt dies aber kaum ins Ge-

```
int AlphaBeta(int height, int alpha, int beta)
{
  int i, max, val;
  POSDELTA delta;

  if (Leaf(&pos, height))           /*     leaf-position? */
    return Eval(&pos);              /* yes => evaluate it */

/* location for the selective extensions */

  max = alpha;                      /* initialize maximum */

  for (i=0; i < pos.movenum; i++) {  /*   forall moves ... */

    Move(&pos, pos.move[i], &delta); /* make move and save */
                                     /*   changes in delta */
    val = -AlphaBeta(height-1, -beta, -max);    /* negamax */
    Undo(&pos, &delta);                 /* restore old pos */

    if (val > max) {
      if (val >= beta) return val;          /*      cutoff */
      max = val;                            /* new maximum */
    }
  }
  return max;
}
```

Abbildung 4.1: Der $\alpha\beta$-Algorithmus wicht. Zur Schätzung von v_b bietet sich ein einfaches lineares Modell der Gestalt $v_b = a \cdot v_s + b + e$ an mit $a, b \in \mathbb{R}$ und e als $\mathcal{N}(0, \sigma^2)$–verteilte Fehlervariable. Man erwartet hier $a \approx 1, b \approx 0$ und hofft, daß die Varianz σ^2 klein ist. Unter der Annahme der richtigen Modellierung ist $\hat{v}_b = a \cdot v_s + b$ ein erwartungstreuer Schätzer für v_b. Zu testen ist nun, ob folgende Bedingungen mindestens mit einer gegebenen Sicherheit gelten:

$$v_b \geq \beta \iff \hat{v}_b + e \geq \beta \iff -e/\sigma \leq (\hat{v}_b - \beta)/\sigma$$

Da $-e/\sigma$ $\mathcal{N}(0,1)$–verteilt ist (mit Verteilungsfunktion Φ), gilt $v_b \geq \beta$ mit Wahrscheinlichkeit mindestens p gdw. $(\hat{v}_b - \beta)/\sigma \geq \Phi^{-1}(p)$ zutrifft. Diese Bedingung ist äquivalent zu $v_s \geq (\Phi^{-1}(p) \cdot \sigma + \beta - b)/a$. Analog erhält man die Aussage, daß $v_s \leq \alpha$ mit Wahrscheinlichkeit mindestens p gilt gdw. $v_s \leq (-\Phi^{-1}(p) \cdot \sigma + \alpha - b)/a$ ist. Mithin interessiert nur, ob v_s größer bzw. kleiner als ein vorgegebener Wert ist. Diese Beobachtung führt direkt zu der in Abbildung 4.2 gezeigten Erweiterung. Bei Knoten in Höhe d_b wird zunächst mittels einer Nullfenstersuche in Tiefe d_s geprüft, ob v_b mit großer Wahrscheinlichkeit außerhalb des Fensters `(alpha,beta)` liegt. In diesem Fall wird der entsprechende Wert zurückgegeben und damit u.U. im Vergleich zu der tieferen Suche ein Fehler gemacht. Die Sicherheit läßt sich mittels `PERCENTILE` vorgeben. Wurde ursprünglich der Spielbaum vollständig in Tiefe $d \geq d_b$ durchsucht, so garantiert der neue Ansatz hier zumindest die Brute–Force–Tiefe $d - (d_b - d_s)$. Andererseits übersteigt bei gleicher Rechenzeit die maximal erreichte Tiefe $d + (d_b - d_s)$ nicht, denn hierbei wird mindestens vollständig Tiefe d untersucht. Wählt man `PERCENTILE` sehr groß, so ergibt sich — bis auf einen kleinen durch die Tiefe–d_s–Suche bedingten Laufzeitverlust — das ursprüngliche Verhalten, da nur selten frühzeitig geschnitten wird. Im anderen Extremfall tritt sehr oft ein Schnitt auf — die Suche in Tiefe d degeneriert zu einer Tiefe–$(d - (d_b - d_s))$–Suche. Bei Benutzung iterativer Vertiefung würde man also auch hier eine Spielstärke erwarten, die der des nichtselektiven Programms ähnelt.

Um die Parameter a, b und σ aus Beispieldaten schätzen zu können, wurden zunächst die Suchtiefen d_b und d_s festgelegt. Bei der Wahl ist folgendes zu beachten: Die Differenz $d_b - d_s$ sollte nicht zu groß sein und d_s selbst nicht zu klein, da sonst die Varianz der Fehlervariable groß wird und demnach bei gleicher Sicherheit seltener geschnitten werden kann. Andererseits darf die Differenz auch nicht zu klein sein, denn sie ist ein Maß für die Ersparnis, die erreicht werden kann. Weiterhin muß man bei der Wahl

```
#define PERCENTILE 1.5      /* i.e. p ca. 93%              */
#define DS 4                /* depth of shallow search     */
#define DB 8                /* check-distance to horizon   */
#define FLOAT2INT 100000.0  /* we are working on integers  */
                            /* so convert floats by multi- */
                            /* plying with FLOAT2INT       */

int AlphaBeta(int height, int alpha, int beta)
{
  ...
  if (height == DB) {
    int bound;

/* vb >= beta with prob. at least p? yes => cutoff */

    bound = round(FLOAT2INT*
              (+PERCENTILE * sigma + beta/FLOAT2INT - b) / a
            );
    if (AlphaBeta(DS, bound-1, bound) >= bound) return beta;

/* vb <= alpha with prob. at least p? yes => cutoff */

    bound = round(FLOAT2INT*
              (-PERCENTILE * sigma + alpha/FLOAT2INT - b) / a
            );
    if (AlphaBeta(DS, bound, bound+1) <= bound) return alpha;
  }
  ...
}
```

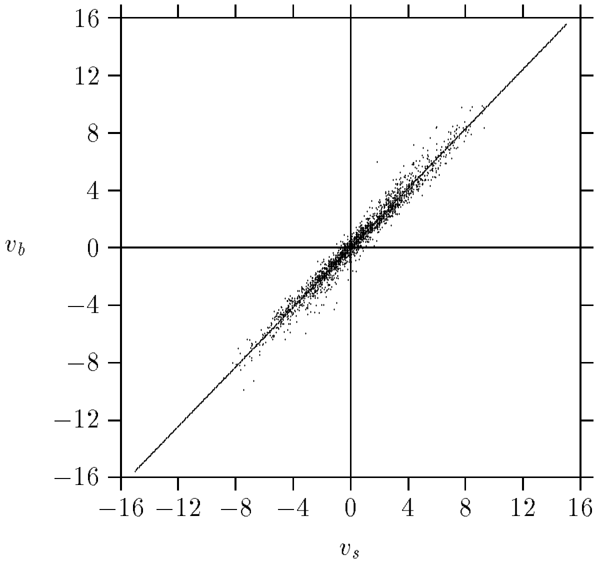
Abbildung 4.2: Die ProbCut–Erweiterung

von d_b berücksichtigen, daß diese Tiefe unter Turnierbedingungen auch erreicht wird und die Berechnung einer genügenden Anzahl von Beispielen zur Schätzung der Parameter in angemessener Zeit möglich ist. Eine Reihe von Experimenten führte schließlich zu der Wahl $d_s = 4$ und $d_b = 8$ für LOGISTELLO. Hierbei wurde beobachtet, daß die Varianz der Fehlervariable mit wachsender Steinanzahl zunimmt. Somit ist es zweckmäßig, die Parameter a, b und σ in Abhängikeit von der aktuellen Phase des Spiels zu schätzen. Als Daten dienten hierzu Bewertungspaare (v_s, v_b) von Spielfeldern, die in der normalen Baumsuche gerade in den Tiefen auftreten, in denen der Test mit der flachen Suche stattfinden soll. Konkret wurden beispielsweise zur Schätzung der Parameter für Steinanzahl 28 Spielfelder benutzt, die ausgehend von Stellungen mit 22 Steinen durch eine Zugfolge der Länge 6 erreicht wurden. In vorangehenden Untersuchungen hatte sich nämlich gezeigt, daß unter Turnierbedingungen in der Eröffnung und im Mittelspiel die selektive Suche nutzend regelmäßig das Erreichen von Tiefe 13 bis 14 zu erwarten ist. Abbildung 4.3 zeigt die jeweils ca. 2.000 Datenpaare samt der approximierenden Geraden und den Parametern für 28 bzw. 44 Steine. Nicht nur der visuelle Eindruck, sondern auch die Bestimmtheitsmaße der Regression — r^2 — jenseits von 0.9 machen deutlich, daß das lineare Modell den Zusammenhang der Bewertungen adäquat beschreibt.

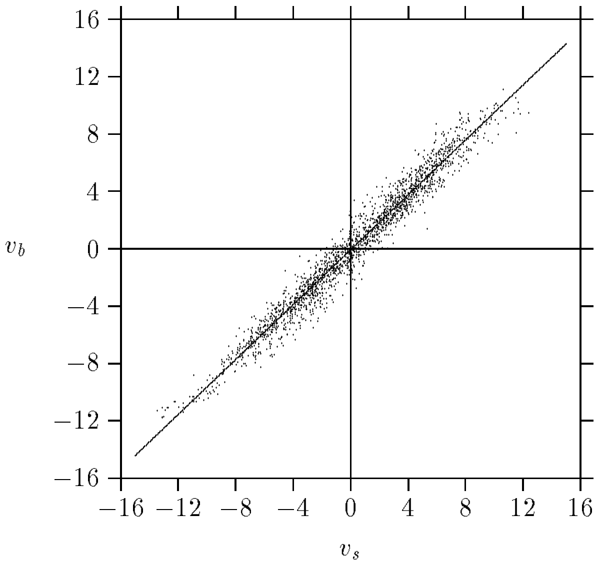
Für den praktischen Einsatz von ProbCut muß noch `PERCENTILE` geeignet gewählt werden. Hierzu wurden eine Reihe von Turnieren gespielt, in denen jeweils ein selektives Programm gegen das alte, nichtselektive Programm antrat. Zu den schon in Kapitel 3 benutzten 35 Spielfeldern aus der Eröffnungsphase wurden Hin– und Rückspiel unter üblichen Turnierbedingungen — 30 Minuten pro Spieler — gemacht, um schließlich das `PERCENTILE` mit der höchsten Gewinnquote zu wählen. Die folgende Tabelle faßt die Turnierergebnisse aus Sicht der selektiven Programme zusammen:

`PERCENTILE`	Resultat	Gewinnquote
1.20 (88.5%)	41 – 6 – 23	62.9%
1.35 (91.2%)	45 – 2 – 23	65.7%
1.50 (93.3%)	50 – 4 – 16	74.2%
1.65 (95.1%)	47 – 4 – 19	70.0%
1.80 (96.4%)	49 – 1 – 20	70.7%
2.00 (97.7%)	39 – 6 – 25	60.0%

Die aktuelle Version von LOGISTELLO sucht selektiv mit `PERCENTILE` = 1.5. Die erhebliche Steigerung der Spielstärke ist auch am Resultat eines Turniers zwischen der selektiven Version mit 30 Minuten Bedenkzeit und der ursprünglichen mit 60 Minuten (ohne Berechnungen auf gegnerischer Zeit) zu erkennen: 41 – 8 – 21 (64.7%).



