# Exakte Analyse von Heuristiken für kombinatorische Spiele

**Dissertation**

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik der
Friedrich-Schiller-Universität Jena

von Diplom-Physiker Thomas Fischer
geboren am 21. Juni 1978 in Jena

**Gutachter**

**1. Prof. Dr. Ingo Althöfer, Friedrich-Schiller-Universität Jena**

**2. PD Dr. Ulrich Tamm, Marmara-Universität Istanbul, Türkei**

**Tag der öffentlichen Verteidigung: 6. April 2011**

# Zusammenfassung

In der vorliegenden Arbeit werden Heuristiken für kombinatorische Spiele untersucht. Jedoch erfolgt dies nicht mit statistischen Methoden, d. h. durch Simulation von Spielverläufen, sondern durch exakte Analyse der Heuristiken. Je nach Spiel und Heuristik sind dabei unterschiedliche Hürden zu überwinden.

Solche exakten Analysen sind zwar zum Teil recht aufwändig, bieten aber gleichzeitig mehrere Vorteile: Neben den rein theoretischen Überlegungen, sind auch die computergenerierten Ergebnisse (je nach verwendeter Arithmetik beliebig) exakt, so dass sich daraus sichere Aussagen ableiten lassen. Zum anderen lassen sich einige der auftretenden Phänomene durch das tiefere Problemverständis besser interpretieren. Schließlich ist sogar der Rechenaufwand der exakten Analysen z. T. deutlich geringer, als der bei Simulationsexperimenten für statistisch signifikante Ergebnisse.

In diesem Kontext wird für einfache kombinatorische Spiele eine Methode zur Analyse von Monte-Carlo-basierten Agenten vorgestellt, die es erlaubt, die Siegquoten der Agenten exakt vorherzusagen. Zum Einsatz kommt diese Methode anschließend bei der Analyse von Self-Play-Experimenten, welche Rückschlüsse auf Spielstärke und Eigenheiten der Agenten bzw. Eigenschaften des Spiels erlauben.

Für Monte-Carlo-Agenten hat die exakte Analyse (ebenso wie bei Simulationsexperimenten) eine lineare Zeitkomplexität in Abhängigkeit von der Simulationsanzahl des Monte-Carlo-Agenten. Der Grenzfall für unendliche Simulationsanzahlen bei Monte-Carlo-Agenten und asymmetrischer Varianten wird für ein einfaches Subtraktionsspiel daher theoretisch behandelt.

In manchen Fällen ist selbst die exakte Implementierung der Heuristik eine Herausforderung. So konnten beim Deduktionsspiel „Beams“ bisherige Computer-Agenten deutlich verbessert werden, indem die Entropie der Anfragen statt näherungsweise geschätzt nun exakt berechnet wurde. Die Entropie-Heuristik wird auf Rényi-Entropien verallgemeinert und ein Kriterium für die Optimalität solcher Entropien für Deduktionsspiele angegeben. Dieses wird anhand verschiedener Spiele bzw. Modelle experimentell verifiziert.

Im letzten Teil der Arbeit wird eine Widerstands-Netzwerk-Heuristik für das Shannon-Switching-Game untersucht. Für verschiedene Graphgrößen bei diesem Spiel wird die Heuristik kombinatorisch analysiert. Um Verlustzüge der Heuristik zu finden, kommen u. a. wieder Monte-Carlo-Agenten zum Einsatz. Deren Verhalten gibt Anlass zur näheren Analyse von Zufallseinflüsse auf Heuristiken. Daraus wird schließlich ein Modell zur künstlichen Randomisierung von Heuristiken abgeleitet und – bisher aber experimentell – untersucht.

# Inhaltsverzeichnis

**Zusammenfassung** iii

**1 Einführung** 1
1.1 Reine Monte-Carlo-Agenten . . . 2
1.2 Zum Aufbau der Arbeit . . . 3
1.3 Danksagung . . . 4

**2 Exakte Analyse von Monte-Carlo-Agenten bei Klassen einfacher Spiele** 5
2.1 Einzelschritt-Rennen . . . 5
2.2 Monte-Carlo-Agenten . . . 6
2.3 Self-Play-Bassin-Strukturen . . . 7
2.4 Exakte Analyse reiner Monte-Carlo-Agenten . . . 8
2.4.1 Konfigurationen . . . 8
2.4.2 Zufallspartien . . . 9
2.4.3 Siegwahrscheinlichkeiten der Monte-Carlo-Experimente . . . 9
2.4.4 Partien mit Monte-Carlo-Agenten . . . 10
2.4.5 Ergebnisse für Einzelschritt-Rennen . . . 10
2.4.6 Schnellere Berechnung der Monte-Carlo-Erwartungswerte . . . 14
2.5 Analyse weiterer Spiele . . . 15
2.5.1 Doppelschritt-Crash-Rennen . . . 15
2.5.2 Zwei-Zug-Subtraktionsspiele . . . 21
2.6 Diskussion . . . 25
2.6.1 Einzelschritt-Rennen . . . 25
2.6.2 Doppelschritt-Crash-Rennen und Zwei-Zug-Subtraktionsspiele . 25
2.7 Ausblick und offene Fragen . . . 27

**3 Asymmetrische Monte-Carlo-Agenten für das (1, 2)-Subtraktionsspiel** 29
3.1 Das klassische Subtraktionsspiel $S_{(1,2)}$ . . . 29
3.2 Reiner Monte-Carlo-Agent . . . 30
3.3 Statisch-asymmetrische Monte-Carlo-Agenten . . . 30
3.3.1 Rekursionsgleichungen und charakteristisches Polynom . . . 31
3.3.2 Reelle Fundamentallösungen . . . 32
3.3.3 Komplexe Fundamentallösungen . . . 32
3.3.4 Ungünstige Züge des Monte-Carlo-Agenten . . . 33
3.4 Intelligent-asymmetrische Monte-Carlo-Agenten . . . 37
3.4.1 Reelle Fundamentallösungen . . . 39
3.4.2 Komplexe Fundamentallösungen . . . 41
3.5 Diskussion . . . 43
3.6 Ausblick und offene Fragen . . . 46

**4 Deduktionsspiele und Rényi-Entropie-Strategien 49**
4.1 Einleitung . . . 49
4.2 Rényi-Entropie und Heuristiken für Suchprobleme . . . 50
4.2.1 Die Rényi-Entropie . . . 50
4.2.2 Suchprobleme . . . 51
4.2.3 Perfekte Strategien . . . 52
4.2.4 Maximierung der Rényi-Entropie als lokale Heuristik . . . 55
4.2.5 Die besondere Rolle der Shannon-Entropie . . . 56
4.3 Mastermind . . . 60
4.3.1 Spielregeln von Mastermind . . . 60
4.3.2 Eigenschaften und Varianten von Mastermind . . . 60
4.3.3 Analyse von Mastermind . . . 61
4.4 Beams . . . 63
4.4.1 Spielregeln von Beams . . . 63
4.4.2 Varianten von Beams . . . 64
4.4.3 Implementierung der Entropie-Heuristik für Beams(6c,6p) . . . 66
4.4.4 Ergebnisse der Analyse von Beams . . . 67
4.4.5 Verifikation der Erwartungswerte durch Simulationen . . . 69
4.5 Versagen der Entropie-Heuristik . . . 70
4.5.1 Das Wurzel-Modell . . . 72
4.5.2 Abschätzung der Suchtiefe des Wurzel-Modells . . . 73
4.5.3 Ergebnisse für das Wurzel-Modell . . . 76
4.6 Auswertung und Diskussion . . . 76
4.6.1 Deduktionsspiele im Vergleich . . . 76
4.6.2 Die besten Rényi-Strategien . . . 78
4.6.3 Vergleich mit approximierten Entropien bei Beams . . . 79
4.6.4 Zum Wurzel-Modell . . . 80
4.7 Ausblick und offene Fragen . . . 81

**5 Analyse von Shannons Analog-Heuristik für das Switching-Game 83**
5.1 Einführung . . . 83
5.1.1 Das Shannon-Switching-Game . . . 83
5.1.2 Perfekte Strategien . . . 84
5.1.3 Planare Graphen und Dualität . . . 85
5.2 Shannons analoge Heuristik . . . 87
5.2.1 Das Widerstandsmodell . . . 87
5.2.2 Das Netzwerk als Gleichungsystem . . . 88
5.2.3 Dualität der Heuristik . . . 89
5.2.4 Direkte und Iterative Löser . . . 91
5.2.5 Implementierung von Computer-Agenten . . . 92
5.3 Ist die Shannon Heuristik optimal? . . . 92
5.3.1 Gewinnen gegen die Heuristik . . . 92
5.3.2 Shannon vs. Shannon . . . 93
5.4 Randomisierung . . . 96
5.4.1 Monte-Carlo vs. Shannon-Heuristik . . . 96
5.4.2 Künstliche Randomisierung . . . 97

5.4.3 Ergebnisse mit künstlicher Randomisierung . . . . . 98
5.4.4 Ergebnisse mit Randomisierung durch finite Iteration . . . . . 100
5.5 Diskussion . . . . . 100
5.6 Ausblick und offene Fragen . . . . . 102

**A Deduktionsspiele** **103**
A.1 Beweis eines Lemmas zum Wurzel-Modell . . . . . 103
A.2 Beispiel für den Spielbeginn von Beams . . . . . 105
A.3 Beams: Offizielle Spielregeln und Varianten von Michail Antonow . . . 106

**B Codes** **111**
B.1 Vorbemerkungen . . . . . 111
B.2 Siegwahrscheinlichkeiten für MC-Analyse . . . . . 112
B.3 Exakte Analyse von Subtraktionsspielen . . . . . 114

**Quellen- und Literaturverzeichnis** **117**

**Ehrenwörtliche Erklärung** **121**

# Kapitel 1

# Einführung

Spiele haben die Menschen schon immer fasziniert. Obwohl jeder einzelne ganz unterschiedliche Vorlieben hat, gibt es kaum jemanden, der nicht gern spielt. Welche Spiele man mag, hängt allerdings von einem selbst und den Eigenschaften der Spiele ab.

Beim Spielen geht es neben dem Vergnügen natürlich auch darum, zu gewinnen. Durch die Art und Weise, wie man Spiele gewinnen kann, lassen sich Spiele in gewissen Rahmen klassifizieren. Fragestellungen, die für eine solche Charakterisierung herangezogen werden können sind beispielsweise:

- Wie stark hängt der Spielausgang vom Zufall ab, kann man also mit Glück gewinnen?
- Gibt es gute Strategien? Kann man durch Spielerfahrung bzw. Training erfolgreicher spielen?
- Gibt es einfache Strategien (Heuristiken), die gut sind?

Charakterisierungen dieser Art[1] sind beispielsweise für das automatisierte Erfinden und Bewerten von Spielen relevant, [15], [3], [21].

Eine beständige Herausforderung ist die Automatisierung intelligenten Spielverhaltens. Schon vor der Verfügbarkeit leistungsstarker Computer gab es Versuche, einfache Heuristiken technisch umzusetzen. Anfang der 1950er Jahre entwickelte Claude Shannon (zusammen mit Edward Moore) Automaten, die das Spiel Bridge-It[2] und die $7 \times 8$-Variante des Spiels Hex spielen konnten [45].

Inzwischen hat sich die Künstliche Intelligenz als eingenständige Forschungsrichtung etabliert und eine Vielzahl von Konzepten zur Umsetzung von starken computergestützten Agenten hervorgebracht.

[1] Darüber hinaus spielen auch persönliche Fähigkeiten wie Geschick, Kommunikationsfähigkeit oder Sportlichkeit ein Rolle.

[2] Shannons analoge Heuristik für Bridge-It behandeln wir in Kapitel 5.

Doch immer wieder sind es einfache Ideen, die erstaunliche Resultate hervorbringen. In der vorliegenden Arbeit werden wir einige dieser Konzepte, nämlich maximale Entropie, maximaler Netzwerkfluss und Monte-Carlo-Agenten, analysieren.

Eine richtungsweisende Arbeit zum Einsatz von Monte-Carlo-Algorithmen stammt aus dem Jahre 1993 von Bernd Brügmann [17]. Da Monte-Carlo-Agenten in mehreren Kapiteln dieser Arbeit eine wichtige Rolle spielen, stellen wir das Konzept schon hier im Einführungsteil vor.

## 1.1 Reine Monte-Carlo-Agenten

Zur Konstruktion von Monte-Carlo-Agenten genügt es, die Spielregeln einschließlich der Siegbedingung zu kennen. Weitere spielstrategische Überlegungen sind nicht nötig.

**Definition 1.1** (Monte-Carlo-Agent MC($k$)). *Der reine Monte-Carlo-Agent mit Monte-Carlo-Parameter $k$, kurz MC($k$), ermittelt in der aktuellen Spielstellung alle regelkonformen Zugkandidaten und simuliert für jeder dieser Nachfolgestellungen genau $k$ Zufallspartien bis zum Ende des Spiels (vgl. Abbildung 1.1).*

*Derjenige Zugkandidat, für den die meisten Zufallspartien gewonnen werden, wird als Zug ausgewählt. Bei Gleichstand wird unter allen „besten“ Zugkandidaten einer gemäß Gleichverteilung ausgewählt.*

Für den Grenzfall $k \to \infty$ (Gesetz der großen Zahlen) definieren wir den asymptotischen Monte-Carlo-Agenten.