$$n = 28,\quad \hat{a} \doteq 1.036,\quad \hat{b} \doteq -0.009,\quad \hat{\sigma} \doteq 0.542,\quad r^2 \doteq 0.971$$



$$n = 44,\quad \hat{a} \doteq 0.956,\quad \hat{b} \doteq -0.067,\quad \hat{\sigma} \doteq 0.884,\quad r^2 \doteq 0.963$$

Abbildung 4.3: Zusammenhang von v_s und v_b bei 28 bzw. 44 Steinen

4.2 Selektive Entscheidungssuche

Othello–Endspiele wurden von Computerprogrammen seit jeher gut beherrscht, stellen aber menschliche Spieler oft vor große Probleme. Es ist für Menschen unter Turnierbedingungen schwer möglich, für eine längere gegebene Zugfolge selbst die letztlich erreichte Steindifferenz zu bestimmen, geschweige denn, die Optimalität einer Variante zu beweisen. Man muß also Ausschau halten nach intuitiven Konzepten, die die menschliche selektive Suche steuern, ohne Steindifferenzen bestimmen zu müssen. Hier sind z.B. die Konzepte der Parität und der Zugriffsmöglichkeit auf wichtige Felder zu nennen. Leider führen diese einfachen Ansätze alleine häufig nicht zum Ziel, denn optimale Zugfolgen sind in knappen Endspielen oft unintuitiv — eine breitere Suche kann also notwendig sein. Für den Computer hingegen stellt sich die Situation folgendermaßen dar:

- Die Bestimmung der Steindifferenz ist inkrementell sehr schnell möglich.
- Die Bewertungen in den Blattpositionen sind exakt.
- Die Anzahl der Züge in jeder Position ist relativ klein.
- Die maximale Tiefe des Suchbaums kann exakt bestimmt werden.

Somit bestehen ideale Voraussetzungen für die Anwendung einer tiefen, vollständigen Suche. Unter Turnierbedingungen beginnen heutzutage gute Programme die Entscheidungssuche auf schnellen PCs oft schon bei 42 Steinen auf dem Brett. Ist es dem Menschen bis dahin nicht gelungen, sich einen nicht zu kleinen Vorsprung zu erarbeiten, so wird er das Spiel aus den oben genannten Gründen wahrscheinlich verlieren. Wie läßt sich nun das Computer–Othello–Endspiel weiter verbessern? Möchte man an exakten Suchergebnissen festhalten, so gibt es die Möglichkeit, während der Suche Prädikate auszuwerten, die frühzeitig, schnell und korrekt Auskunft über Gewinn oder Verlust geben. Beispielsweise ist es hinreichend, mehr als 32 stabile Steine zu besitzen, um ein Spiel zu gewinnen. Ist eine Funktion S bekannt, die zu einer gegebenen Stellung schnell eine gute untere Schranke für die Anzahl stabiler Steine einer Farbe liefert, so können die folgenden Tests innerhalb der `EndAlphaBeta`–Funktion zum Abschneiden großer Suchbäume führen:

```
if (S(&pos, player  ) > 32) return WIN;
if (S(&pos, opponent) > 32) return LOSS;
```

Experimente mit einer verbesserten Version der in Rosenbloom (1982) und Mitchell (1984) beschriebenen Funktionen führten leider zu keiner allgemeinen Beschleunigung der Entscheidungssuche, da mehr als 32 stabile Steine selten vorkommen, die Berechnung zu lange dauerte und die Funktion selbst eine zu grobe untere Schranke berechnete. Dieser Umstand stellt aber den Ansatz nicht in Frage, denn es mögen bessere Prädikate existieren. Bei dem Spiel „Vier gewinnt" gelang es Uiterwijk et al. (1989) mit obiger Methode, den Suchbaum drastisch zu verkleinern, so daß es möglich wurde, den Spielwert zu Beginn des Spiels zu bestimmen: Der anziehende Spieler gewinnt.

Auch im Othello–Endspiel sollte es möglich sein, selektive Suche gewinnbringend einzusetzen. Da Selektivität auch immer etwas mit Fehlerhaftigkeit zu tun hat, stellt sich direkt die Frage, warum man Ungenauigkeiten hinnehmen sollte. Hier ist doch optimales Spiel in greifbarer Nähe und die Erfahrung zeigt, daß auf der Hand liegende Endspielzüge oftmals ein Spiel kippen können. Der Einsatz selektiver Suche kann aber durch die folgende Betrachtung des Endspielszenarios motiviert werden: Wie bei der heuristischen Suche geht es auch hier darum, nicht nur gute Züge zu finden, sondern dieses auch in einer vorgegebenen Zeit zu schaffen. Im Othello–Spiel gibt es einen Zeitpunkt, bei dem von der iterativen heuristischen Suche auf die Entscheidungssuche umgeschaltet wird, um den Ausgang des Spiels exakt zu ermitteln und damit u.U. den bis dato gefundenen Zug durch einen optimalen zu ersetzen. Dieses geschieht für LOGISTELLO typisch im Turnierspiel bei etwa 41 Steinen auf dem Brett nach einer Tiefe–15–ProbCut–Suche. Somit erfolgt die Bewertung der Stellung aufgrund von Einschätzungen von Spielsituationen mit ca. 56 Steinen, die immerhin noch 8 Halbzüge vom Ende des Spiels entfernt sind. Es kann also durchaus zu Fehleinschätzungen und — noch schlimmer — zu Fehlzügen kommen. Um dennoch einen Gewinnzug finden zu können (falls ein solcher existiert und der bisherige Zug nicht gewinnt), wird nun die Entscheidungssuche gestartet. Je schneller man in dieser Situation einen Gewinnzug finden kann, desto früher kann mit der Entscheidungssuche begonnen werden und umso öfter kann man falsche heuristische Züge korrigieren. Hierbei wäre es zwar angenehm, wenn die Entscheidungssuche exakte Ergebnisse lieferte, dieses ist aber beim bisherigen Stand der Othello–Theorie nur mit zeitintensiver vollständiger Suche zu erreichen — und vor allem auch nicht notwendig, denn eine schnelle selektive Entscheidungssuche mit kleiner Fehlerwahrscheinlichkeit erhöht auch die Spielstärke, wie im folgenden gezeigt wird.

Der hier untersuchte Ansatz zur selektiven Entscheidungssuche ähnelt der ProbCut–Erweiterung. Analog dazu wird bei einer gewissen Steinanzahl eine flache heuristische Suche durchgeführt und deren Ergebnis benutzt, um mit einer gegebenen Sicherheit zu entscheiden, ob die vorliegende Stellung für den Spieler am Zug gewonnen oder verloren ist. Die Güte der Entscheidung ist wiederum abhängig von Parametern, die geeignet gewählt werden müssen. Abbildung 4.4 verdeutlicht die Vorgehensweise. Abhängig von der globalen Variable `selective` kann festgelegt werden, ob selektiv gesucht werden soll. Dieses geschieht nur in der Entscheidungssuche — also für [`alpha`,`beta`] $\subseteq [-1, 1]$, an die sich, falls noch Zeit übrig ist, eine exakte Entscheidungssuche und daran die Maximierungssuche anschließt.

Anhand von Endspielstellungen soll die selektive Entscheidungssuche für verschiedene Werte für `CUT_NIVEAU` untersucht werden. Hierzu wurde in der Spieledatenbank nach für Schwarz gewonnenen Stellungen mit 42 Steinen gesucht, bei denen eine Tiefe–14–ProbCut Suche jeweils nur einen Verlustzug als „besten“ Zug berechnet. Genau in diesen Situationen ist es wichtig, daß die sich anschließende Entscheidungssuche existierende Gewinnzüge in der gegebenen Zeit finden kann. Tabelle 4.1 zeigt für die

```
#define CHK_DISCS  50   /* here we start a shallow */
#define CHK_DEPTH  4    /* search with this depth  */
#define CUT_NIVEAU 2.1

int EndAlphaBeta(int alpha, int beta)
{
  ...
  if (selective && discs == CHK_DISCS) {
    val = AlphaBeta(CHK_DEPTH, VAL_MIN, VAL_MAX);

 /* perform a cut if the normalized value is at */
 /* least +CUT_NIVEAU resp. at most -CUT_NIVEAU */

     if (val >= round(+CUT_NIVEAU*FLOAT2INT)) return +1;
     if (val <= round(-CUT_NIVEAU*FLOAT2INT)) return -1;
   }
   ...
 }
```

Abbildung 4.4: Die EndCut–Erweiterung

Pos.	exakt	2.5		2.1		1.8		1.5	
1	383	170	(0.44)	134	(0.35)	117	(0.31)	90	(0.23)
2	672	*21	(1.03)	*18	(1.03)	*11	(1.02)	*8	(1.01)
3	474	358	(0.76)	205	(0.43)	174	(0.37)	106	(0.22)
4	334	230	(0.69)	207	(0.62)	*182	(1.54)	*72	(1.22)
5	198	56	(0.28)	41	(0.21)	30	(0.15)	22	(0.11)
6	584	246	(0.42)	148	(0.25)	85	(0.15)	59	(0.10)
7	126	94	(0.75)	91	(0.72)	48	(0.38)	27	(0.21)
8	454	165	(0.36)	79	(0.17)	73	(0.16)	69	(0.15)
9	553	414	(0.75)	315	(0.57)	229	(0.41)	*1	(1.00)
10	427	185	(0.43)	*401	(1.94)	*184	(1.43)	*146	(1.34)
11	211	101	(0.48)	61	(0.29)	46	(0.22)	*2	(1.01)
12	465	431	(0.93)	392	(0.84)	267	(0.57)	208	(0.45)
13	449	224	(0.50)	180	(0.40)	128	(0.29)	57	(0.13)
14	485	164	(0.34)	150	(0.31)	113	(0.23)	100	(0.21)
15	725	455	(0.63)	310	(0.48)	*18	(1.02)	*17	(1.02)
16	151	65	(0.43)	32	(0.21)	18	(0.12)	13	(0.09)
17	164	77	(0.47)	54	(0.33)	37	(0.23)	31	(0.19)
18	89	50	(0.56)	44	(0.49)	28	(0.31)	27	(0.30)
19	272	181	(0.67)	144	(0.53)	109	(0.40)	*25	(1.09)
20	115	62	(0.54)	35	(0.30)	37	(0.32)	24	(0.21)
21	303	218	(0.72)	148	(0.49)	166	(0.55)	*9	(1.01)
22	203	87	(0.43)	63	(0.31)	38	(0.19)	34	(0.17)
23	197	196	(1.00)	*189	(1.96)	*26	(1.13)	*13	(1.07)
24	1214	985	(0.81)	900	(0.74)	*325	(1.27)	*196	(1.16)
25	325	287	(0.88)	197	(0.61)	193	(0.59)	185	(0.57)
26	269	263	(0.98)	254	(0.94)	205	(0.76)	139	(0.52)
27	378	129	(0.34)	110	(0.29)	*85	(1.22)	*50	(1.13)
28	97	63	(0.65)	30	(0.31)	24	(0.24)	14	(0.14)
29	177	141	(0.80)	90	(0.51)	79	(0.47)	32	(0.18)
30	803	552	(0.69)	*236	(1.30)	*173	(1.22)	*141	(1.18)
31	42	24	(0.57)	21	(0.50)	23	(0.55)	11	(0.26)
32	289	*32	(1.11)	*47	(1.16)	*12	(1.04)	*10	(1.03)
33	234	219	(0.94)	180	(0.77)	172	(0.74)	162	(0.69)
34	130	72	(0.55)	61	(0.47)	51	(0.39)	44	(0.34)
35	88	*1	(1.01)	*1	(1.01)	*1	(1.01)	*1	(1.01)
∅	345.1	230.5	(0.66)	229.8	(0.62)	246.3	(0.60)	246.03	(0.59)

Tabelle 4.1: Zeiten für die Entscheidungssuche in Sekunden

35 im Anhang D verzeichneten Positionen die benötigte CPU–Zeit (einer SPARC10/M30–Workstation) für die exakte bzw. selektive Entscheidungssuche unter Verwendung verschiedener Niveaus. Fehlerhafte Zugbestimmungen sind hierbei mit einem Stern markiert. In der letzten Zeile sind die mittleren Lösungszeiten und deren Verhältnis zur Zeit der exakten Suche für das Auffinden der Gewinnzüge angegeben, wenn zunächst selektiv und danach, falls noch kein Gewinnzug gefunden wurde, exakt gerechnet wird. Auch hier zeigt sich eine deutliche Verbesserung gegenüber der nicht selektiven Programmversion: Die mittlere Zeit zum Finden eines Gewinnzuges ist bei `CUT_NIVEAU` = 2.1 um 38% kleiner und in über 80% der Fälle — in denen auch ein Gewinnzug gefunden wird — um 54%. Somit kann durch Einsatz selektiver Suche in kritischen Situationen sehr oft viel früher ein Gewinnzug gefunden werden.

4.3 Diskussion

In diesem Kapitel wurden zwei Erweiterungen des $\alpha\beta$–Algorithmus' — ProbCut und EndCut — zur Steuerung einer selektiven Suche vorgestellt, die zu einer deutlichen Steigerung der Spielstärke gegenüber der nicht–selektiven Suche führten. Die Idee besteht darin, Spielwerte, die sich aus einer tieferen Suche ergeben, mittels denen einer flacheren Suche zu approximieren und hiermit in einem gewissen Abstand zum Suchhorizont frühzeitig zu entscheiden, ob es mit einer vorgegebenen Sicherheit zu Schnitten kommt, um in diesem Fall den Suchbaum nicht tiefer zu untersuchen. Voraussetzung für die erfolgreiche Anwendung der Verfahren ist eine relativ stabile Bewertungsfunktion oder eine Ruhesuche. Beide Eigenschaften gewährleisten eine kleine Varianz der Bewertungsunterschiede auf verschiedenen Tiefen und ermöglichen dadurch, fundiert ganze Teilsuchbäume von einer tiefen Suche auszuschließen.

Weitere Verbesserungen des Ansatzes sind möglich. So konnte bei ProbCut durch Einführen einer weiteren Schnittebene die Spielstärke noch einmal leicht erhöht werden. Eine offensichtliche Schwäche der vorgestellten Vorgehensweise besteht darin, daß der Abstand der Schnittebenen zum Horizont fix ist. Dadurch nähert sich der mittlere Verzweigungsfaktor mit zunehmender Tiefe dem ursprünglichen und es ergibt sich keine wesentliche Verbesserung mehr. Denkbar ist hier die moderate Vergrößerung des Abstandes in Abhängigkeit von der Iterationsstufe, um so immer tiefere Teilbäume abzuschneiden und damit den mittleren Verzweigungsfaktor klein zu halten.

Im Gegensatz zu bekannten Konzepten für die selektive Suche, die Informationen über den gesamten bisher durchsuchten Baum im Speicher halten müssen — beispielsweise von McAllester (1988), Rivest (1988) oder Palay (1985) — benötigen die vorgestellten Methoden keinen zusätzlichen Speicherplatz und stellen eine natürliche Erweiterung der effizienten $\alpha\beta$–Suche dar. Darüberhinaus ist die Wirkung nicht wie beim Ansatz „Singular Extensions“ von Anantharaman et al. (1990) auf Stellungen taktischer Natur beschränkt, bei denen ein Zug existiert, der wesentlich besser ist als die anderen. Denn jeweils nach schlechten Zügen, die in der normalen Baumsuche häufig untersucht werden, ist die Wahrscheinlichkeit für einen frühzeitigen Schnitt groß.

Kapitel 5

Strategien für eine Partienfolge

In den vorangegangenen Kapiteln sind Lernansätze vorgestellt worden, bei denen Musterbewertungen, Gewichte von Merkmalen und Parameter für die selektive Suche geschätzt wurden. Hier kann von indirekten Lernverfahren gesprochen werden, da eine Menge von Beispielpositionen Gegenstand statistischer Untersuchungen waren und die Verfahren dazu dienten, die allgemeine Qualität der Zugentscheidung zu steigern. In diesem Kapitel wird ein Ansatz betrachtet, mit dessen Hilfe aus gespielten Partien direkt im Hinblick darauf gelernt werden kann, daß zwei Spiele nicht auf die gleiche Weise verloren werden sollen. Darüberhinaus kann mit der Methode ein ansonsten stärkerer Gegner, der aber nicht über einen vergleichbaren Mechanismus verfügt, in einer Folge von Spielen kompromittiert werden. Zusätzlich wird es möglich, eine auf die eigenen Schwächen eingehende Eröffnungsbibliothek automatisch zu erzeugen.

5.1 Strategien

Möchte man nicht nur in einem einzigen Spiel gegen einen unbekannten Gegner gut spielen, sondern auch über eine Folge von Spielen erfolgreich sein, so sieht man sich mit einfachen, aber wirkungsvollen Strategien des Gegenspielers konfrontiert, denen nicht mit den bisher besprochenen Ansätzen zu begegnen ist. Die vielleicht naheliegenste Strategie ist

I. *Hast Du ein Spiel gewonnen, so versuche es auf die gleiche Art nochmal.*

Ein nur mit einer MiniMax–Suche ausgestattetes Programm verfolgt diese Strategie, ist ihr aber auch hilflos ausgeliefert. Die Gegenmaßnahme liegt auf der Hand: Notwendig ist ein Mechanismus, mit dem geeignete Zugalternativen bestimmt werden können. Ein scheinbarer Ausweg besteht in der Benutzung einer großen, randomisierten Eröffnungsbibliothek, bei der die Wahrscheinlichkeit, überhaupt ein Spiel zu wiederholen, klein ist. Hierbei muß aber gesehen werden, daß dieser Ansatz unflexibel ist und vor allem mit jedem verlorenen Spiel die Wahrscheinlichkeit wächst, ein weiteres zu verlieren. Eine einfache, aber verblüffend wirkungsvolle Gegenmaßnahme zu Strategie I ist die folgende

II. *Versuche, eigene verlorene Spiele aus der Sicht des Gegners zu wiederholen.*

Die Idee ist hierbei, sich die eigenen Fehler vom Gegenspieler zeigen zu lassen, um in Zukunft des Gegners Züge selbst zu spielen! Auf diese Art kann ein in allgemeiner Situation stärkerer Spieler ohne Lernmechanismus in einer Folge von Spielen kompromittiert werden, da er letztlich gegen sich selber spielt und aus Verlusten nichts lernt. Ohne viel Aufwand kann somit selbst ein guter Spieler unter Druck gesetzt werden. Bei obiger Argumentation wurde stillschweigend vorausgesetzt, daß der nicht–lernende Spieler aus seiner Sicht optimale Züge macht. Ist sich ein Spieler der Schwäche des Gegners bewußt, so muß er aber keinegswegs derart ziehen. Er kann sich die guten Züge vielmehr als „Überraschung“ für eines der nächsten Spiele aufheben in der begründeten Hoffnung, auch mit einem suboptimalen Zug zu gewinnen, der dann prompt vom einem Gegner, der Strategie II benutzt, im nächsten Spiel nachgespielt wird und dann u.U. verliert. Auf diese Feinheiten soll hier nicht näher eingegangen werden, da es gegen einen ungefähr gleichstarken Gegenspieler gefährlich ist, aus eigener Sicht suboptimale Züge zu machen.

Die obigen Strategien können als passiv bezeichnet werden, da nicht aktiv nach Zugalternativen gesucht wird. Aber schon wenn ein Gegner die einfache Strategie II anwendet, ist diese Suche notwendig, wenn man sich nicht mit einer ausgeglichenen Bilanz zufrieden geben will. Dieses führt schließlich zu

III. *Kam die aktuelle Position schon einmal vor und ist dort keine Gewinnfortsetzung bekannt, so suche nach einer vielversprechenden Zugalternative.*

Ein die besprochenen Strategien verfolgendes Programm ist ein unangenehmer Spielpartner. So versucht es, eigene und gegnerische Gewinne zu wiederholen und verliert nicht zweimal auf die gleiche Art, da es aktiv nach Zugalternativen sucht. Weiterhin lernt es gute Eröffnungspfade, die es vielleicht nie aus eigenem Antrieb beschreiten würde, und vermag eigene Fehler in folgenden Spielen zu korrigieren. Schließlich ist es ihm möglich, auch die Güte gefundener Abweichungen in Spielen gegen sich selbst zu prüfen, und damit u.U. einem etwaigen Verlust in einem bewerteten Turnierspiel vorzugreifen. Auf die algorithmische Realisation der Strategien geht der folgende Abschnitt ein.