**Definition 1.2** (Monte-Carlo-Agent MC($\infty$)). *Der asymptotische Monte-Carlo-Agent MC($\infty$) wählt denjenigen Zug aus, in dessen folgenden Spielverlauf eine Zufallspartie die höchste Siegwahrscheinlichkeit hat. Bei gleichwertigen besten Zugkandidaten wird einer dieser Züge gemäß Gleichverteilung ausgewählt.*

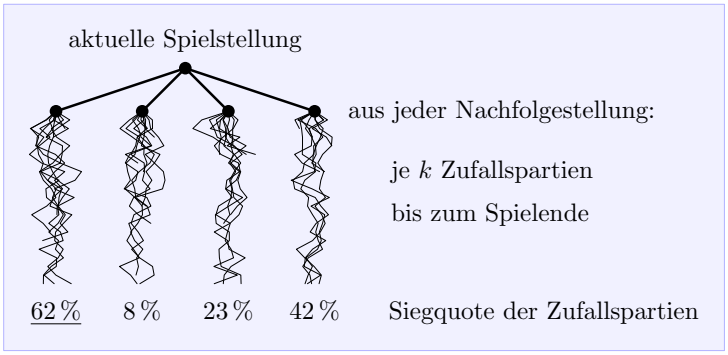



**Abbildung 1.1:** Monte-Carlo-Agent MC($k$)

Es sei angemerkt, dass Agent MC($\infty$) keinesfalls immer einen optimalen Zug vorschlägt (siehe Kapitel 2 und 3). Außerdem sind reine Monte-Carlo-Agenten nicht die beste Methode, den Zugkandidaten mit der höchsten Siegwahrscheinlichkeit zu finden [10].

## 1.2 Zum Aufbau der Arbeit

Der Einführung folgen vier Hauptkapitel. Jedes dieser Kapitel beginnt mit einer kurzen Zusammenfassung und endet mit einer Diskussion der Ergebnisse sowie einem Ausblick. Darin sind sowohl weiterführende Fragestellungen, die sich für zukünftige Bachelor oder Master-Arbeiten eignen, als auch recht offen formulierte Probleme und konzeptionelle Ideen aufgezählt. Vorarbeiten, die in der Regel jeweils nur für einzelne Abschnitte maßgeblich sind, werden an den betreffenden Stellen genannt.

In Kapitel 2 untersuchen wir die Spielstärke von MC($k$)-Agenten, indem wir die Siegwahrscheinlichkeiten der Zufallspartien für einfache Spiele exakt berechnen.

Das Kapitel 3 widmet sich der theoretischen Analyse von MC($\infty$)-Agenten für das (1,2)-Subtraktionsspiel.

Im Kapitel 4 werden entropiebasierte Heuristiken für das Deduktionsspiel Beams untersucht und Kriterien für die Optimalität bestimmter Rényi-Entropien experimentell erforscht.

Zum Abschluss wird in Kapitel 5 eine Analog-Heuristik für das Verbindungsspiel Bridg-It untersucht und ein Konzept für randomisierte Heuristiken entwickelt.

Im Anhang A befinden sich neben dem Beweis eines technischen Lemmas aus Kapitel 4 die ausführlichen Spielregeln und ein Beispiel für das Deduktionsspiel Beams.

Im Anhang B folgt eine kurze Erläuterung zu den für diese Arbeit entwickelten Programm-Codes sowie Programmbeispiele zum Kapitel 2.[3]

Das Literatur- und Quellenverzeichnis befindet sich ganz am Ende der Arbeit ab Seite 117.

### Anmerkungen

Auf Abkürzungen wurde weitestgehend verzichtet. Die Monte-Carlo-Agenten MC($k$) bzw. MC($\infty$) sind in Kapitel 1 und die Spiele ESR($n, m, s$), DSC($n$) sowie $\mathrm{S}_{(a,b)}(n)$ in Kapitel 2 definiert.

Zum leichteren Nachschlagen der Definitionen, Sätze, Beispiele, etc. folgen diese einer gemeinsamen Nummerierung. Ebenso sind Abbildungen und Tabellen mit einem gemeinsamen Zähler versehen. Bei Verweisen über mehrere Seiten ist häufig auch die

[3] Ein Datenträger mit allen für diese Arbeit relevanten Programm-Codes und berechneten Daten wird am Lehrstuhl Mathematische Optimierung hinterlegt.

Seitennummer mit angegeben. Auf ein separates Abbildungs- bzw. Tabellenverzeichnis wurde verzichtet.

Dezimalzahlen sind sowohl in den Abbildungen wie auch im Text einheitlich mit Dezimalpunkt geschrieben, um sie von Aufzählungen, Intervallen, etc. zu unterscheiden.

Daten, die mehrere Größenordnungen umfassen, sind in den Diagrammen logarithmisch dargestellt. In den Diagrammen und Abbildungen sind wichtige Zusammenhänge durch den Einsatz von Farben herausgearbeitet. Dennoch wurde versucht, durch Beschriftungen, Linienstile und kontrastreiche Farben, das Manuskript auch für den Schwarz-Weiss-Audruck bzw. zum Kopieren geeignet zu gestalten. Insbesondere wurden bei Farbverläufen solche mit monotoner Helligkeitsskala gewählt.

Die angegebenen Rechenzeiten sind nur zur groben Orientierung gedacht. Sie sind natürlich hardware- und compilerabhängig. Gerechnet wurde hauptsächlich auf mehreren Vierkern-Maschinen (Intel-Core2-Q9300-CPU mit 2.5 GHz und 4GB RAM) im Institut für Angewandte Mathematik der FSU Jena. Als Compiler kamen die GNU-Compiler-Collection (http://gcc.gnu.org) bzw. der ebenfalls freie G95-Fortran-Compiler (http://www.g95.org) zum Einsatz.

In dieser Arbeit verwendete Markennamen und Produktbezeichnungen sind Eigentum bzw. eingetragene Markenzeichen der jeweiligen Personen, Firmen oder Einrichtungen.

## 1.3 Danksagung

Zunächst möchte ich mich bei meinem Betreuer Prof. Dr. Ingo Althöfer für die intensive fachliche Betreuung während der Bearbeitung dieses interessanten Themenkomplexes bedanken. Hervorzuheben sind dabei die regelmäßigen anregenden Gespräche und konstruktiven Diskussionen. Sehr hilfreich war die Möglichkeit, regelmäßig im Oberseminar über die aktuelle Forschung vortragen und diskutieren zu können.

Außerdem danke ich Prof. Dr. Wolfgang Kinzel für die Einladung, um am Lehrstuhl Theoretische Physik III der Universität Würzburg vorzutragen, sowie für die anschließende intensive Diskussion mit ihm und seinen Mitarbeitern über das Konzept der künstlichen Randomisierung.

Für das Korrekturlesen einzelner Teile dieser Arbeit bedanke ich mich bei den Doktoranden des Lehrstuhls Mathematische Optimierung, namentlich Lisa Schreiber und Tim Lucke genannt Schönberg.

Den Kollegen vom Institut für Angewandte Mathematik danke ich für die nette Zusammenarbeit ebenso wie meiner Familie und meinen Freunden für die Unterstützung und Begleitung während der Promotionsphase.

Zu guter Letzt bedanke ich mich für die finanzielle Förderung durch das 3-Hirn-Stipendium.

# Kapitel 2

# Exakte Analyse von Monte-Carlo-Agenten bei Klassen einfacher Spiele

*Bei der Analyse von Spielen und Computer-Agenten kommen häufig sogenannte Self-Play-Experimente zum Einsatz, [27]. Dazu lässt man unterschiedlich starke Versionen eines Computer-Programms gegeneinander spielen, was insbesondere eine hohe Anzahl von Partien ermöglicht. Aus den Ergebnissen möchte man Rückschlüsse auf Eigenheiten der Agenten bzw. Eigenschaften des Spiels ziehen, [3], [15].*

*In diesem Kapitel wollen wir das Spielverhalten von Monte-Carlo-Agenten an sehr einfachen Spielen analysieren und eine Möglichkeit aufzeigen, mit endlichem Aufwand die Siegwahrscheinlichkeiten der Agenten exakt zu berechnen.*

## 2.1 Einzelschritt-Rennen

Als Modell betrachten wir in diesem Kapitel zunächst Einzelschritt-Rennen.

**Definition 2.1.** *Ein Einzelschritt-Rennen, kurz ESR$(n, m, s)$, ist ein Zwei-Personen-spiel mit folgenden Regeln (vgl. Abbildung 2.1):*

- *Jeder Spieler hat $m$ Bahnen der Länge $n$, auf deren Startfeldern (Position 0) jeweils eine Figur steht.*
- *Die Spieler ziehen abwechselnd jeweils eine beliebige eigene Figur ein Feld vor (Richtung Ziel), passen ist nicht zulässig.*
- *Wer zuerst $s$ ($s \leq m$) seiner Figuren in ihren Zielfeldern (Position $n$) hat, gewinnt das Spiel.*

Es ist leicht einzusehen, dass der anziehenden Spieler eine Gewinnstrategie hat. Dazu wählt er genau $s$ seiner Figuren aus und zieht diese in beliebiger Zugfolge bis zu ihren Zielfeldern. Weil keine Züge „vergeudet“ werden, gewinnt der Spieler nach genau $n \cdot s$ eigenen Zügen.

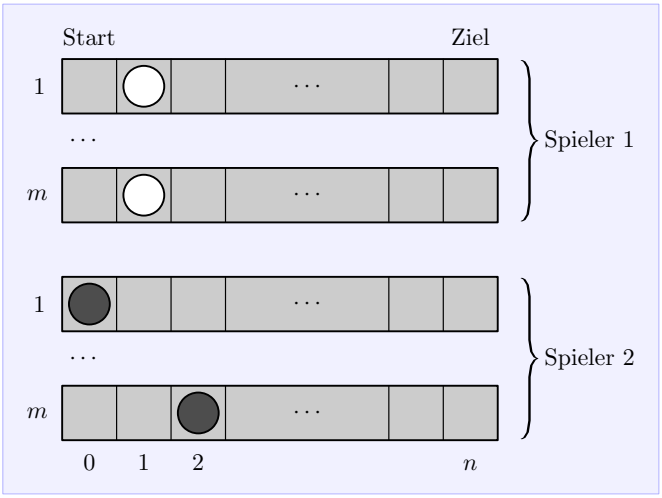



**Abbildung 2.1:** Einzelschritt-Rennen ESR$(n, m, s)$.

### Anmerkung

In [4] und [6] wurde ein ähnliches Modell, sogenannte Doppelschritt-Rennen, untersucht. Beim Doppelschritt-Rennen gibt es nur eine Bahn pro Spieler, jedoch dürfen die Spieler mit ihrer Figur wahlweise ein oder zwei Felder vorziehen. Eine Variante des Doppelschritt-Rennens werden wir in Abschnitt 2.5.1 analysieren.

## 2.2 Monte-Carlo-Agenten

Ein Monte-Carlo-Agent (vgl. Abschnitt 1.1), der nur die Spielregeln einschließlich der Siegbedingung kennt, findet diese Siegstrategie jedoch nicht so einfach. Ein solcher Agent mit Monte-Carlo-Parameter $k$, kurz MC$(k)$, ermittelt in der aktuellen Stellung alle gültigen Züge und simuliert im Anschluss an jeden dieser Züge jeweils $k$ Zufallspartien bis zum Ende des Spiels. Tatsächlich gespielt wird dann derjenige Zug, bei dem die meisten Simulationen gewonnen wurden. Bei Gleichstand wird unter allen „besten“ Zugkandidaten einer gleichwahrscheinlich ausgewählt.

Für das Einzelschritt-Rennen lässt sich induktiv zeigen, dass der Agent MC$(k)$ für $k \to \infty$ tatsächlich einen optimalen Zug auswählt. Für andere Spiele muss dies nicht unbedingt zutreffen. Beispiele dafür sind Subtraktionsspiele oder das Double-Step-Crash-Rennen (siehe Abschnitt 2.5.1 und Abbildung 2.9).

Um den Anzugsvorteil auszugleichen wird im Folgenden bei den Self-Play-Serien der Startspieler abgewechselt, d. h. die Gesamtsiegquote ist der Mittelwert der Siegquoten als anziehender bzw. nachziehender Spieler.

## 2.3 Self-Play-Bassin-Strukturen

Beim Testen von Computer-Agenten spielen Self-Play-Experimente eine große Rolle. Eine einfache Möglichkeit für solche Experimente besteht darin, mehrere Instanzen desselben Algorithmus', jedoch mit unterschiedlichen Ressourcen, gegeneinander antreten zu lassen. Typische Ressourcen sind dabei Rechenzeit und Speicherplatz, welche sich beispielsweise durch die Größe bzw. Tiefe von Suchbäumen explizit vorgeben lassen.

Im Falle reiner Monte-Carlo-Agenten ist nur die Rechenzeit relevant, die über die Zahl der Zufallspartien je Zugkandidat gesteuert werden kann.

Es hat sich als praktisch herausgestellt, Agenten zu vergleichen, die sich in ihren Ressourcen um den Faktor zwei unterscheiden. Dabei konnte in vielen Fällen ein etwa konstanter Spielstärkezuwachs über ein großes Ressourcenintervall beobachtet werden. Eine ausführliche Diskussionen am Beispiel von Schach findet man in [27, Kap. 7].