5.2 Realisation

Grundlegende Schwierigkeiten bereitet die Verwirklichung der Strategien I und II nicht: So kann man sich eigene verlorene Spiele merken und daraus einen Spielbaum erstellen, in dem die Spielresultate gemäß der NegaMax–Strategie von den Blättern zur Wurzel propagiert werden. Im Spiel ist dann die jeweils aktuelle Position im Baum zu suchen und der Spielwert zu betrachten. Im Fall, daß eine aus bisheriger Erfahrung gewonnene Position vorliegt, ist die Entscheidung einfach: Gemäß Strategie II macht man einen Zug, der früher zu einem Sieg führte. Über das Vorgehen im anderen Fall gibt Strategie III Auskunft — es ist nach einer geeigneten Zugalternative zu suchen. Dieses ist natürlich nicht nur auf die betrachtete Position beschränkt, denn ein Abweichen zu einem späteren Zeitpunkt kann erfolgversprechender sein. Notwendig ist in diesem Zusammenhang also die Bewertung der bestmöglichen Zugalternative in jeder Stellung des Baums, um damit eine global optimale Alternative finden zu können. Hierbei ist natürlich wünschenswert, daß sämtliche zeitaufwendigen Berechnungen vor einer Partie stattfinden, um im Spiel keine Zeit zu verlieren. Speziell bei Othello wird man nur in der Eröffnung und im Mittelspiel nach Alternativen suchen — im Endspiel ist dies zwecklos, da dort optimal gespielt wird. Die folgende Definition konkretisiert das Vorgehen:

Es sei T_0 der vollständige Spielbaum und T der aus den gelernten Partien gebildete Teilbaum. Für eine Stellung v in T sei $S(v)$ die Menge der Nachfolger von v in T und $S'(v)$ die Menge der Nachfolger in $T_0 \backslash T$. $V_T(v)$ bezeichne den NegaMax–Spielwert der Stellung v bezüglich des Spielausgangs in T — dieser wird im folgenden Baumwert genannt. Dann ist die Chance $V_C(v) \in I\!R \cup \{-\infty, +\infty\}$ folgendermaßen rekursiv definiert:

- *Stellungen v in $T_0 \backslash T$ werden durch $V_C(v)$ z.B. mittels einer Baumsuche heuristisch bewertet. Für alle nicht in T vorkommenden Stellungen liegt hiermit eine Bewertung vor, die zum Finden global guter Zugalternativen benutzt werden kann.*

- *Für Blattstellungen v von T wird durch*

$$V_C(v) = \begin{cases} +\infty, & \textit{falls } V_T(v) > 0 \\ 0, & \textit{falls } V_T(v) = 0 \\ -\infty, & \textit{falls } V_T(v) < 0 \end{cases}$$

  modelliert, daß der Baumwert bezüglich T_0 exakt ist und somit Zugabweichungen nicht mehr betrachtet werden müssen.

- *Die Chance für innere Stellungen v von T ergibt sich rekursiv gemäß*

$$V_C(v) = \begin{cases} \max\{-V_C(w) \mid w \in S(v) \ \textit{mit } V_T(w) < 0\} \\ \max\{-V_C(w) \mid w \in S(v) \ \textit{mit } V_T(w) = 0 \ \textit{oder } w \in S'(v)\} \\ \max\{-V_C(w) \mid w \in S(v) \ \textit{mit } V_T(w) > 0 \ \textit{oder } w \in S'(v)\} \end{cases}$$

  je nachdem, ob $V_T(v) > 0$, $= 0$ oder < 0 ist. In jedem Fall werden nur Nachfolger betrachtet, die zum bestmöglichen Baumwert führen, und falls die Stellung bzgl. T remis oder verloren ist, zusätzlich auch der Nachfolger, der sich aus der (heuristisch) besten Zugalternative ergibt. Aus der Sicht des in v am Zug befindlichen Spielers ist $V_C(v)$ dann die Bewertung der Stellung, die sich am Ende einer Zugfolge in T ergibt, wenn beide Spieler die drei Strategien verfolgen.

Für eine Stellung v wird derjenige Bibliothekszug gewählt, der in Abhängigkeit von $V_T(v)$ zu einem Nachfolger $w \in S(v)$ bzw. $w \in S'(v)$ mit maximalem Wert $-V_C(w)$ führt. Dadurch werden in bzgl. T gewonnenen Stellungen nur Gewinnzüge in Betracht gezogen, und anderenfalls nach global guten Zugabweichungen gesucht. Ein Beispiel soll das Verfahren verdeutlichen. In Abbildung 5.1 ist ein fiktiver Spielbaum dargestellt, bei dem die

Stellungen mit Baumwert und Chance gemäß obiger Definition markiert sind. Die bezüglich der heuristischen Bewertung besten Zugalternativen in jeder inneren Stellung sind durch gestrichelte Linien gekennzeichnet. Die drei Blätter auf der letzten Ebene mit Chancen $\pm\infty$ seien Terminalknoten. Hier kann man sich z.B. Othello–Stellungen mit 41 Steinen vorstellen, deren exakter Spielwert bekannt ist und bei denen deshalb etwaige zukünftige Zugalternativen nicht mehr betrachtet werden müssen. In der Wurzelstellung v_1 ist nun ein vielversprechender Zug zu finden. Wie man sieht, ist die Stellung im Hinblick auf die drei Spiele verloren. Es geht also darum, sich für einen „verlierenden“ Zug oder die Zugalternative zu entscheiden. Unter der Voraussetzung, daß der Gegner aus bisheriger Sicht gewinnende Züge wiederholt, ist die für den Spieler bezüglich seiner Bewertung opti-

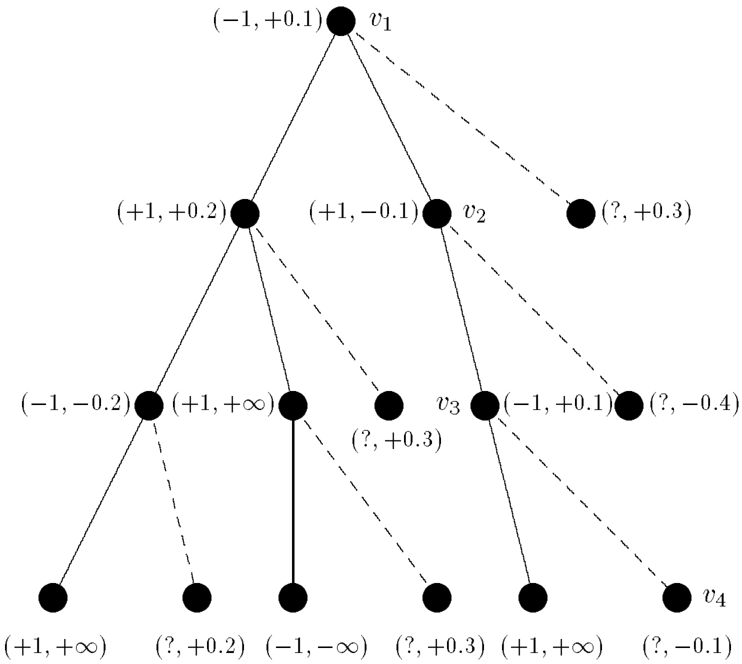


Abbildung 5.1: Ein Beispielbaum

male Stellungsfolge $v_1, \ldots, v_4$. In v_4 besteht begründete Hoffnung für den Spieler, da für den Gegner die Bewertung -0.1 — also negativ ist. Somit wird bei v_1 der zweite Zug und bei v_3, falls der Gegner den früheren Gewinnzug wiederholt, der Alternativzug gewählt.

Für den praktischen Einsatz des Verfahrens sind zwei Fragen zu beantworten: Aus welchen Spielen soll der Baum erstellt werden und wie werden Zugalternativen bewertet? Die im Kapitel 3 entwickelten Bewertungsfunktionen besitzen eine Spielphasen–unabhängige Interpretation: Sie approximieren die auf die Merkmale bezogene Wahrscheinlichkeit, daß eine Stellung für den anziehenden Spieler gewonnen ist. Somit sind Vergleiche von Bewertungen über Spielphasen hinweg ohne weiteres möglich. Es genügt also, in jeder Stellung des Spielbaums T mittels einer NegaMax–Suche den besten bisher noch nicht gespielten Zug samt des dazugehörigen Spielwertes zu bestimmen und als potentielle Zugalternative anzufügen. Hierbei sollte natürlich die Suchtiefe möglichst nicht kleiner als die in einem Turnierspiel erreichte sein, um später nach der Entscheidung für eine Zugalternative nicht direkt eine Überraschung zu erleben.

Die Auswahl der Spiele erfolgt im Hinblick auf die verfolgten Strategien: Verlorene Partien werden ohne Einschränkung aufgenommen — sie sollen mit vertauschten Rollen nachgespielt werden. Das Importieren gewonnener Spiele birgt dagegen eine große Gefahr, denn es werden auch Partien nachgespielt, in denen der Gegner gut stand, dann aber nach einem groben Fehler verlor. Leider sind dieses aber gerade die lernenswerten gewonnenen Spiele — sie sollten korrigiert werden. Allgemein kann dieses dadurch geschehen, daß auch die gewonnenen Spiele mit einer guten gegnerischen Zugalternative aufgenommen werden und dann in der aktuellen Bibliothek fortwährend nach „suspekten“ Positionen gesucht wird, die gemäß des Baumwertes für den Spieler am Zug nicht gewonnen sind, bei denen aber eine chancenreiche Zugalternative existiert. Die ermittelten Partieanfänge können danach ab diesen Stellen mit Turnierstärke beendet und in diesem Sinne korrigiert werden.

Weiterhin ist bei Othello auch die direkte Endspielkorrektur lohnenswert, denn hier ist man nicht auf eine fehlerhafte heuristische Bewertung angewiesen. Wenn genügend Rechnerleistung zur Verfügung steht, kann der exakte Spielwert ein paar Halbzüge früher als im Turnierspiel ermittelt werden. Ergibt sich hierbei eine Änderung bezüglich des Gewinns, so sollte das Spiel aufgenommen werden, damit man eigene Endspielfehler nicht wiederholt und nicht mit denen des Gegners rechnet.

5.3 Diskussion

Durch die Betrachtung einfacher Strategien für eine Folge von Spielen konnte hier ein Verfahren entwickelt werden, das das direkte Lernen aus eigenen Partien ermöglicht. Durch Aufbau eines Spielbaums aus den gelernten Partien und der heuristischen Bewertung aller Zugalternativen wird es möglich, gegnerische Siege aus der Sicht des Gegners zu wiederholen, um sich eigene Fehler zeigen zu lassen, oder in folgenden Spielen nach global — d.h. bezüglich aller importierten Partien — guten Zugalternativen zu suchen, um nicht zweimal auf die gleiche Art zu verlieren. Der Zeitaufwand für die Bewertung der Alternativen, die sich durch die Aufnahme eines neuen Spiels ergeben, ist bei Benutzung der Turniertiefe vergleichbar mit der Dauer des Spiels selbst — also akzeptabel. Aufwendig ist hingegen die Neubewertung aller Stellungen, wenn die Bewertungsfunktion in starkem Maße geändert wird. Da diese Aufgabe aber leicht optimal zu parallelisieren ist, können mehrere Rechner wirkungsvoll zur Bewertung eingesetzt werden.

Der entwickelte Ansatz zum direkten Lernen von Partien geht über gebräuchliche Verfahren wie die Benutzung einer randomisierten Eröffnungsbibliothek oder früher berechneter Stellungsbewertungen bei der Suche — wie beispielsweise von Samuel (1959) und Scherzer et al. (1990) beschrieben — hinaus, da hier im Gegensatz dazu brauchbare Zugalternativen global bestimmt werden.

Im praktischen Einsatz hat sich das Verfahren in leicht modifizierter Form bewährt: Vertrauend auf seine Spielstärke sieht LOGISTELLO unentschiedene Spiele als verloren an, um diese nicht zu wiederholen. Angeschlossen an den Internet–Othello–Server (IOS) der Universität–GH–Paderborn hat LOGISTELLO in Hunderten von Spielen gegen starke Othello–Programme, die auch mit Lernmechanismen ausgestattet waren, seine große allgemeine Spielstärke demonstrieren können, da die Gegner nach gewonnenen Spielen sozusagen gegen sich selbst antraten oder immer wieder vor neue Aufgaben gestellt wurden.