Ein typisches Beispiel für eine solche Spielstärkeabhängigkeit zwischen zwei Monte-Carlo-Agenten MC($k$) und MC($2k$) zeigt Abbildung 2.2. Diese Daten entstanden durch Simulation von 100 000 Partien mit Startspieler 1 und 100 000 Partien mit Startspieler 2 für jedes $k$. Wir interessieren uns vornehmlich für die zwischen Hin-

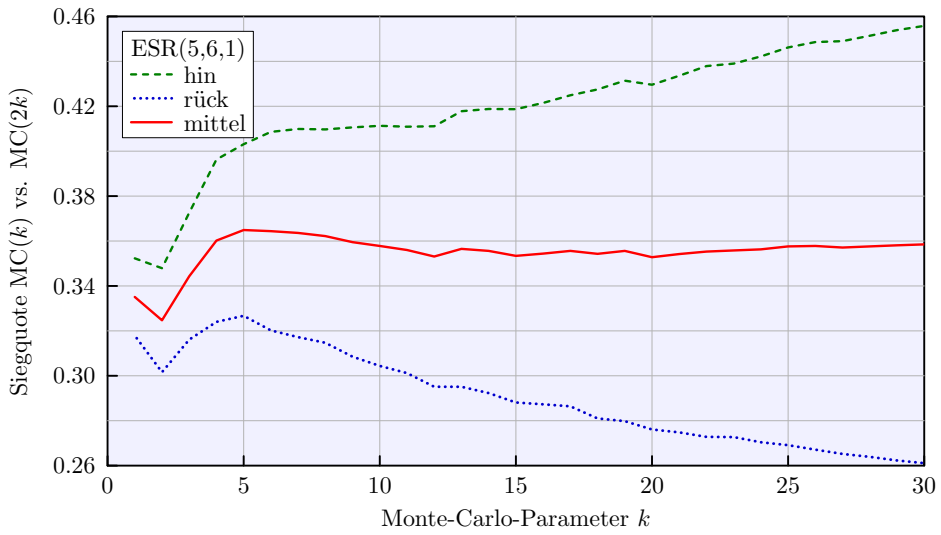



**Abbildung 2.2:** Simulation von Einzelschritt-Rennen ESR($n=5, m=6, s=1$) mit zwei Monte-Carlo-Agenten MC($k$) vs. MC($2k$). Die Siegquote ist aus Sicht des „schwächeren" Spielers MC($k$) aufgetragen. Die Ergebnisse im Hinspiel sind durch den Anzugsvorteil deutlich besser als im Rückspiel. Im Bereich $4 \leq k \leq 30$ gewinnt der schwächere Agent nahezu konstant ca. 36 % der Partien, wobei trotz der simulationsbedingten Schwankungen ($2 \cdot 10^5$ Partien für jedes $k$) eine leichte Delle („Bassin") erkennbar ist.

und Rückspiel[1] gemittelte Kurve (durchgezogen), um dem Anzugsvorteil Rechnung zu tragen.

Bei sehr kleinen Werten von $k$ ($k \leq 5$), d. h. recht schwachen Agenten, gibt es ein kleines Tal mit Minimum bei $k = 2$. Nur von MC(1) : MC(2) nach MC(2) : MC(4) haben wir hier einen fallenden Bereich.

Weiterhin ist im Bereich $5 \leq k \leq 30$ eine eine leichte, langgezogene Delle („Bassin“) zu erkennen. Für große $k$, also starke Agenten, wird der Vorteil durch die doppelten Ressourcen kleiner, die Graphen für an- und nachziehenden Spieler nähern sich den spieltheoretischen Werten 1 und 0 an – dies werden wir später noch einmal genauer unter die Lupe nehmen (siehe Abschnitt 2.4.5 und Abbildung 2.5).

Es zeigt sich also, dass sich die Spielstärke nicht gleichmäßig mit dem Monte-Carlo-Parameter verbessert. Das o. g. Bassin innerhalb des Plateaus ist statistisch signifikant [6]. Das Treffen genauerer Aussagen, insbesondere Erkennen eventueller Substrukturen, würde einen wesentlich höheren Simulationsaufwand erfordern. Wir wollen hier einen anderen Weg einschlagen und Spielstärkegraphen wie in Abbildung 2.2 exakt[2] berechnen.

## 2.4 Exakte Analyse reiner Monte-Carlo-Agenten

### 2.4.1 Konfigurationen

In den oben definierten Einzelschritt-Rennen ESR$(n, m, s)$ hat jeder Spieler $m$ Steine, von denen jeder auf einer Bahn mit $n + 1$ Feldern platziert ist. Da es auf die Reihenfolge nicht ankommt, kann die Position jedes Spielers durch ein monotones $m$-Tupel über $\{0, \ldots, n\}$ ausgedrückt werden. Dabei ist die Summe der Elemente des Tupels gleich der Anzahl der bisher gemachten Züge. Es ergeben sich $L = \binom{n+m}{m}$ mögliche Positionen für die Steine eines Spielers[3].

Für zwei Spieler haben wir somit höchstens $L^2$ Konfigurationen, denn jede Konfiguration kann durch ein Paar $C = (C_A, C_B) \in \{1, \ldots, L\}^2$ ausgedrückt werden. Welcher Spieler gerade am Zug ist, erkennt man aus der Anzahl der gemachten Züge beider Spieler.

**Beispiel 2.2.** *Bei ESR$(5, 6, 1)$ hat jeder Spieler $L = \binom{11}{6} = 462$ Positionen. Insgesamt ergeben sich daraus höchstens $462^2 = 213\,444$ Konfigurationen. Tatsächlich sind davon genau 21 434 zulässig[4], d. h. etwa 10 %.*

[1] Hinspiel: MC($k$) beginnt, Rückspiel: MC($2k$) beginnt. Siegquote stets aus Sicht von MC($k$).

[2] Exakt heißt hierbei, mit endlichen vielen Operationen praktisch bis auf Maschinengenauigkeit zuverlässige Ergebnisse zu erhalten, vgl. auch Diskussion in Abschnitt 2.6.1.

[3] Vgl. Kombinationen mit Wiederholung: Es gibt genau $\binom{M+m-1}{m}$ Möglichkeiten, aus $M$ Elementen $m$ auszuwählen, wobei es nicht auf die Reihenfolge ankommt und Wiederholungen zulässig sind. Bei uns ist $M = n + 1$

[4] D. h. der Startspieler hat genauso viele Züge bzw. einen Zug mehr als der nachziehende Spieler gemacht. Die Zahl der Figuren im Ziel wurde nicht berücksichtigt.

### 2.4.2 Zufallspartien

Ausgehend von den Endstellungen, d. h. einer der Spieler hat genau $s$ und der andere weniger als $s$ Figuren im Ziel, kann man durch Rückwärtsanalyse den Erwartungswert für den Sieg des Spielers $A$ bei zufälligem Spiel bestimmen. Dabei geht man absteigend in der Zahl der insgesamt von $A$ und $B$ gemachten Züge vor.

Bei geeigneter Verwaltung der Konfigurationen liegt der Rechenaufwand dabei maximal in der Größenordnung $\mathcal{O}(mL^2)$, da es maximal $m$ Kandidaten je Zug gibt. Da aber nur ein Teil der Konfigurationen zulässig sind, ist die tatsächliche Zeitkomplexität unter $\mathcal{O}(L^2)$.

### 2.4.3 Siegwahrscheinlichkeiten der Monte-Carlo-Experimente

Da wir nun für jede Konfiguration den Gewinnerwartungswert $p$ für eine einzelne Zufallspartie kennen, entspricht die Simulation von $N$ unabhängigen Zufallspartien einer Binomialverteilung $b_N(j,p) = p^j \cdot (1-p)^{N-j} \cdot \binom{N}{j}$.

Gegeben seien ein Vektor $(p_1, \ldots, p_m)$ der Siegwahrscheinlichkeiten der Zugkandidaten und eine natürliche Zahl $N$. Es sei $f_N : \{1, \ldots, m\} \to [0,1]$ diejenige Abbildung, die jedem Zugkandidaten die Wahrscheinlichkeit zuordnet, die meisten Siege in den Zufallspartien zu erzielen. Bei Gleichstand soll dabei unter allen besten Zügen gleichwahrscheinlich einer ausgewählt werden. Die Schwierigkeit in dieser Konstruktion besteht darin, dass es $2^m - 1$ Möglichkeiten gibt, welche Teilmenge der Zugkandidaten zu diesen besten Zügen gehört. Gerade bei kleinen Anzahlen $N$ an Zufallspartien sind aber Gleichstände häufig zu erwarten. Deshalb wollen wir hier die Berechnung der Funktion $f_N$ exakt vornehmen, obwohl für große $N$ Näherungen denkbar sind.

Dazu sei $M = \{0,1\}^m \setminus \{0\}$ die Menge aller vom Nullvektor verschiedenen 0-1-Vektoren der Länge $m$, wobei eine 1 an der $i$-ten Stelle bedeutet, dass der Zug $i$ zu den besten Zugkandidaten gehört. Mit $X_p^N$ bezeichnen wir unabhängige $(N,p)$-binomialverteilte Zufallsgrößen. Zunächst berechnen wir die Wahrscheinlichkeit $g_N : M \to [0,1]$ für die Optimalität jeder Teilmenge von Zugkandidaten:

$$\begin{aligned} g_N(x) &= \sum_{j=0}^{N} P\left(X_{p_i}^N = j \text{ für } x_i = 1 \quad \text{und} \quad X_{p_i}^N < j \text{ für } x_i = 0\right) \\ &= \sum_{j=0}^{N} \left(\prod_{x_i=1} b_N(j,p_i)\right)\left(\prod_{x_i=0} B_N(j-1,p_i)\right), \end{aligned}$$

mit

$$B_N(j,p) = \sum_{k=0}^{j} b_N(k,p)\,.$$

Nun müssen diese Wahrscheinlichkeiten auf jeden Zugkandidaten aufgeteilt und über alle relevanten Tie-Break-Situationen summiert werden:

$$f_N(i) = \sum_{x \in M, x_i = 1} \frac{g_N(x)}{\|x\|_1} \,. \tag{2.1}$$

Die Berechnung der Funktion $f$ erfordert damit $\mathcal{O}(N2^m m)$ Operationen, wenn man die Größen $b_N(j,p)$ und $B_N(j,p)$ sukzessive in $j$ hochrechnet. In Abschnitt 2.4.6 stellen wir eine Methode vor, die durch geschickte Summation diese Berechnung in $\mathcal{O}(Nm^3)$ Operationen bewerkstelligt.

Da die Berechnung in Abhängigkeit der Eingangsgrößen $p$ und $1-p$ nur Additionen, Multiplikationen, Divisionen nicht-negativer Operanden enthält, ist der Algorithmus numerisch stabil, d. h. es sind nur kleine Rundungsfehler im Bereich der Maschinengenauigkeit zu erwarten.

### 2.4.4 Partien mit Monte-Carlo-Agenten

Analog zur Berechnung der Zufallspartien gelangen wir durch Rückwärtsanalyse zu den Siegquoten der Monte-Carlo-Agenten. Dabei werden die Zugkandidaten gemäß der Wahrscheinlichkeitsverteilung $f_N(i)$ aus (2.1) ausgewählt, wobei für $N$ der Monte-Carlo-Parameter des jeweiligen Agenten einzusetzen ist. Bei Self-Play-Experimenten mit doppelten Resourcen setzt man $N = k$ bzw. $N = 2k$, je nachdem welcher Agent gerade am Zug ist.

Der Rechenaufwand liegt dabei maximal in der Größenordnung $\mathcal{O}(N2^m m^2 L^2)$, da es maximal $L^2$ Konfigurationen und $m$ Kandidaten je Zug gibt und deren Analyse gemäß Abschnitt 2.4.3 einen Rechenaufwand von $\mathcal{O}(N2^m m)$ erfordert.

### 2.4.5 Ergebnisse für Einzelschritt-Rennen

Wie man im Vergleich von Abbildung 2.3 mit Abbildung 2.2 (Seite 7) erkennen kann, liefert die hier vorgestellte Methode tatsächlich die Wahrscheinlichkeiten, die aus den Simulationsexperimenten näherungsweise bekannt waren. Exemplarisch wurden einzelne Datenpunkte mit bis zu $10^8$ Simulationen überprüft, vgl. Tabelle 2.4. Interessant sind auch die Laufzeiten (auf Intel i7-950 CPU) der Algorithmen. Die exakte Berechnung dauert etwa solange wie 160 Monte-Carlo-Simulationen. Die verbesserte Variante (s. Abschnitt 2.4.6) ist noch schneller.[5]

Die Abbildung 2.5 zeigt dieselbe Situation (ESR$(5,6,1)$), jedoch für noch weit größere Monte-Carlo-Parameter $k$ und in logarithmischer $k$-Skala. Ab etwa $k = 1000$ spielen die Agenten perfekt, d. h. der Anzugsvorteil führt praktisch sicher zum Sieg. Die Graphen sind relativ glatt. Weitere Bassins ergeben sich nicht. In Abbildung 2.6 sind die Ergebnisse für ESR$(5,6,2)$ zu sehen. Es ergibt sich ein ähnliches Bild, jedoch ist das Bassin etwas tiefer geworden.

[5] Bei noch mehr Zugmöglichkeiten wird der Vorteil größer. Für $m = 8$ statt $m = 6$ ist die verbesserte Variante knapp Faktor acht schneller als die direkte Berechnung aus Abschnitt 2.4.3.