Kapitel 6

Schlußbemerkungen

Für das Gebiet der Spielbaumsuche sind in dieser Arbeit folgende Beiträge geliefert worden:

- Bei der Merkmalskonstruktion ist eine Möglichkeit aufgezeigt worden, global definierte Merkmale lokal schnell mit großer Güte zu approximieren, um so die Suchgeschwindigkeit bei ähnlicher Bewertungsgüte deutlich zu erhöhen. Darüberhinaus ist eine neue Klasse von Merkmalen vorgestellt worden, bei der eine sehr schnelle Bewertung verschiedenartigster Stellungsaspekte auf der Basis von Zugriffen auf Tabellen möglich wird, deren Einträge mittels nach Gewinn und Verlust klassifizierter Beispielpositionen geschätzt werden.

  [Kapitel 2]

- Mit der logistischen Regression und der linearen Diskriminanzanalyse sind zwei im Bereich der Merkmalskombination bisher noch nicht eingesetzte statistische Verfahren eingeführt und mit ihnen bestimmte lineare Kombinationsfunktionen mit der von Lee & Mahajan (1988,1990) benutzten quadratischen Diskriminanzfunktion hinsichtlich der resultierenden Spielstärke empirisch verglichen worden. Es stellte sich heraus, daß im Rahmen des hier entwickelten Spielprogramms keineswegs die Verwendung der quadratischen Kombinationsfunktion zu einer größeren Spielstärke führt. Im Paarvergleich erzielte die Programmversion mit den mitteln der logistischen Regression bestimmten Gewichten die besten Resultate. Somit steht eine weitere adäquate Methode zur Merkmalsgewichtung zur Verfügung, die zudem den Vorteil hat, keine bestimmte Verteilung der Merkmale vorauszusetzen und auch Merkmale mit nur wenigen Ausprägungen zuzulassen. Im Hinblick auf die bei der Parameterschätzung aufgetretenen — aber unerwünschten — negativen Vorzeichen von Merkmalsgewichten wurde desweiteren ein Verfahren vorgestellt, das die Vorzeichenkonsistenz gewährleistet und damit zu einer Stabilisierung der Bewertungsfunktionen führte.

[Kapitel 3]

- Im Bereich der selektiven Suche wurden zwei natürliche Erweiterungen des $\alpha\beta$–Algorithmus' — ProbCut und EndCut — entwickelt, die auf der Schätzung der Spielwerte von Wurzelstellungen tiefer Suchbäume mittels aus flachen Suchen gewonnenen Werten beruhen. Die Verfahren benötigen keinen zusätzlichen Speicherplatz und sind nicht nur auf taktische Situationen beschränkt. Ihr Einsatz bewirkte eine deutliche Steigerung der Spielstärke.

[Kapitel 4]

- Die Forderung, nicht zwei Spiele auf die gleiche Weise zu verlieren, führte zu der Betrachtung einfacher Strategien für eine Folge von Partien. Es ist ein Algorithmus entwickelt worden, der obiger Bedingung gerecht wird und es zudem ermöglicht, sich eigene Fehler vom Gegner zeigen zu lassen und eine auf die eigenen Schwächen eingehende Eröffnungbibliothek zu erstellen. Kernpunkt ist ein Verfahren, das mittels gemerkter Partien und heuristischen Bewertungen aller Zugalternativen das Finden global guter Zugabweichungen zuläßt.

[Kapitel 5]

Die Wirksamkeit der vorgestellten Techniken konnte neben den eigenen Untersuchungen auch durch die erfolgreiche Teilnahme des Othello–Programms LOGISTELLO an etlichen internationalen Turnieren gezeigt werden. Da die Verfahren allgemeiner Natur sind, ist deren Anwendung auch bei anderen Spielen denkbar und sollte auch dort zu spielstärkeren Programmen führen können, bei denen keine Wahl eines Parameters der Intuition überlassen wird.

Die Verwendung selektiver Suche oder statistischer Methoden zur Merkmalskonstruktion und -gewichtung ist nicht nur auf das Gebiet der Spielbaumsuche beschränkt. Auch bei Suchverfahren, bei denen heuristische Funktionen nur die Suchreihenfolge, nicht aber das Suchergebnis selbst bestimmen, kann der Einsatz statistischer Vefahren oder selektiver Suche sinnvoll sein, um einerseits die Entscheidungsgüte zu erhöhen und hierdurch u.U. die Laufzeit zu verringern, oder andererseits das schnelle Finden von Näherungslösungen zu ermöglichen. Die Übertragung der vorgestellten Methoden auf andere Suchprobleme soll beispielhaft am Erfüllbarkeits–Problem für aussagenlogische Formeln kurz erläutert werden. Analog zur Anwendung im Bereich der Spielbaumsuche können hierbei Bewertungsfunktionen eingesetzt werden als Maß für die Wahrscheinlichkeit der Erfüllbarkeit einer Booleschen Formel hinsichtlich einer Menge von Merkmalen. Wie dort ist es auch hier möglich, die Merkmalsgewichte bei einer in den Merkmalen linearen oder quadratischen Bewertungsfunktion anhand von hier gemäß ihrer Erfüllbarkeit klassifizierten Beispiele wie besprochen zu schätzen. Anwendung findet ein so erhaltener Erfüllbarkeits–Prediktor beispielsweise in Davis–Putnam–Algorithmen, die zum Auffinden einer erfüllenden Belegung — durch eine Heuristik gesteuert — Variablen jeweils mit *wahr* und *falsch* belegen, um die entstandenen kürzeren Formeln rekursiv weiter zu untersuchen. In Verbindung mit einer Vorausschau können hierbei Teilbelegungen im Hinblick auf eine Maximierung der Erfüllbarkeits–Wahrscheinlichkeit gewählt werden, so daß also im Falle der Erfüllbarkeit eine dies beweisende Belegung schnell gefunden wird. Darüberhinaus besteht nun die Möglichkeit, die Suche selektiv zu gestalten, indem beispielsweise bei Überschreiten einer bestimmten Wahrscheinlichkeit die Erfüllbarkeit einer Formel einfach angenommen wird, oder bei Unterschreiten der entsprechende aktuelle Suchpfad nicht weiter verfolgt wird. Hierdurch kann die Suchdauer bei Hinnahme einer gewissen Fehlerwahrscheinlichkeit verkürzt werden.

Literatur

AGRESTI, A. (1990)
Categorical Data Analysis
Wiley

ANANTHARAMAN, T., CAMPBELL, M.S., HSU, F.–H. (1990)
Singular Extensions: Adding Selectivity to Brute–Force Searching
Artificial Intelligence 43, S. 99–109

BRUDNO, A.L. (1963)
Boundaries and Estimates for Abriding the Search of Estimates
Problems of Cybernetics 10, S. 225–241

DUDA, R., HART, P. (1973)
Pattern Classification and Scene Analysis
Wiley

FELDMANN, R. (1993)
Spielbaumsuche mit massiv parallelen Systemen
Dissertation, Universität–GH–Paderborn

HAND, D.J. (1981)
Discrimination and Classification
Wiley

HARTUNG, J. (1987)
Statistik
R. Oldenburg Verlag

KNUTH, D.E., MOORE, R.W. (1975)
An Analysis of Alpha–Beta Pruning
Artificial Intelligence 6, S. 293–326

LANDAU, T. (1987)
Othello: Brief & Basic
USAO P.O. Box 342, Falls Church, VA 22040 U.S.A.

LEE, K.F., MAHAJAN, S. (1986)
BILL: A Table–Based Knowledge–Intensive Othello Program
Technical Report CMU–CS–86–141, Computer Science Department, Carnegie–Mellon University, Pittsburgh, PA 15213 U.S.A.

LEE, K.F., MAHAJAN, S. (1988)
A Pattern Classification Approach to Evaluation Function Learning
Artificial Intelligence 36, S. 1–25

LEE, K.F., MAHAJAN, S. (1990)
The Development of a World Class Othello Program
Artificial Intelligence 43, S. 21–36

MAIBAUM, G. (1980)
Wahrscheinlichkeitstheorie und mathematische Statistik
VEB Deutscher Verlag der Wissenschaften

MARSLAND, T.A. (1985)
Evaluation Function Factors
ICCA Journal 8(2)

MCALLESTER, D.A. (1988)
Conspiracy Numbers for MinMax Search
Artificial Intelligence 35, S. 287–310

MCCULLAGH, P., NELDER, J.A. (1989)
Generalized Linear Models
Chapman & Hall

MITCHELL, D.H. (1984)
Using Features to Evaluate Positions in Experts' and Novices' Othello Games
Master Thesis, Northwestern University, Evanston Illinois U.S.A.

MYSLIWIETZ, P. (1994)
Konstruktion und Optimierung von Bewertungsfunktionen beim Schach
Dissertation, Universität–GH–Paderborn

PALAY, A.J. (1985)
Searching with Probabilities
Pitman

REINEFELD, A. (1989)
Spielbaum–Suchverfahren
Springer–Verlag Berlin Heidelberg

RIVEST, R.L. (1988)
Game Tree Searching by MinMax Approximation
Artificial Intelligence 34, S. 77–96

ROSENBLOOM, P.S. (1982)
A World–Championship–Level Othello Program
Artificial Intelligence 19, S. 279–320

SAMUEL, A.L. (1959)
Some Studies in Machine Learning Using the Game of Checkers
IBM Journal of Research and Development 3, S. 210–229

SAMUEL, A.L. (1967)
Some Studies in Machine Learning Using the Game of Checkers II
IBM Journal of Research and Development 11, S. 601–617

SCHERZER, T., SCHERZER, L., TJADEN, D. (1990)
Learning in Bebe
In: T.A. Marsland, J. Schaeffer (Hrsg.), Computers, Chess and Cognition, Springer–Verlag

SHANNON, C.E. (1950)
Programming a Computer for Playing Chess
Philosophical Magazine 41, S. 256–275

SLATE, D.J., ATKIN, L.R. (1983)
CHESS 4.5 – The Northwestern University Chess Programm
In: P. Frey (Hrsg.), Chess Skill in Man and Machine, Springer–Verlag

UITERWIJK, J.W., VAN DEN HERIK, H.J., ALLIS, V. (1989)
A Knowledge–Based Approach to Connect–Four
In: Heuristic Programming in Artificial Intelligence: The First Computer Olympiad, Ellis Horwood Ltd.

van den Meulen, M. (1989)
Weight Assesment in Evaluation Functions
In: D.F. Beal (Hrsg.), Advances in Computer Chess 5, Elsevier Science Publishers

Wedderburn, R.W.M. (1976)
On the Existence and Uniqueness of Maximum Likelihood Estimates for Certain Generalized Linear Models
Biometrika 63, S. 27–32

Young, T.Y., Calvert T.W. (1974)
Classification, Estimation and Pattern Recognition
Elsevier

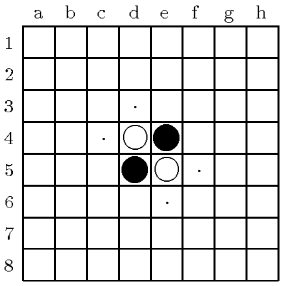
Anhang A

Othello

Das Spiel Othello ist eine Variante des im späten 19. Jahrhundert entwickelten Brettspiels Reversi, dessen Regeln 1974 von Goro Hasegawa verfeinert wurden. Othello wurde jahrelang vorwiegend in Japan gespielt, wo es das zweitpopulärste Spiel nach Go ist, und es gewinnt anderenorts an Popularität. So gibt es mittlerweile in verschiedenen Ländern nationale Othello–Verbände und seit Ende 1993 auch einen Internet–Othello–Server (IOS) an der Universität–GH–Paderborn, der es ermöglicht, zu jeder Tageszeit gegen Spieler und Computer–Programme aus aller Welt anzutreten.