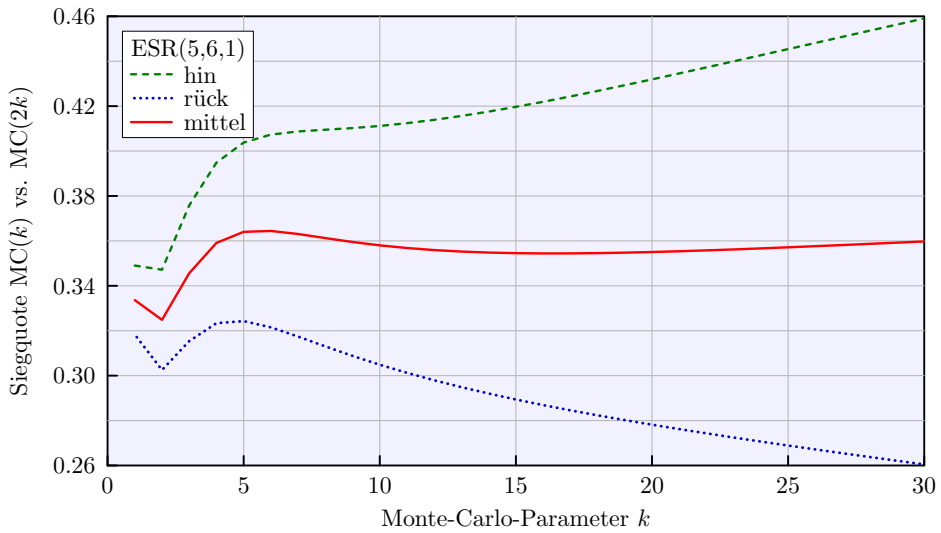



**Abbildung 2.3:** Einzelschritt-Rennen ESR$(5, 6, 1)$ mit MC$(k)$ vs. MC$(2k)$, vgl. Abbildung 2.2.

| $k$ | Simulat. | Hinspiel | Rückspiel | Mittelwert | Rechenzeit |
|---|---|---|---|---|---|
| 3 | $10^8$ | 0.3757581800 | 0.3153726200 | 0.3455654000 | 1510 min |
| | $\infty$ | 0.3758100195 | 0.3153747770 | 0.3455923983 | 0.19 s (0.09 s) |
| 5 | $10^7$ | 0.4038141000 | 0.3243585000 | 0.3640863000 | 272 min |
| | $\infty$ | 0.4037732550 | 0.3242759771 | 0.3640246161 | 0.26 s (0.12 s) |
| 10 | $10^7$ | 0.4109209000 | 0.3047928000 | 0.3578568500 | 453 min |
| | $\infty$ | 0.4111841910 | 0.3048624298 | 0.3580233104 | 0.42 s (0.19 s) |
| 15 | $10^7$ | 0.4197141000 | 0.2896762000 | 0.3546951500 | 609 min |
| | $\infty$ | 0.4196865181 | 0.2893856412 | 0.3545360797 | 0.56 s (0.26 s) |
| 20 | $10^7$ | 0.4321263000 | 0.2782291000 | 0.3551777000 | 732 min |
| | $\infty$ | 0.4319218854 | 0.2781711913 | 0.3550465383 | 0.72 s (0.33 s) |
| 25 | $10^7$ | 0.4453044000 | 0.2688080000 | 0.3570562000 | 839 min |
| | $\infty$ | 0.4454074238 | 0.2688789094 | 0.3571431666 | 0.90 s (0.40 s) |
| 30 | $10^7$ | 0.4300287000 | 0.2741187000 | 0.3520737000 | 990 min |
| | $\infty$ | 0.4590431929 | 0.2604627193 | 0.3597529561 | 1.06 s (0.46 s) |

**Tabelle 2.4:** Einzelschritt-Rennen ESR$(5, 6, 1)$ mit zwei Monte-Carlo-Agenten MC$(k)$ vs. MC$(2k)$: Vergleich von Simulation und exakter Berechnung (Zeilen „$\infty$“). Die Rechenzeit in Klammern steht für die schnelle Variante des Algorithmus' (s. Abschnitt 2.4.6).

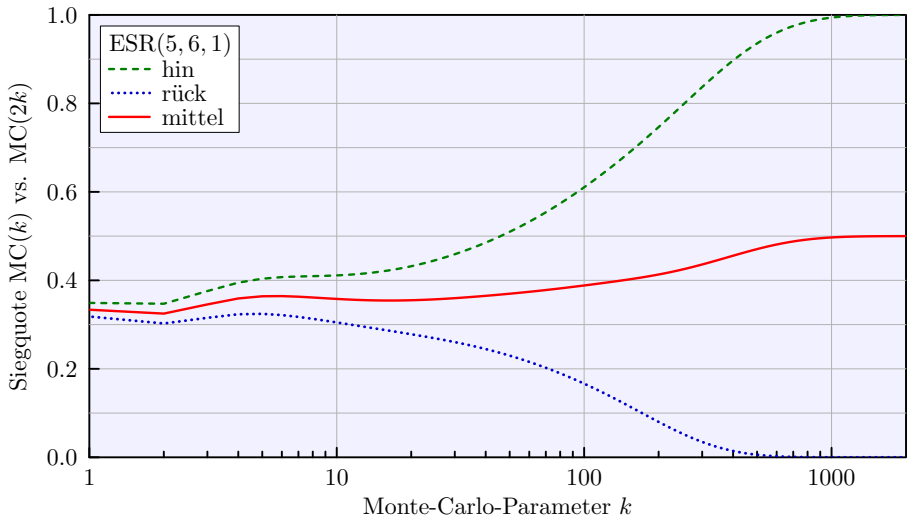



**Abbildung 2.5:** Einzelschritt-Rennen ESR$(5, 6, 1)$ mit MC$(k)$ vs. MC$(2k)$.

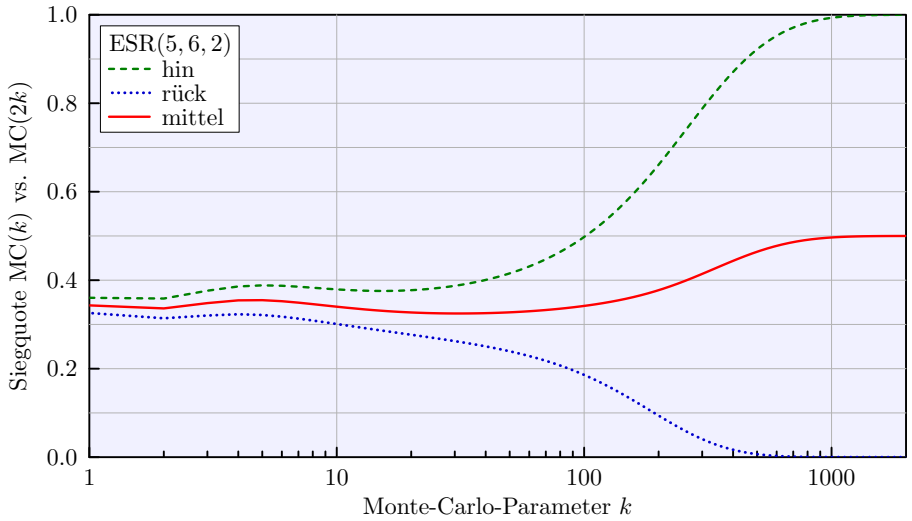



**Abbildung 2.6:** Einzelschritt-Rennen ESR$(5, 6, 2)$ mit MC$(k)$ vs. MC$(2k)$.

Anschließend wurden, ausgehend von ESR$(5, 6, 2)$, die drei Parameter einzeln variiert, vgl. Abbildung 2.7. Besonders tiefe Bassins erhält man für große Bahnlängen $n$ bzw. große Bahnanzahlen $m$, während bzgl. der Zielfiguren $s$ mittlere Werte die ausgeprägtesten Bassins liefern.

Mit zunehmenden $m$ verschieben sich beide Extrema nach rechts, d. h. die Zahl der Instanzen in den Monotoniebereichen lässt sich steigern. Lediglich der Knick beim zweiten Datenpunkt ($k = 2$) bleibt erhalten, wobei es sich aber nicht zwingend um eine Änderung des Monotonieverhaltens handeln muss (vgl. Abbildung 2.7 Mitte).

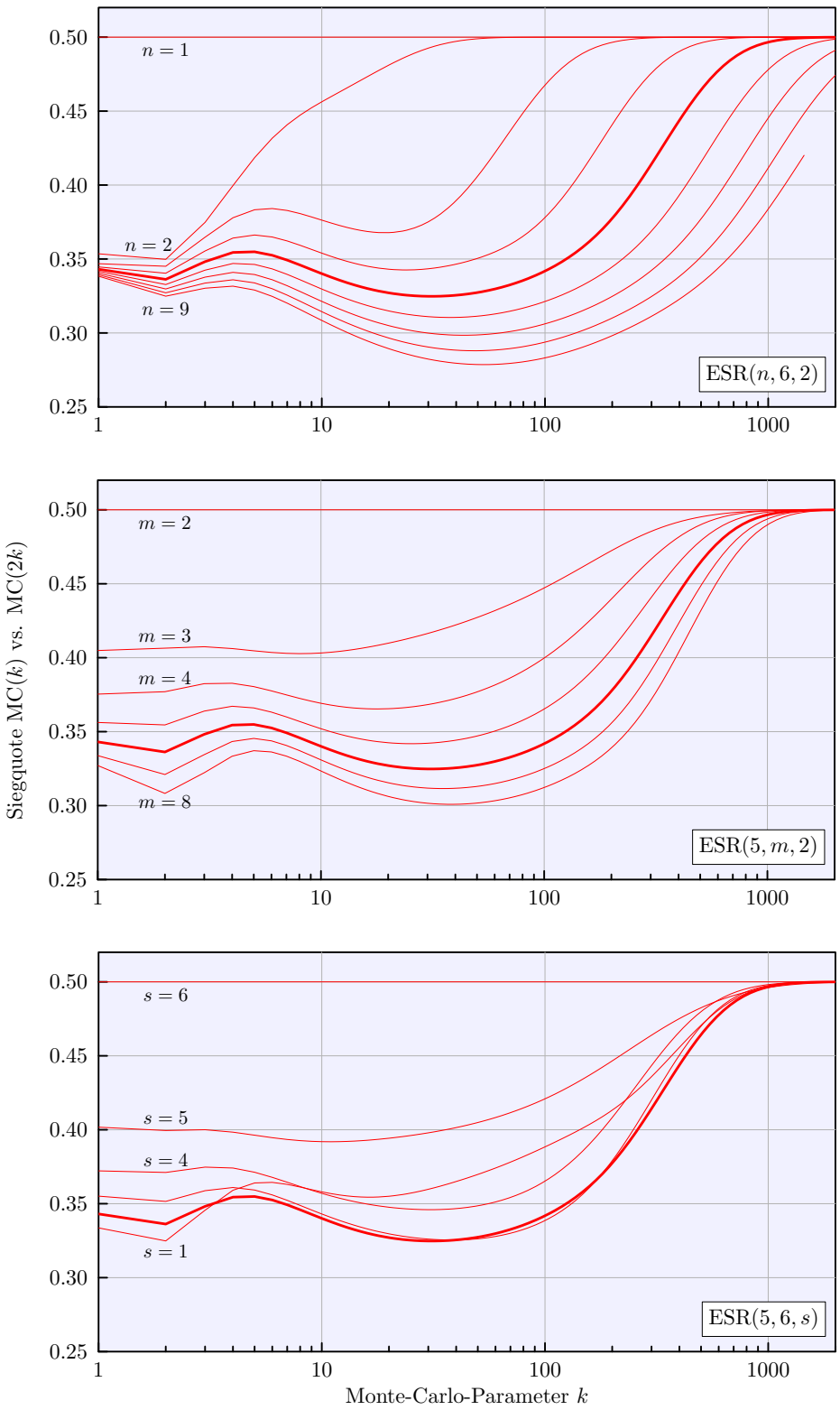



**Abbildung 2.7:** Einzelschritt-Rennen ESR$(n, m, s)$ mit MC$(k)$ vs. MC$(2k)$. Als Referenz ist jeweils ESR$(5, 6, 2)$ hervorgehoben.

### 2.4.6 Schnellere Berechnung der Monte-Carlo-Erwartungswerte

Bevor wir weitere Spiele betrachten, die auch komplexeres Monotonieverhalten aufweisen können, werden wir einen schnelleren Algorithmus zur Berechnung der Monte-Carlo-Erwartungswerte (2.1) konstruieren.[6] Statt $\mathcal{O}(N2^m m)$ Operationen (Monte-Carlo-Parameter $N$, Anzahl der Zugmöglichkeiten $m$) wird nur polymonielle Laufzeit $\mathcal{O}(Nm^3)$ benötigt.

Die Grundidee besteht darin, die Zugkandidaten mit ihren jeweiligen Simulationen nacheinander zu betrachten. Haben wir für $K$ Zugkandidaten ($K < m$) schon alle relevanten Tie-Break-Daten berechnet, fügen wir die Simulation (bestehend aus $N$ Läufen) des $(K+1)$-ten Zugkandidaten hinzu. Für einen festen Wert $j$ der maximalen Sieganzahl gilt:

- Ist die neue Simulation schlechter (Wahrscheinlichkeit $p_- = B_N(j-1, p_{K+1})$) als $j$, so sind die bisherigen Wahrscheinlichkeiten lediglich mit $p_-$ zu skalieren.
- Hat die neue Simulation genau $j$ Siege (Wahrscheinlichkeit $p_0 = b_N(j, p_{K+1})$), so sind die Siegwahrscheinlichkeiten der bisherigen Zugkandidaten mit $p_0$ zu skalieren und neu aufzuteilen. Dafür brauchen wir aber die Informationen, wie viele Zugkandidaten an den bisherigen Tie-Breaks beteiligt waren. Genauer: $h_{j,i} = (h_{j,i}(k))_{k=1}^K$ bezeichne die Wahrscheinlichkeiten, dass Zugkandidat $i$ den Maximalwert $j$ gemeinsam mit $k$ Zugkandidaten erreicht. Diese Wahrscheinlichkeiten $h_{j,i}(k)$ seien gemäß Tie-Break-Regel schon durch $k$ geteilt. Es gilt also $\sum\limits_{i,j,k} h_{j,i}(k) = 1$.
- Ist die neue Simulation besser (Wahrscheinlickkeit $p_+ = 1 - p_0 - p_-$), sind die Siegwahrscheinlichkeiten der bisherigen Zugkandidaten nicht weiter relevant. Den (zunächst alleinigen) Sieganteil des Zuges $K+1$, der auf die Maximalwerte $j' > j$ entfällt, kann man direkt berechnen

$$h_{j',K+1}(1) = b_N(j', p_{K+1}) \prod_{i=1}^{K} B_N(j'-1, p_i)\,.$$

Auf diese Weise arbeitet man sich schrittweise in $j$ und $K$ vor, indem man die einzelnen Summanden der Produkte

$$(h_{j,i}(1) + \cdots + h_{j,i}(K)) \cdot (p_- + p_0 + p_+)\,, \quad i = 1, \ldots, K$$

richtig zuordnet. Für jeden Maximalwert $j = 0, \ldots, N$ und jeden Zugkandidaten $i = 1, \ldots, m$ müssen alle vorherigen Züge $k = 1, \ldots, i-1$ mit ihren maximal $i-1$ Tie-Break-Anzahlen durchgegangen werden. Die Zeitkomplexität beträgt damit $\mathcal{O}(Nm^3)$.

[6] Weil bei den Einzelschritt-Rennen auch die Zahl der Zustände $L = \binom{m+n}{m} = \binom{m+n}{n} > \frac{m^n}{n!}$ schnell mit der Anzahl $m$ der Bahnen wächst, ist die Zeitkomplexität $\mathcal{O}(N \cdot m \cdot 2^m)$ für die Berechnung der Wahrscheinlichkeiten (Abschnitt 2.4.3) noch vertretbar. Jedoch ist diese Berechnung unabhängig vom konkreten Spiel, wenn man $m$ als die Anzahl der Zugmöglichkeiten interpretiert. Daher erscheint eine schnellere Implementierung für andere Zwecke (Spiele mit vielen Zügen bei einer noch handhabbaren Zahl von Konfigurationen) durchaus sinnvoll.

Ein ausführlich kommentierter Programm-Code zur Berechnung der Monte-Carlo-Erwartungswerte befindet sich im Anhang in Abschnitt B.2.

## 2.5 Analyse weiterer Spiele

### 2.5.1 Doppelschritt-Crash-Rennen

Als zweites Modell betrachten wir im Verlauf dieses Abschnittes ein „Renn-Spiel“, in welchem Doppelschritte und gegenseitiges Schlagen erlaubt sind.

**Definition 2.3.** *Das Doppelschritt-Crash-Rennen, kurz DSC($n$), ist ein Zwei-Personenspiel mit folgenden Regeln (vgl. Abbildung 2.8):*

- *Es gibt eine Bahn der Länge $n$. Jeder Spieler hat eine Figur, die anfangs auf dem Startfeld (Position 0 für Spieler W bzw. Position $n$ für Spieler S) steht.*
- *Die Spieler ziehen abwechselnd die eigene Figur wahlweise ein oder zwei Felder vor (Richtung Ziel), passen ist nicht zulässig. Schlagen oder Überspringen des Gegners sind erlaubt.*
- *Wer zuerst seine Figur ins Ziel setzt oder die gegnerische Figur schlägt, hat gewonnen.*

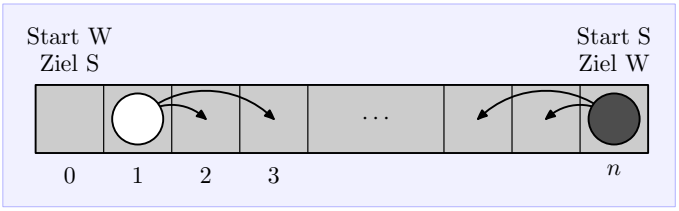



**Abbildung 2.8:** Doppelschritt-Crash-Rennen DSC($n$) nach dem ersten Zug von Spieler W.

Es ist auch hier der Fall, dass es einfache Gewinnstrategien gibt. Es ist klar, dass man den Gegner statt zu überspringen (Doppelschritt) auch schlagen kann (Einzelschritt). Folglich ist nur der Abstand zwischen den Figuren relevant, welcher sich in jedem Zug um 1 oder 2 Felder verkürzt. Damit handelt es sich – spieltheoretisch – um ein Subtraktionsspiel.

Wie bei Einzelschritt-Rennen untersuchen wir reine Monte-Carlo-Agenten. Es zeigt sich, dass diese – im Gegensatz zu Einzelschritt-Rennen – asymptotisch nicht immer einen optimalen Zug finden. Dies ist grundsätzlich in der Natur der Subtraktionsspiele begründet, wie in Kapitel 3 eingehend theoretisch untersucht wird. Die Abbildung 2.9 zeigt für das Doppelschritt-Crash-Rennen mit Bahnlänge 10, d. h. DSC(10), die komplette Analyse für Zufallspartien beginnend aus der Startstellung. Dabei kommt heraus, dass der Startspieler mit einem Einzelschritt im Eröffnungszug 53.1 % der

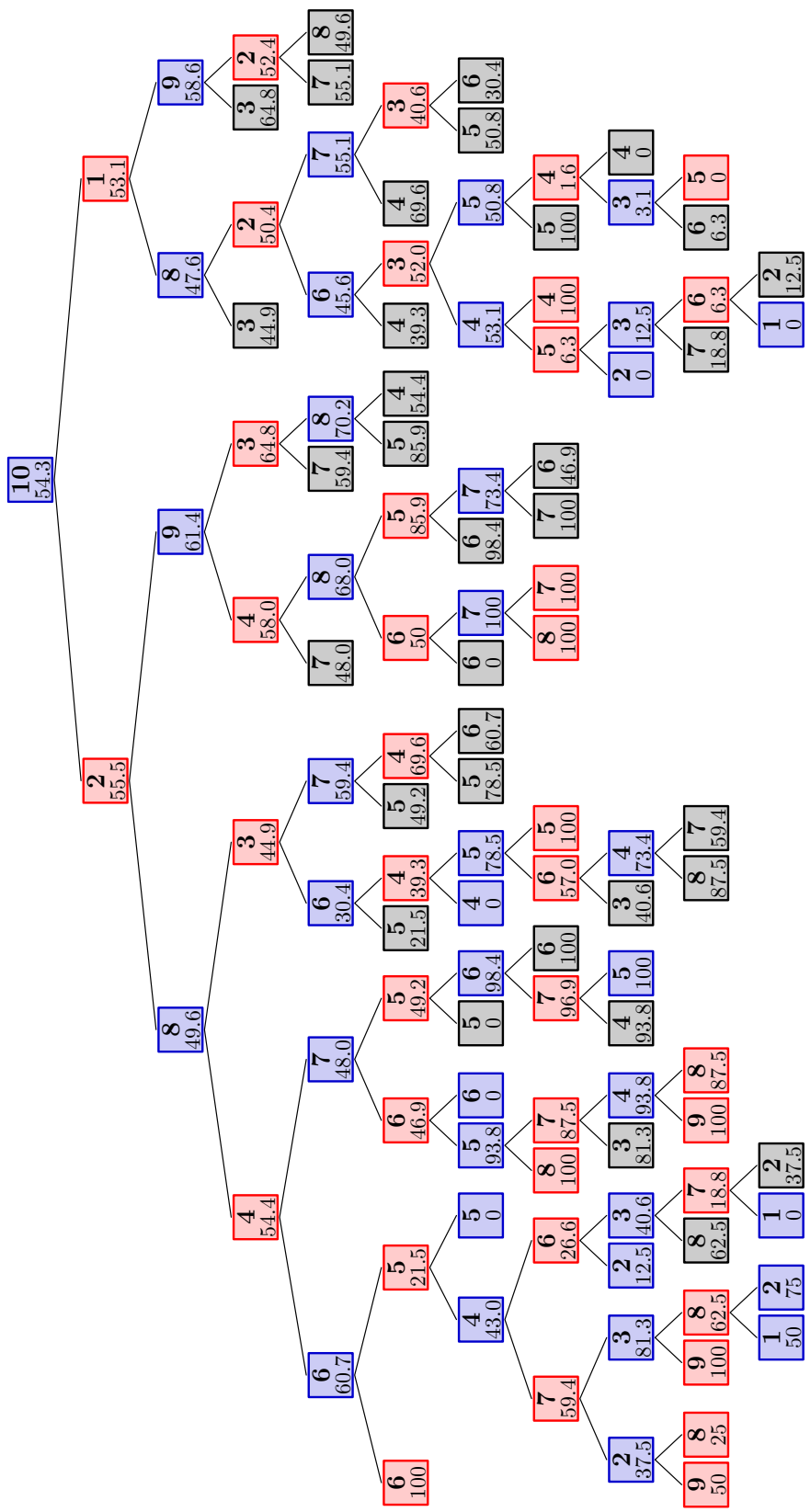



**Abbildung 2.9:** Doppelschritt-Crash-Rennen: Vollständige Analyse von Zufallspartien bei DSC(10) aus der Startposition (Rot auf Feld 0, Blau auf Feld 10). In jedem Kästchen ist der Zug des jeweiligen Spielers und die Siegquote aus Sicht vom Startspieler Rot angegeben (Die Werte sind exakt berechnet, aber gerundet eingetragen). Die Blattwerte ergeben sich aus elementaren Überlegungen. Bei den schwarzen Kästchen sind Teile des Baums abgeschnitten, die schon weiter links im Baum auftauchen. Der spieltheoretische Gewinnzug 1 schneidet mit 53.1 % schlechter ab als Verlustzug 2 mit 55.5 %.

Zufallspartien (ab dem zweiten Zug) und mit einem Doppelschritt 55.5 % der Zufallspartien gewinnt. D. h. bei einer hinreichend großen Simulationsanzahl würde sich der Monte-Carlo-Agent praktisch immer für den Doppelschritt entscheiden.

Aus spieltheoretischer Sicht ist jedoch nur ein Einzelschritt ein Gewinnzug. Die Gewinnstrategie besteht nämlich darin, so zu ziehen, dass der Abstand beider Figuren nach dem Zug ein Vielfaches von drei beträgt. Bei Bahnlänge 10 muss man also einen Einzelschritt machen, um den Abstand 9 zu erreichen. DSC(10) ist die kleinste Instanz, bei der der Monte-Carlo-Agent MC($\infty$) „irrt“.

Im weiteren Verlauf dieses Abschnittes werden wir nun experimentelle Ergebnisse für das Doppelschritt-Crash-Rennen zusammenstellen. Insbesondere konnten bei dieser Self-Play-Analyse tatsächlich mehrerer Bassins gefunden werden.

Die folgenden Abbildungen behandeln jeweils die Siegquoten eines MC($k$)-Agenten gegen einen MC($2k$)-Agenten. Die Siegquoten sind stets aus Sicht des MC($k$)-Agenten angegeben, jedoch gemittelt über beide Möglichkeiten, wer Startspieler ist.

**Zu Abbildung 2.10:** Für $n = 3$ hat der nachziehende Spieler stets einen direkten Gewinnzug, den der MC-Agent auch sicher erkennt. Daher ist die Siegquote (wie auch für $n = 1$ und $n = 2$) konstant bei 50 %.

Von $n = 3$ bis $n = 9$ nähern sich die Kurve nach einem Bassin (fehlt bei $n = 5$) von unten der 50 %-Marke an. Im Gegensatz zu den Einzelschritt-Rennen, vgl. Abbildung 2.7 oben, sind die Graphen nicht systematisch bzgl. der Bahnlänge $n$ angeordnet. Danach treten wegen des asymptotischen Irrtums ab $n = 10$ Siegquoten über 50 % auf, d. h. der vermeintlich schwächere Agent spielt stärker. Dieses Phänomen ist erst ab $k = 147$ zu beobachten.

**Zu Abbildung 2.11:** Hier sind alle Graphen bis zur Bahnlänge $n = 40$ eingezeichnet. Die Abbildung soll einen Gesamtüberblick geben, ohne die einzelnen Graphen im Detail aufzulösen (vgl. Abbildung 2.10 und Abbildung 2.13). Für $n > 25$ und $k > 500$ gibt es weitere Bassins. Zunächst scheint es so, dass sich alle Graphen bis $k = 5000$ von unten der 50 %-Marke nähern oder über diese Marke steigen und (ggf. später) sich von oben annähern. Für $k > 100$ und $n < 40$ liegen alle Graphen oberhalb der Kurve für $n = 7$. Überraschenderweise liegt die Kurve für $n = 40$ auch für große $k$-Werte deutlich unterhalb der 50 %-Marke und hat sogar mehr als zwei Bassins.

**Zu Abbildung 2.12:** Der Fall $n = 40$ ist hier bis zu sehr großen $k$-Werten aufgelöst und man erkennt, dass genau drei Bassins (und ein halbes bei $k \leq 3$) auftreten, bevor die Siegquoten für Hinspiel (Startspieler MC($k$)) und Rückspiel (Startspieler MC($2k$)) gegen 0 bzw. 1 konvergieren. Die Hin- und Rückspiel-Graphen sind für $k < 200$ sehr dicht beieinander und trennen sich dann zügig auf. Jedoch gibt es bei $k \approx 3000$ nochmals eine deutliche Einschnürung.