Die Regeln von Othello sind einfach: Es wird auf einem 8×8–Brett mit 64 flachen zweifarbigen Spielsteinen gespielt. Schwarz beginnt in der unten gezeigten Startposition, danach wird — wenn möglich — abwechselnd gezogen. Hierzu setzt die Partei am Zug einen Stein mit ihrer Farbe nach oben auf eine noch unbesetzte Stelle des Bretts, so daß mindestens ein gegnerischer Stein dadurch zwischen dem gerade gesetzten und einem anderen eigenen Stein „eingeklemmt“ wird. Alle derart horizontal, vertikal und diagonal gefangenen Steine werden herumgedreht, sie zeigen dann die Farbe des am Zug befindlichen Spielers. Ein Beispiel ist auf der nächsten Seite zu sehen. Hat ein Spieler keine Zugmöglichkeit, so muß er passen. Das Spiel ist beendet, wenn beide Spieler keine Züge mehr haben. Gewonnen hat dann derjenige Spieler mit den meisten Steinen auf dem Brett.

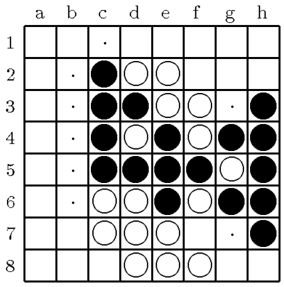
Bis jetzt gibt es sehr wenig nicht japanische Literatur über Othello. Empfehlenswert sind die Einführung von LANDAU (1987) und die Journale der Othello–Verbände.



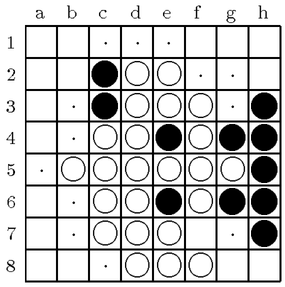
Startposition

	a	b	c	d	e	f	g	h
1		C	A	B	B	A	C	
2	C	X					X	C
3	A							A
4	B							B
5	B							B
6	A							A
7	C	X					X	C
8		C	A	B	B	A	C	

Feldbezeichnungen



Weiß am Zug



Stellung nach Zug b5

Anhang B

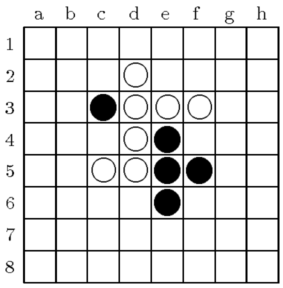
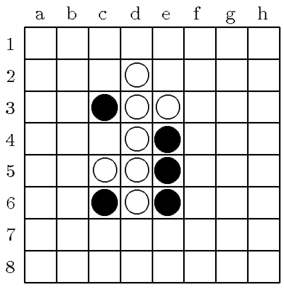
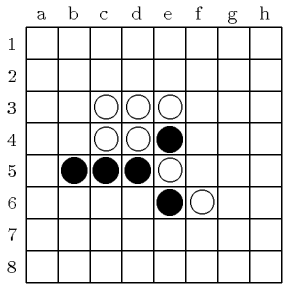
LOGISTELLO

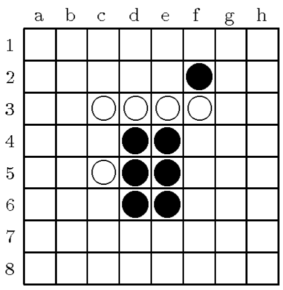
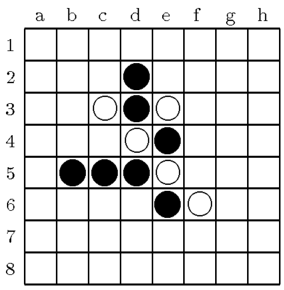
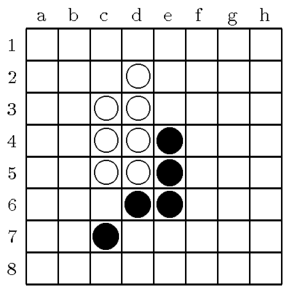
Suchverfahren	– NegaScout mit selektiven Erweiterungen – Ruhesuche – iterative Vertiefung
Zugvorsortierung	– Hashtabelle mit Zügen und Bewertungen – Response–Killer Listen – flache Suchen
Merkmale	– aktuelle Mobilität – potentielle Mobilität – Horizontalen/Vertikalen–Muster der Länge 8 – Diagonalen–Muster der Längen 5–8 – (2×4)–Eckenmuster
Kombinationsfunktion	– linear – Parametersatz für jede Steinanzahl 12–62 bestimmt mittels logistischer Regression
Suchgeschwindigkeit	– Mittelspiel: ca. 30.000 Stellungen/Sekunde – Endspiel: ca. 100.000 Stellungen/Sekunde
Suchiefe	in einer 30 Minuten Partie: – Mittelspiel: regelmäßig 12–15 (selektiv, ohne Ruhesuche) – Endspiel: Start regelmäßig bei 41–43 Steinen
Bibliothek	– derzeit ca. 2.000 Spiele – wird automatisch erweitert
Sonstiges	– Rechnen auf gegnerische Zeit – Kommunikation mit IOS über Socket oder mit grafischer Oberfläche über Filesystem
Sprache/Compiler	C/gcc
Rechner	Sun–SPARC10/M20

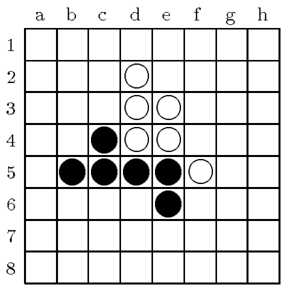
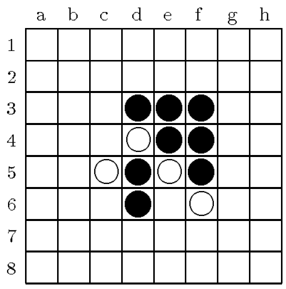
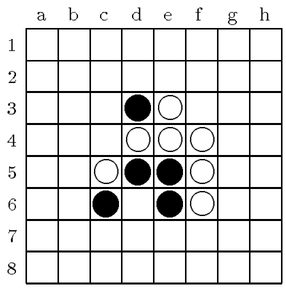
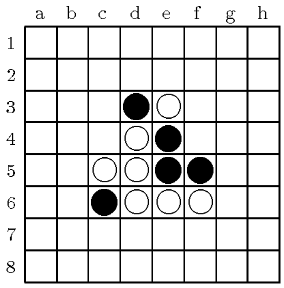
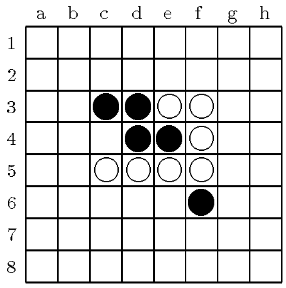
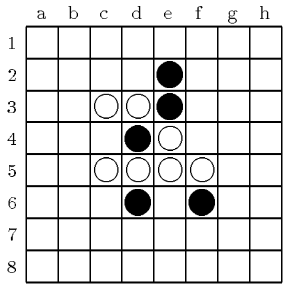
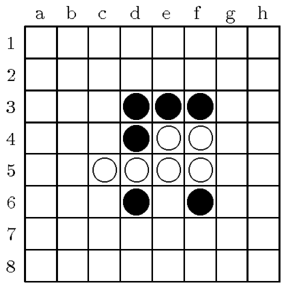
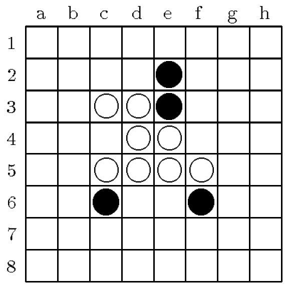
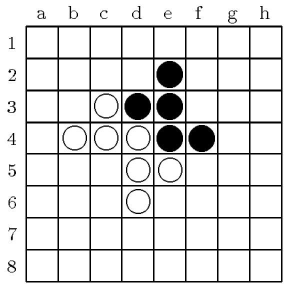
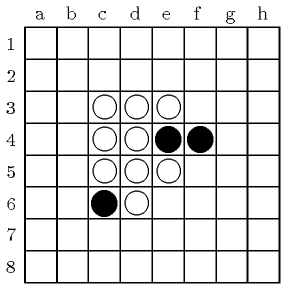
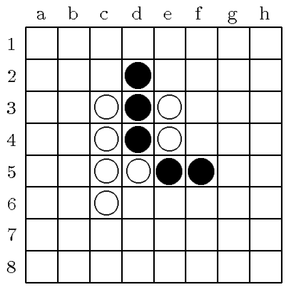
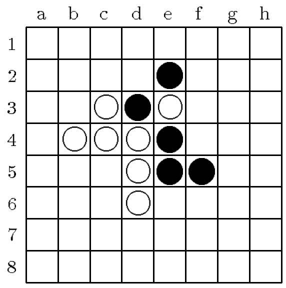
Anhang C

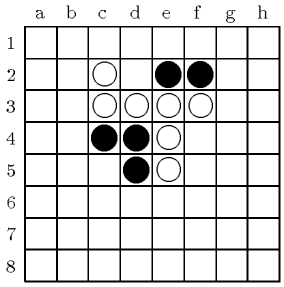
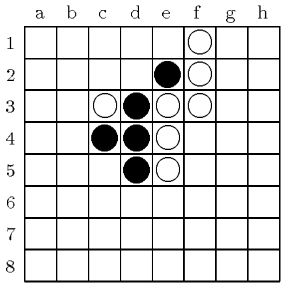
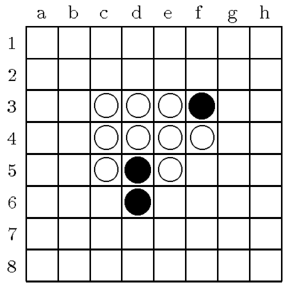
Startpositionen

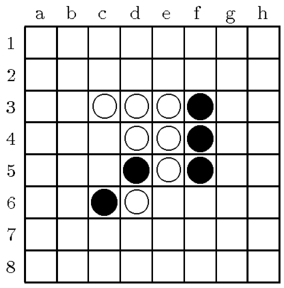
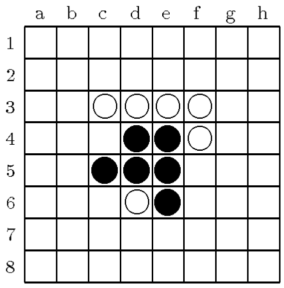
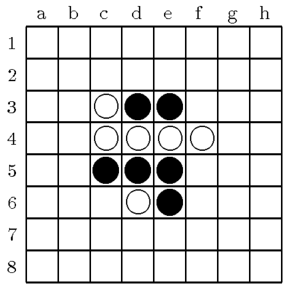
Die hier aufgelisteten Othello–Stellungen aus der Eröffnungsphase (Schwarz am Zug) dienten als Startpositionen für die Turnierpartien zum Spielstärkevergleich.

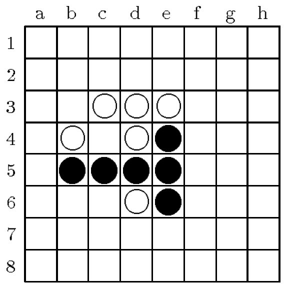
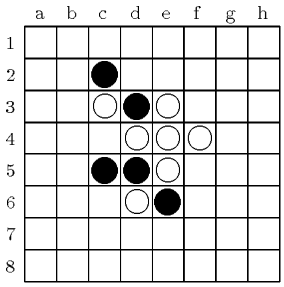
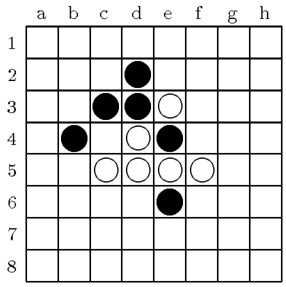




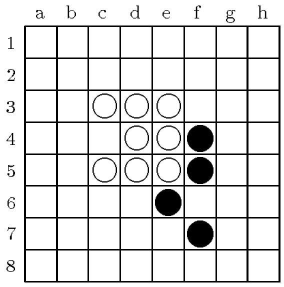
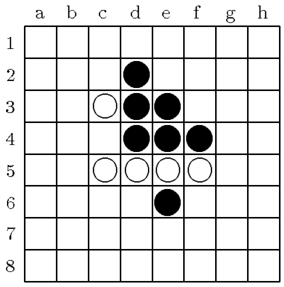
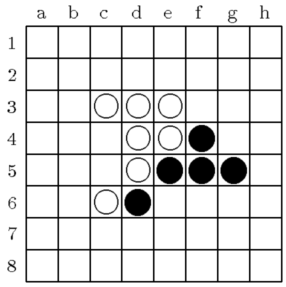




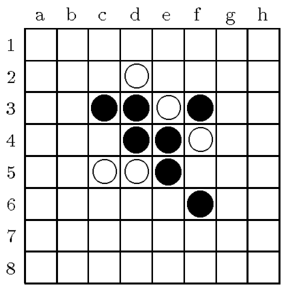













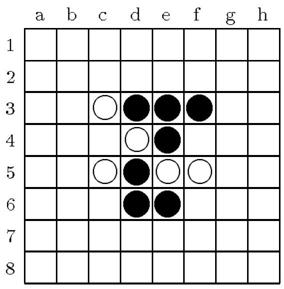




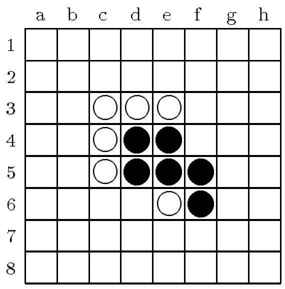




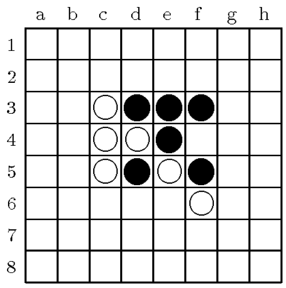




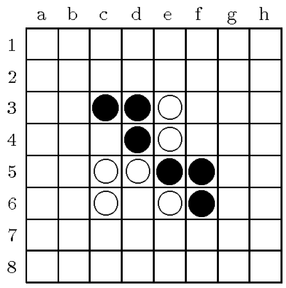










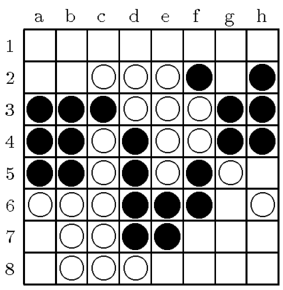




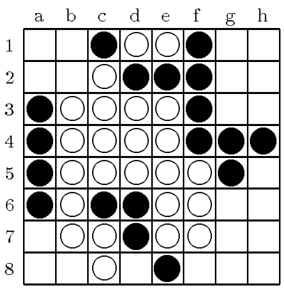
Anhang D

Endspielpositionen

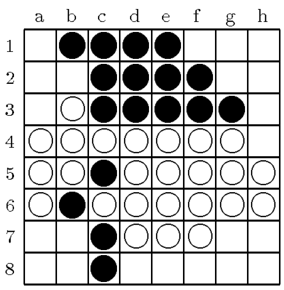
Im folgenden sind 35 für Schwarz am Zug gewonnene Endspielstellungen aufgelistet, bei denen LOGISTELLOs heuristische Suche vor Beginn der Entscheidungssuche nur einen Verlustzug ermittelte. Sie wurden zur Bestimmung der Güte verschiedener Versionen der selektiven Endspielsuche benutzt.



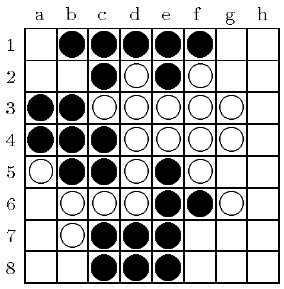
1. c1? f1!



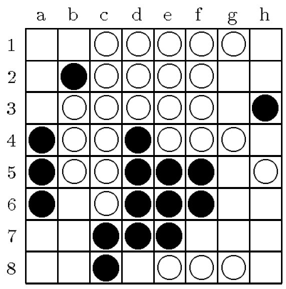
2. f8? g6!



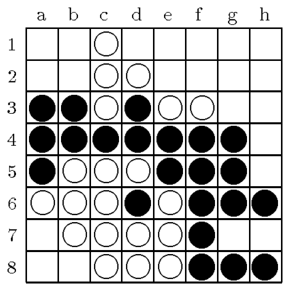
3. b7? e8!



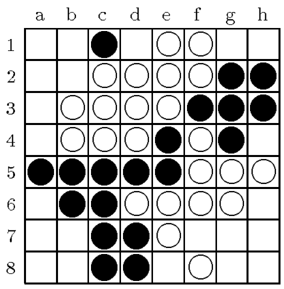
4. g1? g5!



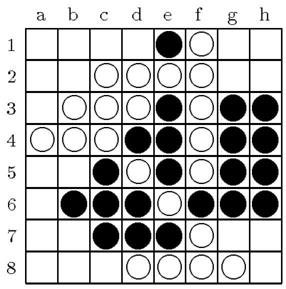
5. g3? a3!



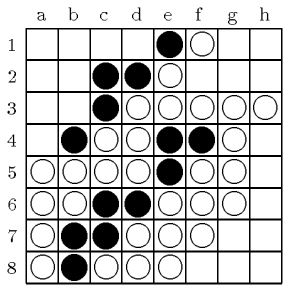
6. b2? e2!



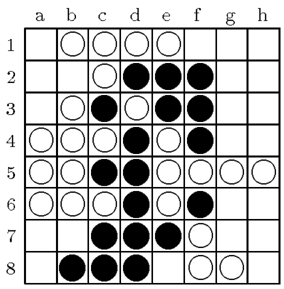
7. e8? a4!



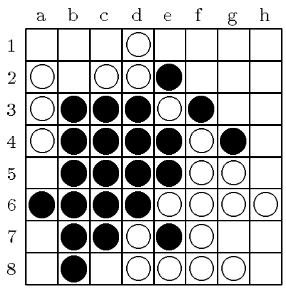
8. b2? a3!



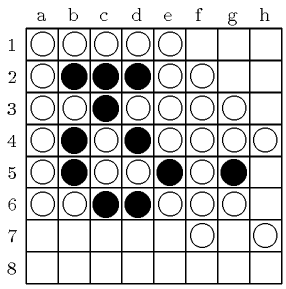
9. f2? a4!



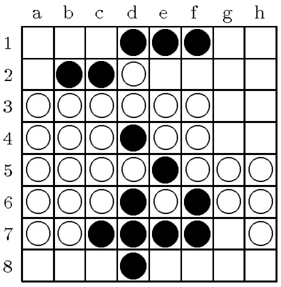
10. a2? h6!



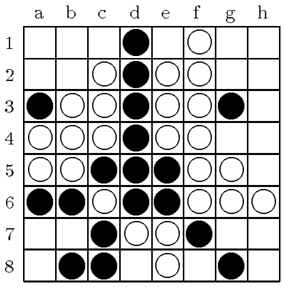
11. g3? h4!



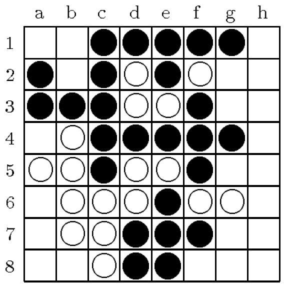
12. e7? h3!



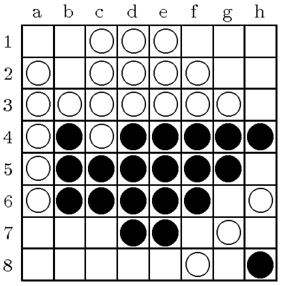
13. e2? g3!



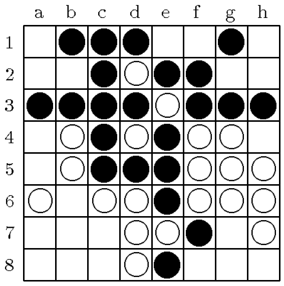
14. g7? c1!



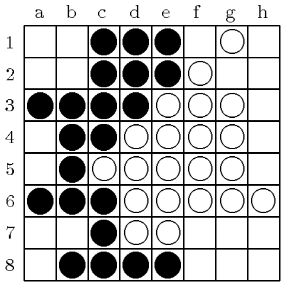
15. a8? g5!



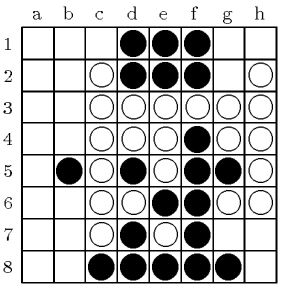
16. b2? g1!



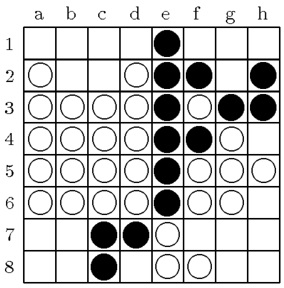
17. h4? c8!



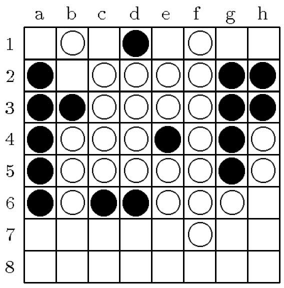
18. f7? f8!