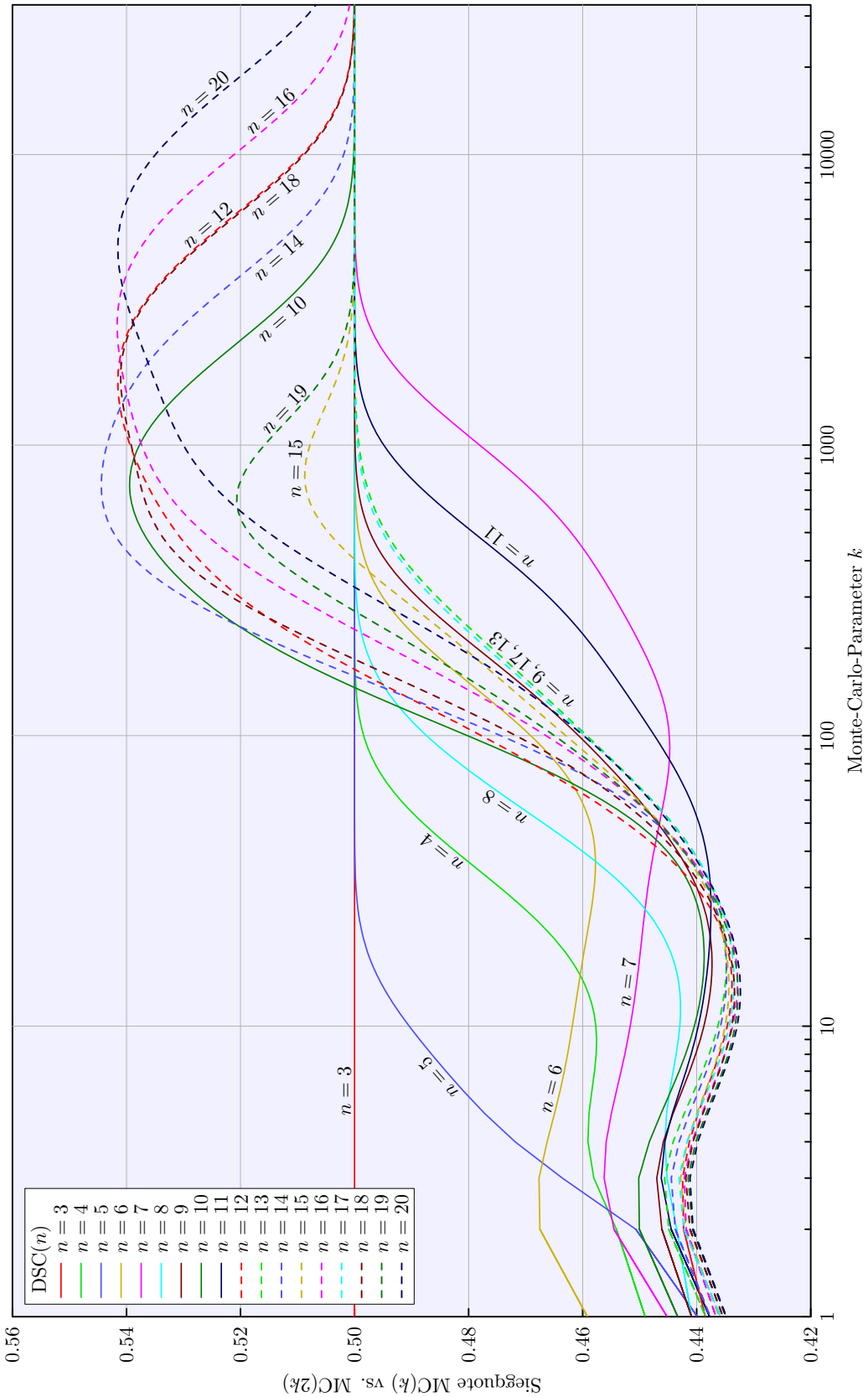



**Abbildung 2.10:** Doppelschritt-Crash-Rennen DSC($n$): MC($k$) vs. MC($2k$) für Bahnlängen $n = 3, \ldots, 20$, Siegquote aus Sicht von MC($k$).

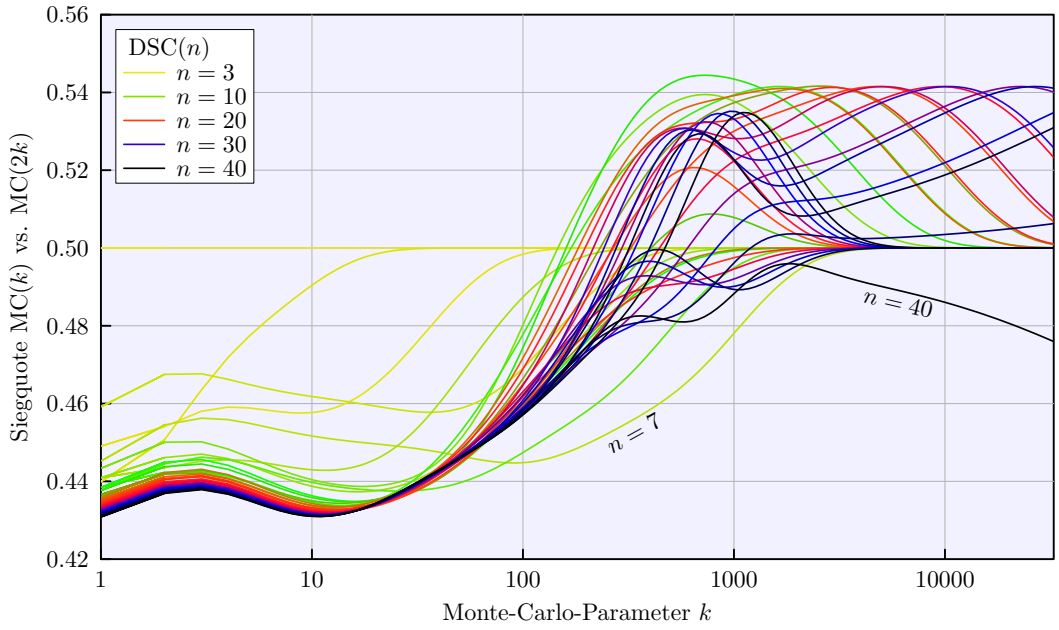



**Abbildung 2.11:** Doppelschritt-Crash-Rennen DSC($n$): MC($k$) vs. MC($2k$) für Bahnlängen $n = 3, \ldots, 40$. Die Farbe wurde schrittweise mit der Bahnlänge von gelb ($n = 3$) bis schwarz ($n = 40$) variiert. Für $k \leq 100$ erkennt man die punktweise Konvergenz der Graphen.

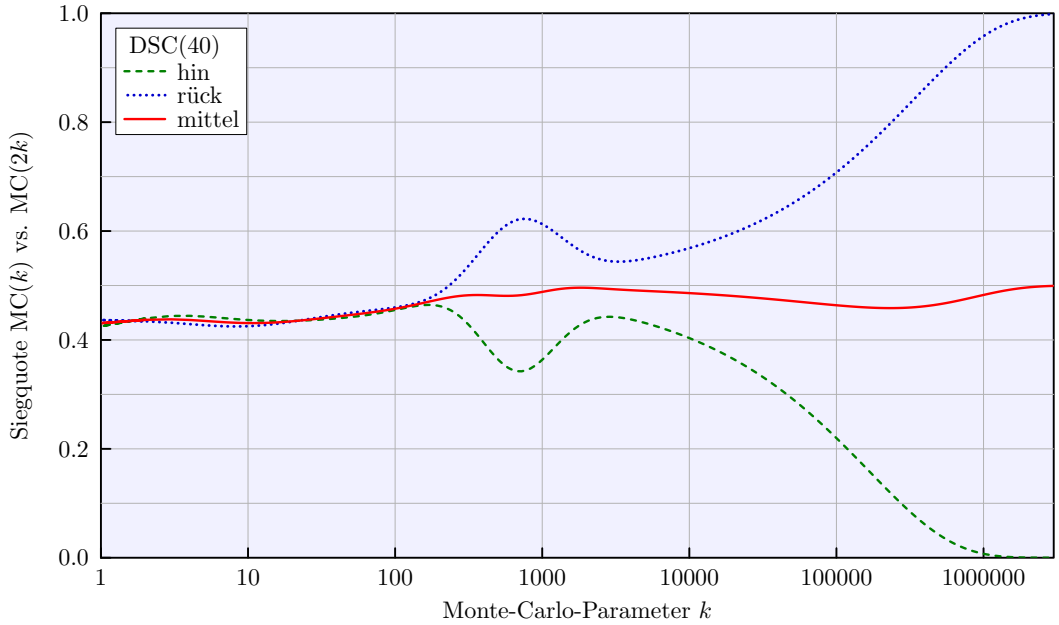



**Abbildung 2.12:** Doppelschritt-Crash-Rennen DSC(40): MC($k$) vs. MC($2k$) für Bahnlänge $n = 40$. Es sind genau drei Bassins zu erkennen (bei $k \approx 10$, $k \approx 600$ und $k \approx 200\,000$).

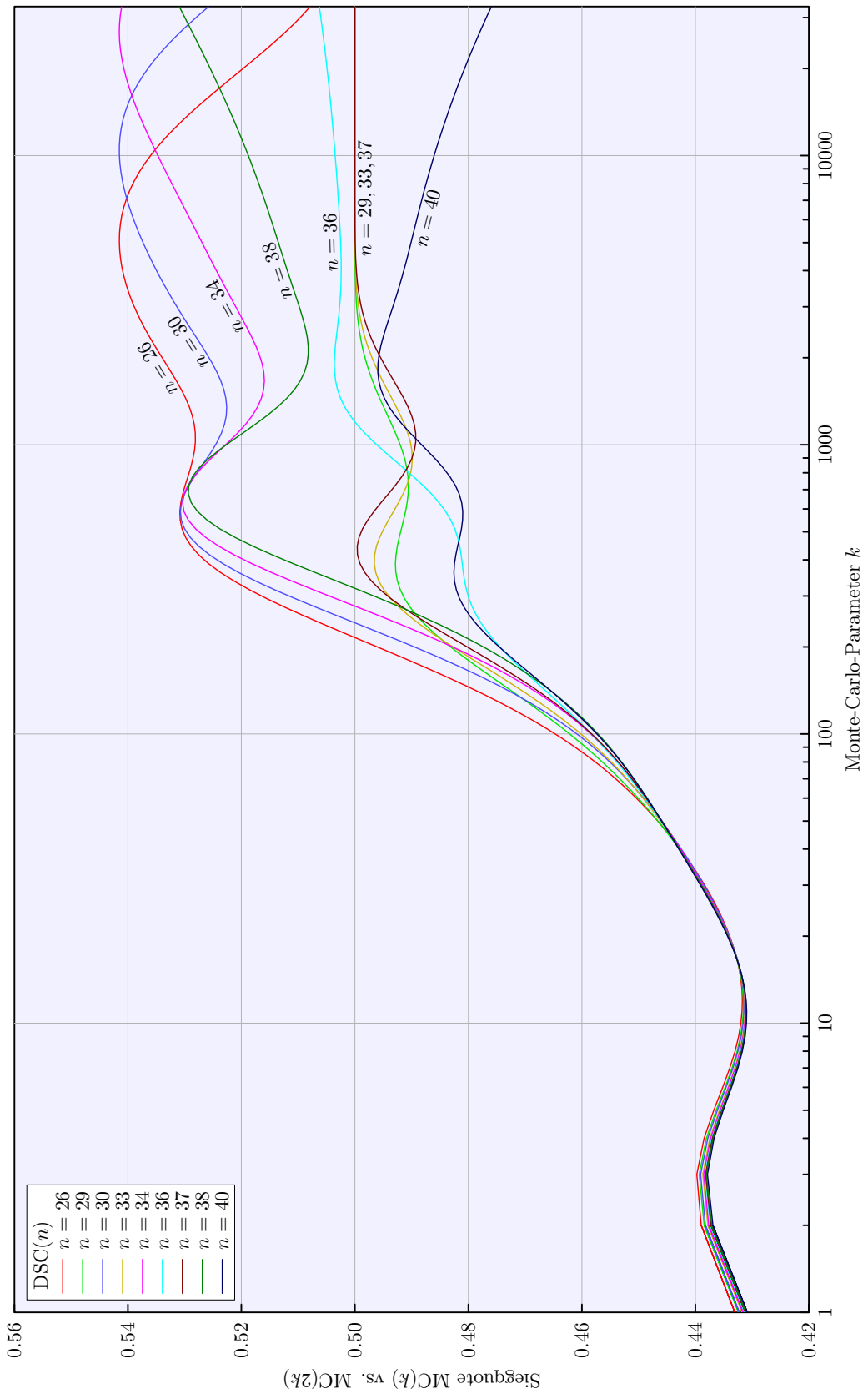



**Abbildung 2.13:** Doppelschritt-Crash-Rennen DSC($n$): MC($k$) vs. MC($2k$) für alle Bahnlängen $n < 40$ bei denen sich Doppelbassins ergeben. Bei $n = 40$ deutet sich sogar das erste Dreifach-Bassin an.

**Zu Abbildung 2.13:** Für $n < 40$ sind alle Graphen mit Doppelbassins dargestellt, sowie das Trippel-Bassin für $n = 40$. Es ist zu vermuten, dass für größere Bahnlängen und Monte-Carlo-Parameter $k$ noch komplexere Graphenverläufe auftreten können.

Bei diesen Abbildungen erkennt man, dass die Graphen sich einer Grenzkurve anzunähern scheinen. Zumindest für $k < 100$ ist das in Abbildung 2.11 gut zu sehen. Dies deckt sich mit Feststellung, dass bis zu einem bestimmten $k$-Wert die Hin- und Rückspiel-Graphen dicht beieinander liegen (Abbildung 2.12). Mit steigenden $n$ verschiebt sich vermutlich die Stelle an der sich Hin- und Rückspiel trennen langsam zu immer größeren $k$-Werten.

### 2.5.2 Zwei-Zug-Subtraktionsspiele

Als dritte und letzte Klasse von Spielen in diesem Kapitel betrachten wir Zwei-Zug-Subtraktionsspiele.