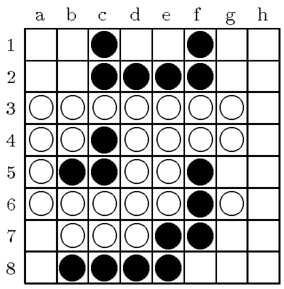
19. b8? b4!



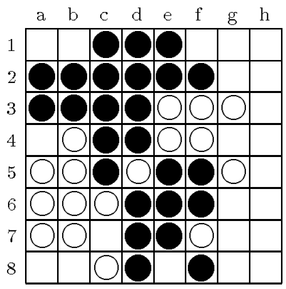
20. c1? h7!



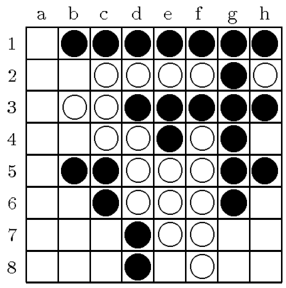
21. h6? c1!



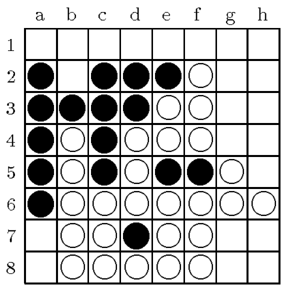
22. h6? g5!



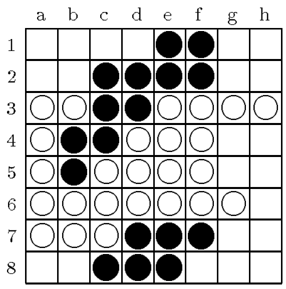
23. c7? g4!



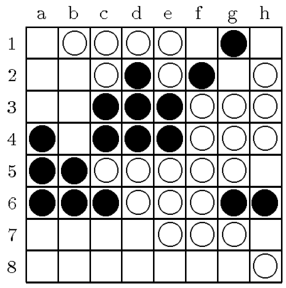
24. b4? a3!



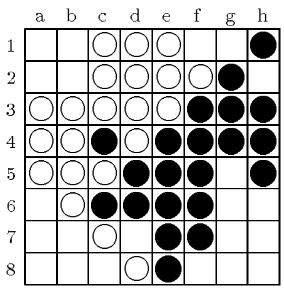
25. g3? g4!



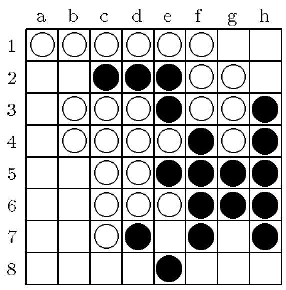
26. h5? g7!



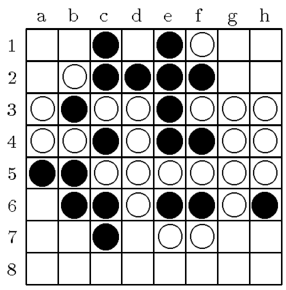
27. h5? d7!



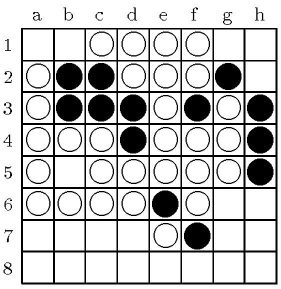
28. a6? c8!



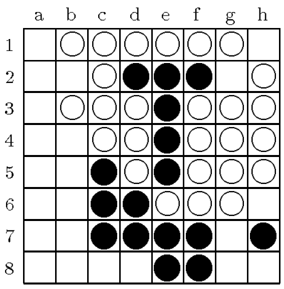
29. b5? e7!



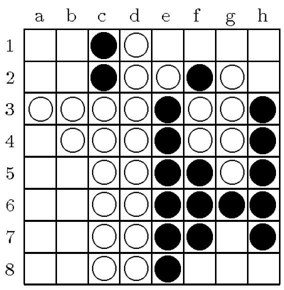
30. h2? d7!



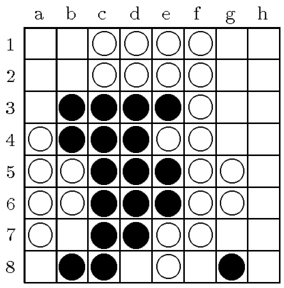
31. d7? g1!



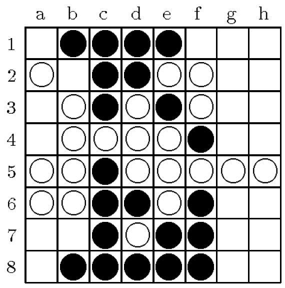
32. b5? g7!



33. e1? f1!



34. h5? g4!



35. f1? h6!

Anhang E

Turnierergebnisse

Paderborn 1993

(Computer–Turnier, Universität–GH–Paderborn)

	Programm	Hardware	Autor(en)	Punkte gegen die besten 8
1.	LOGISTELLO	SPARC10[1]	M. Buro	12.5
2.	KEYANO	SPARC10	M. Brockington	9.5
3.	REV	SPARC10	I. Đurđanović	8
4.	VERS2	486DX2/66	B. de Wolf	7.5
5.	OTHEL DU NORD	486DX/50	J.C. Delbarre	6
6.	MODOT	SPARC10	J.F. Feinstein	5
	DESDEMONA	spezial[2]	L. Johansson O. Liljedahl I. Lindgren	5
8.	THOR	486DX/50	S. Quin	2.5
9.	OOT	spezial[3]	M. Poysti	1
10.	COUNTMAX	486DX/50	E.L. Grau	0
11.	MONYCA	486DX/50	A. Gomez–Perez C. Linares–Lopez	0
12.	ADAMS OTHELLO	486DX/50	B.T. Gray	0

[1] Eine SPARC10/M20 ist ungefähr doppelt so schnell wie ein 486DX2/66–PC

[2] PC–Steckkarte, besucht über 250.000 Stellungen pro Sekunde

[3] PC–Steckkarte, bewertet ca. 1.500 Stellungen pro Sekunde

Waterloo 1993

(Computer–Turnier, University of Waterloo/Internet)

	Programm	Punkte aus 4 Spielen
1.	STELL	3.5
	LOGISTELLO	3.5
3.	BRUTUS	3
	KEYANO	3
	REV	3
	NICKSREV	3
	VERS2	3
8.	REV6.5	2.5
9.	RODIONOV	2
	OOT	2
	CHIGOREV	2
12.	PEEWEE	1.5
13.	SDC	1
	CHOJIN	1
	ZEROTH 17	1
	HANS N FRANS	1
17.	MODOT	0
	DUMB	0
	PC'ED	0
	IAGO	0

Waterloo–Finale 1993

(Computer–Turnier, Internet)

	Programm	Hardware	Autor(en)	Punkte aus 10 Spielen
1.	LOGISTELLO	SPARC10	M. Buro	6.5
2.	KEYANO	SPARC10	M. Brockington	5.5
3.	REV	SPARC10	I. Đurđanović	5
	BRUTUS	486DX2/66	L. Geoffroy M. Piotte	5
5.	STELL	486DX/33	D. Petrovsky	4
	VERS2	486DX2/66	B. de Wolf	4

1. IOS–Turnier 1994

(gemischtes Turnier, Internet)

	Spieler/Programm	etw. Hardware	etw. Autor(en)	Punkte aus 5 Spielen
1.	REV	SPARC10	I. Đurđanović	4.5
2.	LOGISTELLO	SPARC10	M. Buro	4
	C. Springer			4
	OTHELLO MASTER	CM5[4]	J.C. Weill	4
	ENIGMA	486DX2/66	M. Giles C. Springer	4
6.	BRUTUS	486DX2/66	L. Geoffroy M. Piotte	3
	KEYANO	SPARC10	M. Brockington	3
	J.F. Feinstein			3
	HAMMER	486DX/50	M. Giles	3
	M. Masten			3
11.	OOT	spezial	M. Poysti	2.5
	C. Brisson			2.5
13.	VERS2	486DX2/66	B. de Wolf	2
	M. Brockington			2
	D.J. Summers			2
	O. Berthier			2
	R. Gatliff			2

[4]Parallelrechner mit der Leistung von ca. 16 SPARC10

Courchelettes 1994

(Computer–Turnier, Nordfrankreich)

	Programm	Hardware	Autor	Punkte aus 7 Spielen
1.	LOGISTELLO	SPARC10	M. Buro	7
2.	POLYGON	ACORN[5]	A. Selby	5
	OTHEL DU NORD	486DX2/66	J.C. Delbarre	5
4.	SPOCK	PENTIUM/66	J. Delteil	4.5
5.	CASSIO	68040/25	S. Nicolet	4
	REVOTH	486DX2/66	E. Jerome	4
	VERS2	486DX2/66	B. de Wolf	4
	PUREE	486SX/25	O. Thill	4
9.	THOR	486SL/33	S. Quin	3.5
10.	THEOLE	486DX2/66	N. Becquet	3
	GROS'THELLO	486SX/25	S. Pinta	3
	TOM POUCE	68000/8	B. Andriani	3
13.	ASOTH	486DX/33	J.F. Coulon	2
	JAMOTH	486DX/33	G. Rump	2
15.	O	386DX/25	S. Koudache	1
	INTHELLO	486DX/33	B. Bras	1

IOS–Computer–Turnier 1994

(Internet)

	Programm	Hardware	Autor(en)	Punkte aus 12 Spielen
1.	LOGISTELLO	SPARC10	M. Buro	11
2.	ECLIPSE	RS6000[6]	M. Giles C. Springer	8
3.	PEEWEE	PENTIUM/90	A. Whinery	7
	REV6.8	PENTIUM/60	D. Parsons	7
5.	OOT	spezial	M. Poysti	3.5
6.	YINGYANG	PENTIUM/90	V. Sempranio	3
7.	CHOJIN	RS6000	P. Rechstein	2.5

[5] ACORN ARCHIMEDES, ungefähr so schnell wie ein 486DX/33–PC
[6] hat ungefähr die Leistung von drei 486DX2/66–PCs

Paderborn 1994

(Computer–Turnier[7], Universität–GH–Paderborn)

	Programm	Hardware	Autor(en)	Punkte/Spiele
1.	LOGISTELLO	SPARC10	M. Buro	11/12 (ø15.2)
2.	ECLIPSE	SPARC10	M. Giles C. Springer	11/12 (ø11.7)
3.	KITTY	SPARC10	I. Đurđanović	6.5/12
4.	BUGS	SPARC10	J.C. Weill	5.5/12
5.	KEYANO	SPARC10	M. Brockington	5.5/11
6.	SPOCK	486DX/50	J. Delteil	5.0/11
7.	BALOO	SPARC10	J.C. Weill	4.5/11
8.	WIGALD	SPARC10	R.T. Brunner	4.0/11
9.	VERS2	486DX2/66	B. de Wolf	3.0/11
10.	OOT	spezial	M. Poysti	1.0/11

[7]Dieses Turnier fand Anfang Oktober — also nach Abgabe der Arbeit — statt