**Definition 2.4.** *Das Zwei-Zug-Subtraktionsspiel mit Zugmenge $(a, b)$, $1 \leq a < b$, und Starthaufengröße $n$, kurz $S_{(a,b)}(n)$, hat folgende Regeln:*

- *Zwei Spieler entfernen abwechselnd, ausgehend von einem Haufen mit $n$ Elementen, wahlweise $a$ oder $b$ viele Elemente.*
- *Derjenige Spieler, der das letzte Element entfernt, gewinnt.*
- *Der Spieler am Zug gewinnt auch, wenn die Haufengröße für den gewählten Zug nicht mehr ausreicht.*

Wir schreiben die Zugmenge als Zahlenpaar $(a, b)$ mit runden Klammern und können ohne Einschränkung der Allgemeinheit $1 \leq a < b$ fordern. Die letzte Regel bedeutet damit, dass alle Haufengrößen kleiner oder gleich $b$ Gewinnstellungen für den Spieler am Zug sind.

Diese Definition und ein Beispiel befinden sich nochmals am Beginn von Kapitel 3. Dort werden Monte-Carlo-Agenten für das Subtraktionsspiel $\mathrm{S}_{(1,2)}$ mit Zugmenge $(a, b) = (1, 2)$ theoretisch untersucht.

Wie schon bei den Einzelschritt- und Doppelschritt-Crash-Rennen lässt sich eine perfekte Strategie leicht ermitteln. Ausgehend von den Gewinnstellungen mit Haufengröße $n \leq b$, kann man rekursiv die nachfolgenden Stellungen bewerten. Für die Subtraktionsspiele $\mathrm{S}_{(1,2)}$, $\mathrm{S}_{(1,3)}$, $\mathrm{S}_{(1,4)}$, $\mathrm{S}_{(2,3)}$, $\mathrm{S}_{(3,4)}$ und $\mathrm{S}_{(2,5)}$ ist dies in Tabelle 2.14 dargestellt.

Da nur auf endlich viele Vorgängerzustände zurückgegriffen wird, wird das Muster stets periodisch. Die ersten Periode ist in der Tabelle unterstrichen. Diese Strukturen wurden in [25] eingehend untersucht (insbesondere auch für Subtraktionsspiele mit mehr als zwei Zugmöglichkeiten). Es soll nur erwähnt werden, dass die Vorperiode und die Periode schon für vier bis fünf Zugmöglichkeiten sehr lang werden können.

| Zugmenge | | Haufengröße | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $(a,b)$ | $n^*$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| $(1,2)$ | 14 | $+$ | $b$ | $-$ | $a$ | $b$ | $-$ | $a$ | $b$ | $-$ | $a$ | $b$ | $-$ | $a$ | $b$ | $-$ | $a$ | $b$ | $-$ |
| $(1,3)$ | $\infty$ | $+$ | $b$ | $b$ | $-$ | $a$ | $-$ | $+$ | $-$ | $+$ | $-$ | $+$ | $-$ | $+$ | $-$ | $+$ | $-$ | $+$ | $-$ |
| $(1,4)$ | 59 | $+$ | $b$ | $b$ | $b$ | $-$ | $a$ | $-$ | $a$ | $b$ | $-$ | $a$ | $-$ | $a$ | $b$ | $-$ | $a$ | $-$ | $a$ |
| $(2,3)$ | 148 | $+$ | $+$ | $b$ | $-$ | $-$ | $a$ | $+$ | $b$ | $-$ | $-$ | $a$ | $+$ | $b$ | $-$ | $-$ | $a$ | $+$ | $b$ |
| $(2,5)$ | 333 | $+$ | $+$ | $b$ | $b$ | $b$ | $-$ | $-$ | $a$ | $a$ | $-$ | $b$ | $+$ | $-$ | $-$ | $+$ | $a$ | $-$ | $b$ |
| $(3,4)$ | 816 | $+$ | $+$ | $+$ | $b$ | $-$ | $-$ | $-$ | $a$ | $+$ | $+$ | $b$ | $-$ | $-$ | $-$ | $a$ | $+$ | $+$ | $b$ |

**Tabelle 2.14:** Zwei-Zug-Subtraktionsspiele $\mathrm{S}_{(a,b)}$: Ein „$-$“ markiert Haufengrößen, die für den Spieler am Zug Verluststellungen bedeuten. Alle anderen Stellungen sind Gewinnstellungen, wobei mit $a$ bzw. $b$ der Gewinnzug angegeben ist. Bei „$+$“ sind beide Zugmöglichkeiten Gewinnzüge. Die Periode ist unterstrichen.
Die kleinste Haufengröße bei der ein MC($\infty$)-Agent in einer Gewinnstellung einen Verlustzug vorschlägt, ist mit $n^*$ bezeichnet.

Wir wollen hier wiederum Monte-Carlo-Agenten an einfachen Subtraktionsspielen untersuchen. Dazu verwenden wir die Zwei-Zug-Spiele aus Tabelle 2.14 und Haufengrößen $3 \leq n \leq 50$. Wie beim Doppelschritt-Crash-Rennen (welches ja fast ein Subtraktionsspiel ist) kommt es vor, dass Monte-Carlo-Agenten einen nicht optimalen Zug vorschlagen. Die kleinste Haufengrößen, bei der das passiert, ist in Tabelle 2.14 mit $n^*$ bezeichnet.[7]

Bei $\mathrm{S}_{(1,2)}$ tritt dies erstmals bei $n^* = 14$ auf.[8] In den anderen Beispielen treten diese kritischen Stellen erst später auf. In [40] findet man einige der hier angegebenen und noch weitere Beispiele.[9] Eine Ausnahme bildet das Spiel $\mathrm{S}_{(1,3)}$, bei welchem es nur für $n \leq 5$ strategische Optionen gibt. Ab $k \approx 20$ erkennen dies die Agenten auch zuverlässig (vgl. Abbildung 2.15), so dass keine falschen Züge vorgeschlagen werden.

Weiterhin sind in Abbildung 2.15 Self-Play-Ergebnisse für die anderen schon genannten Zwei-Zug-Subtraktionsspiele $\mathrm{S}_{(1,2)}$, $\mathrm{S}_{(1,4)}$, $\mathrm{S}_{(2,3)}$, $\mathrm{S}_{(3,4)}$ und $\mathrm{S}_{(2,5)}$ dargestellt. Wir fassen die Beobachtungen (auch in Bezug zu den bisherigen Ergebnissen) zusammen:

- Die Graphen haben einen ähnlichen Verlauf wie beim Doppelschritt-Crash-Rennen DSC($n$) (vgl. Abbildung 2.11). Die meisten Graphen starten bei ca. 40–45 %, durchlaufen ein Bassin und konvergieren dann gegen die 50 %-Marke.

[7] Zur Bestimmung von $n^*$ siehe Kapitel 3.

[8] Beim Doppelschritt-Crash-Rennen trat dies schon bei $n = 10$ auf, wobei der Agent statt einem Einzelschritt ($a$) einen Doppelschritt ($b$) vorschlägt. Durch die scheinbare Möglichkeit ohne Schlagen das Ziel zu erreichen wurde der Doppelschritt gewissermaßen provoziert (Konstruktionsidee von DSC).

[9] Statt wie in [40] die Berechnung exakt per Computer-Algebra-System durchzuführen, kann man auch die numerisch stabilere Formel (3.1) (mit $\varepsilon = 0$) aus Kapitel 3 dieser Arbeit verwenden.

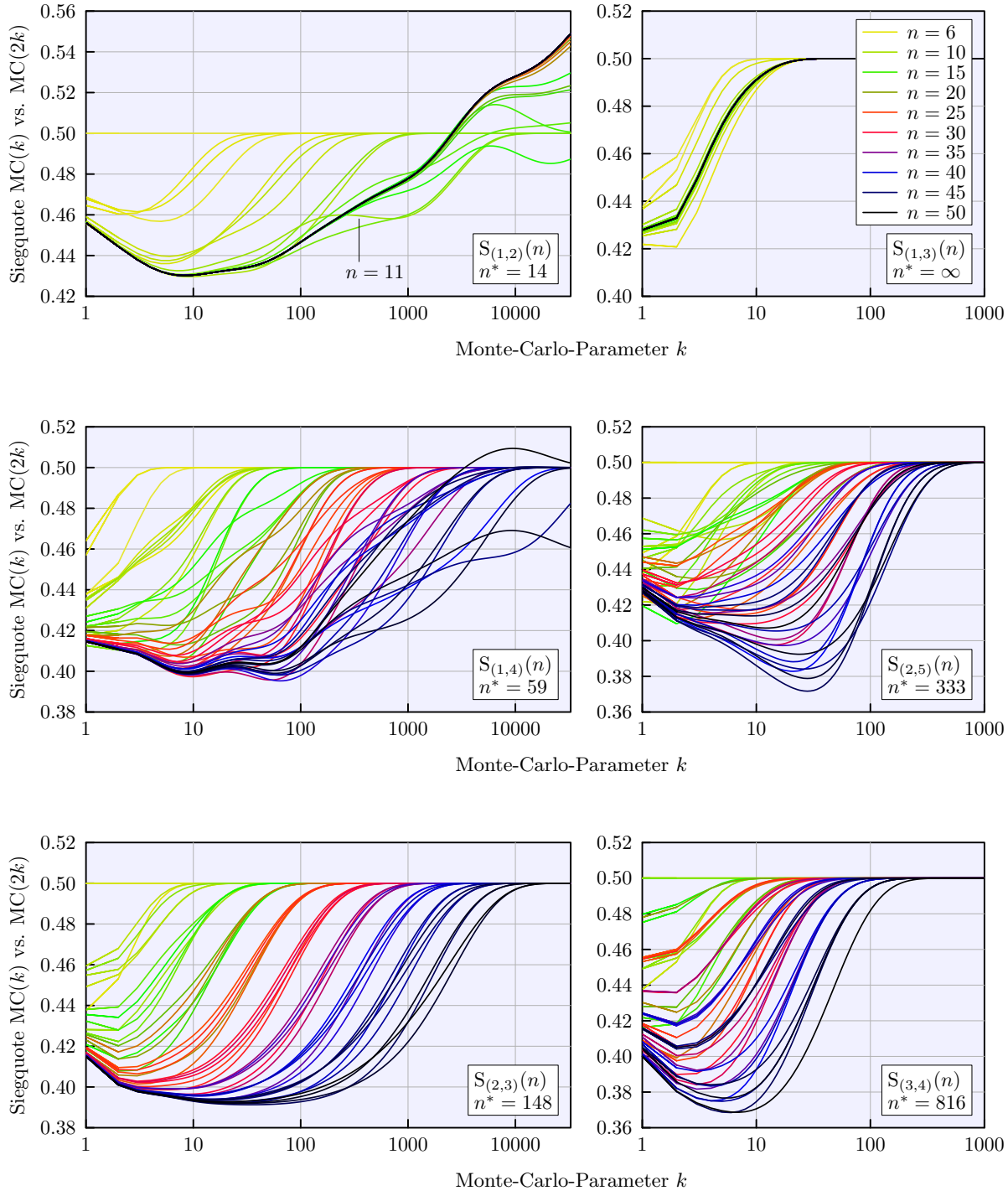



**Abbildung 2.15:** Zwei-Zug-Subtraktionsspiele $S_{(a,b)}(n)$: MC($k$) vs. MC($2k$) für Haufengrößen bis $n = 50$. Mit $n^*$ ist jeweils die kleinste Haufengröße angegeben, bei der Monte-Carlo-Agent MC($\infty$) einen ungünstigen Zug vorschlägt.

- Das (erste) Bassin wird mit zunehmender Starthaufengröße $n$ tiefer und verschiebt sich zu größeren $k$-Werten.
- Bei $S_{(1,2)}$ gibt es das erste Doppelbassin bei $n = 11$, also erheblich früher als beim Doppelschritt-Crash-Rennen (mit $n = 26$).
- Werte über 50 % traten nur bei $S_{(1,2)}$ und $S_{(1,4)}$ auf und nur für $k > 2500$. Bei $S_{(1,4)}$ wird aus dem Bild noch nicht klar, wie sich die Graphen für $n > 50$ verhalten, ein typisches Muster ist für $n > 40$ und $k > 100$ nicht erkennbar. Bei $S_{(2,3)}$, $S_{(2,5)}$ und $S_{(3,4)}$ ist ebenso unklar, ob die 50 %-Marke für größere $n$ noch überschritten wird. Dies ist aber durchaus denkbar, da die kritischen Haufengrößen $n^*$ mit 148, 333 bzw. 816 recht groß sind.
- Das Konvergenzverhalten der Graphen für $n \to \infty$ ist sehr unterschiedlich ausgeprägt. Bei $S_{(3,4)}$ und $S_{(2,5)}$ ist außer der tendenziellen Verschiebung der Graphen in Richtung des Bassins noch keine Konvergenz sichtbar. Bei $S_{(1,4)}$ und $S_{(2,3)}$ ist eine Konvergenz zumindest zu erahnen.

  Bei $S_{(1,2)}$ beobachten wir eine schnelle Konvergenz. Die einzelnen Graphen lösen sich nach und nach bei immer größeren $k$-Werten aus der „Hauptfaser“. Bei $S_{(1,3)}$ überrascht die schnelle Konvergenz nicht weiter, da die Strategie nicht von Haufengrößen $n > 5$ abhängt.
- Es zeigt sich die Tendenz, dass die punktweise Konvergenz der Graphen für $n \to \infty$ umso langsamer ist, je größer die kritische Haufengröße $n^*$ ist (außer $S_{(1,3)}$). Dass es immer eine Konvergenz geben muss, ist so zu erklären: Für festes $k$ spielen die Agenten bei großen Bahnlängen praktisch zufällig. Erst bei kleineren Haufengrößen machen sich die unterschiedlichen „Spielstärken“ (von MC($k$) und MC($2k$)) bemerkbar.
- Betrachten wir umgekehrt für feste Bahnlängen $n$ die Graphen für $k \to \infty$, so findet ebenfalls eine Konvergenz statt. Im Grenzfall spielen schließlich beide Agenten mit der wohldefinierten MC($\infty$) Strategie. Damit ergibt sich wegen Mittelung bzgl. des Startspielers eine Siegquote von 50 %.[10]

  Hier zeigt sich die umgekehrte Tendenz, nämlich dass die Konvergenz gegen die 50 %-Marke für $k \to \infty$ langsamer ist, wenn die kritische Haufengröße $n^*$ klein ist.

[10] Die Hin- und Rückspiel-Siegewahrscheinlichkeiten konvergieren zwar gegen 0 bzw. 1, aber gegen welchen dieser beiden Werte, hängt von $n$ ab. Dies ist (für das Spiel $S_{(1,2)}$) u. a. Gegenstand der theoretischen Analysen in Kapitel 3.

## 2.6 Diskussion

### 2.6.1 Einzelschritt-Rennen

Die vorgestellte Methode erlaubt es, die Siegwahrscheinlichkeiten reiner Monte-Carlo-Agenten exakt vorherzusagen, vorausgesetzt der Konfigurationsraum des Spiels ist nicht zu groß. Abweichungen ergeben sich aufgrund der stabilen Berechnungsmethode nur durch Rundungsfehler im Bereich der Maschinengenauigkeit.[11]

Selbst wenn man keinen Wert auf so genaue Ergebnisse legt, ist die Methode bei einfachen Spielen erheblich schneller als Simulationsexperimente, da diese für statistisch signifikante Aussagen ca. $10^4$ bis $10^6$ Einzelexperimente je Datenpunkt erfordern.

Beschränkungen ergeben sich jedoch durch die Größe $C$ des Konfigurationsraumes, da dieser komplett analysiert werden muss. Die Zeitkomplexität beträgt $\mathcal{O}(Nm^4C)$, dabei sind $m$ die maximale Zahl von Zugkandidaten und $N$ die Zahl der Zufallspartien. Der Speicherbedarf wird ebenfalls durch den Konfigurationsraum bestimmt, er beträgt $\mathcal{O}(C)$.[12]

Angewendet wurde die Methode bei der Analyse von Self-Play-Bassin-Strukturen. Durch die Effizienz des Programmes kann innerhalb weniger Minuten der gesamte Monte-Carlo-Parameterbereich ($1 \leq k \leq 1000$) exakt untersucht werden. Experimentell konnte so gezeigt werden, dass sich bei den Einzelschritt-Rennen, abgesehen von einem Knick bei MC(2) vs. MC(4), maximal drei Monotoniebereiche einstellen. Echte Doppelbassins konnten also nicht gefunden werden.

Im Bereich von $1 \leq k \leq 100$ ist der Nachteil von MC($k$) gegenüber MC($2k$), abgesehen von den Bassin-Strukturen, relativ konstant zwischen 30 und 40 % und lässt dann für $k \geq 100$ nach. Ab etwa $k = 1000$ (bei mehr und längeren Bahnen später) spielen MC-Agenten praktisch perfekt.

### 2.6.2 Doppelschritt-Crash-Rennen und Zwei-Zug-Subtraktionsspiele

Vor allem aufgrund des überschaubaren Konfigurationsraumes (Positionen der beiden Figuren bei DSC($n$) bzw. Haufengröße bei $\mathrm{S}_{(a,b)}(n)$) ließ sich ein größerer Parameterbereich von $n$ und $k$ analysieren. Die Rechenzeit für die sechs Diagramme in Abbildung 2.15 betrug jeweils ca. 10 Minuten. Sie wächst linear mit dem Monte-Carlo-Parameter $k$ und ebenso linear mit der Größe $C$ des Konfigurationsraumes, vgl. Abschnitt 2.4.4. Bei den Subtraktionsspielen ist $C = 2(n + 1)$.

Es konnten mehrere Self-Play-Bassins gefunden werden. Bei DSC(40) gibt es sogar ein Dreifach-Bassin, und bei $\mathrm{S}_{(1,2)}$ und $\mathrm{S}_{(1,4)}$ deutet sich dies an.

[11] Durch Verwendung von Intervallarithmetik [30], könnte man sich die Größtfehler sogar automatisch abschätzen lassen.

[12] Lässt sich der Konfigurationsraum nach der Anzahl der Züge (oder einer anderen geeigneten monotonen Invariante) separieren, kann man den Speicherbedarf entsprechend reduzieren.

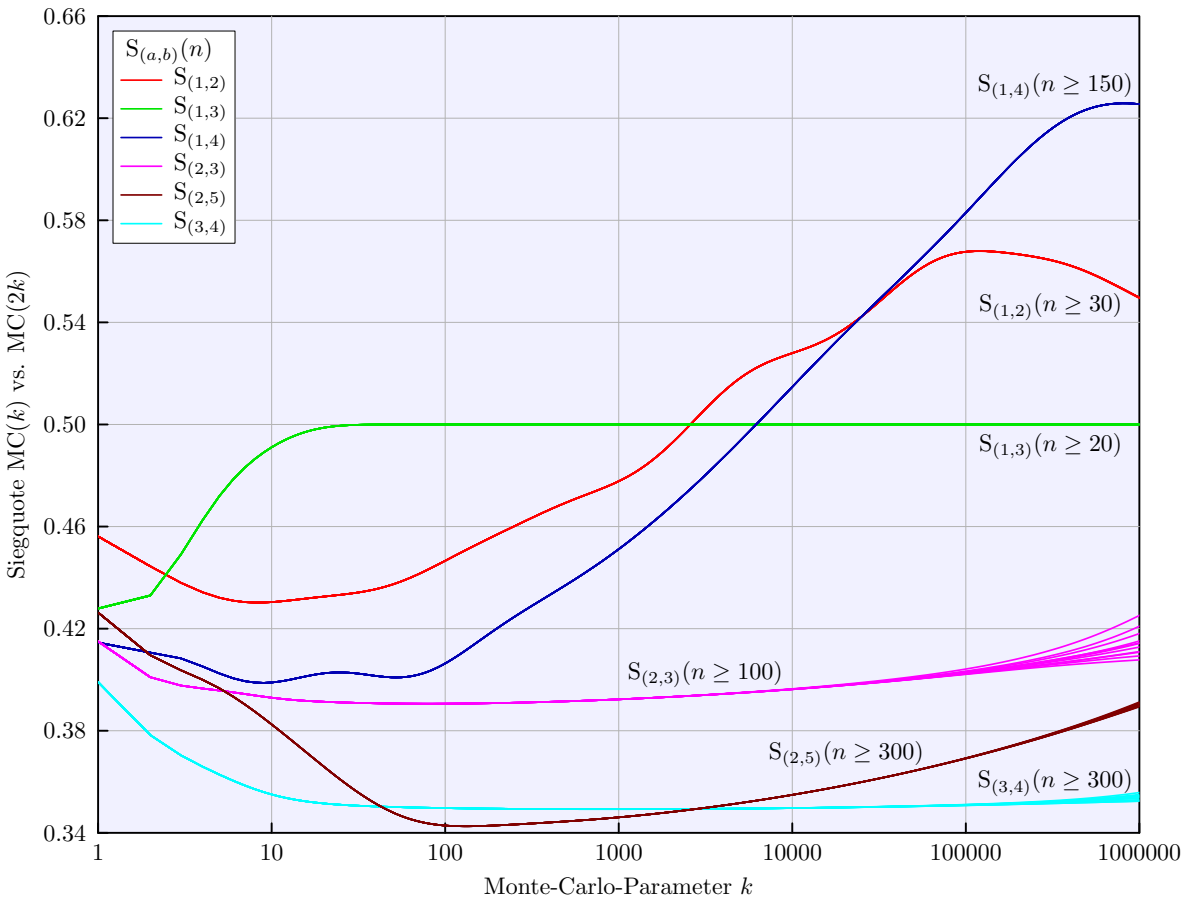



**Abbildung 2.16:** Zwei-Zug-Subtraktionsspiele $S_{(a,b)}(n)$: Grenzgraphen für MC($k$) vs. MC($2k$). Ab der angegebene Grenze sind jeweils noch 10 weitere Graphen eingezeichnet, beispielsweise bei $S_{(2,3)}$ für $n = 100, \ldots, 110$.

Für große Bahnlängen bzw. Starthaufengrößen zeichnet sich die Konvergenz der Self-Play-Graphen für MC($k$) vs. MC($2k$) gegen eine Grenzkurve ab. Um dies genauer zu untersuchen wurden die Subtraktionsspiele bis $k = 10^6$ und große Starthaufengrößen analysiert. Tatsächlich stellen sich Grenzgraphen ein, wie man in Abbildung 2.16 erkennt. Einige der Graphen fächern sich am Ende noch etwas auf – hier kann man den Grenzprozess gut nachvollziehen.

Generell ist festzuhalten, dass sich bei allen untersuchten Spielen (mit gelegentlichen Ausnahme bei kleinen Instanzen wie ESR(2,6,2) oder $S_{(1,4)}(n \lesssim 20)$) ein deutliches Self-Play-Bassin ausbildet. Bei den Subtraktionsspielen ist dieses Bassin umso tiefer und zu größeren $k$-Werten verschoben, je länger der MC($k$)-Agent keine ungünstigen Züge macht.

Im Bereich der Bassins ist der Spielstärkezuwachs zwischen MC($k$) und MC($2k$) besonders groß. Es hat also nicht viel Sinn, beispielsweise in Verfahren wie UCB/UCT-Spielbaumsuche [10], Monte-Carlo-Agenten mit sehr großen Simulationsanzahlen zu verwenden.

## 2.7 Ausblick und offene Fragen

Mit den entwickelten Werkzeugen ist es möglich, eine ganze Reihe von Spielen exakt zu analysieren. Hier bieten sich Subtraktionsspiele mit $m$ Zugmöglichkeiten für weitere Untersuchungen an. Für kleine Werte von $m$, beispielsweise $m \in \{2, 3, 4, 5\}$, ist die Zeitkomplexität für die Berechnung der Monte-Carlo-Siegwahrscheinlichkeiten mit $\mathcal{O}(Nm^4n)$ ($N$=Zahl der Zufallspartien, $n$=Starthaufengröße) u. U. sogar kleiner als die Simulation einer einzelnen Partie.[13] Von diesen Simulationen bräuchte man zudem noch eine größere Anzahl, um statistisch signifikante Aussagen treffen zu können.

Auch bei den hier behandelten Spielen gibt es noch einige offene Fragen:

- Gibt es bei weiteren Instanzen der Einzelschritt-Rennen (oder anderen Spielen mit perfekt spielenden MC($\infty$)-Agenten) kompliziertere Abhängigkeiten als in Abbildung 2.7 dargestellt (Mehrfachbassins, Überschreitung der 50 %-Marke)?
- Wie setzt sich das Bild bei DSC($n$) bzw. $\mathrm{S}_{(1,2)}(n)$ für $k > 10^6$ fort?
- Wie setzen sich die Grenzgraphen für $k > 10^6$ fort?
- Kann man die Beobachtungen zur Konvergenzgeschwindigkeit ($n \to \infty$ bzw. $k \to \infty$) in Abhängigkeit vom kritischen Parameter $n^*$ (kleinste Instanz bei der MC($\infty$) nicht-optimale Züge vorschlägt) verallgemeinern?
- Welche weiteren Aussagen lassen sich aus Grenzgraphen über die Spiele bzw. die Monte-Carlo-Agenten ableiten?

Um zu Agenten MC($k$) mit noch größeren $k$-Werten vorzudringen, kann man den vorgestellten Algorithmus jedoch nicht direkt verwenden. Die kritische Stelle ist dabei nicht der eigentliche Algorithmus, sondern die Berechnung der Binomialverteilung. Für $N \gg 10^6$ sollte man $b_N(j, p)$ bzw. $B_N(j, p)$ über den Logarithmus der Gammafunktion bzw. die unvollständige Betafunktion ausrechnen [1].

Für große $k$-Werte sind Tie-Breaks sehr selten. Kann man durch Ignorieren von Tie-Breaks gute Näherungen für (2.1) konstruieren, die sich schnell berechnen lassen?

[13] Die Simulation einer Partie hat die Zeitkomplexität $\mathcal{O}(Nm\tilde{n}^2)$, wobei $\tilde{n}$ die durchschnittliche Spieldauer einer (Zufalls-)Partie ist. Damit kommt es auf die Parameter $m$, $n$ und $\tilde{n}$ an, welches Verfahren schneller ist.

# Kapitel 3

# Exakte Analyse asymmetrischer Monte-Carlo-Agenten für das (1, 2)-Subtraktionsspiel

*In diesem Kapitel werden Monte-Carlo-Agenten untersucht, die bei den Zufallspartien systematisch bestimmte Züge bevorzugen. Diese Asymmetrie der Zugwahrscheinlichkeiten kann statisch oder auch in Abhängigkeit des Spielwertes modelliert werden.*

*Für das (1,2)-Subtraktionsspiel, kurz $S_{(1,2)}$, werden beide Varianten analytisch untersucht. Während statische Asymmetrie ungünstige Züge des Monte-Carlo-Agenten nicht verhindern kann, führt ein hinreichender Anteil an Zusatzwissen zu perfektem Spiel.*

## 3.1 Das klassische Subtraktionsspiel $S_{(1,2)}$

**Definition 3.1.** *Das Subtraktionsspiel $S_{(1,2)}(m)$ hat die folgenden Regeln: Gegeben ist ein Haufen mit $m$ Streichhölzern. Zwei Spieler entfernen abwechselnd wahlweise ein oder zwei Hölzchen. Derjenige Spieler, der das letzte Hölzchen nimmt, hat gewonnen.*

**Beispiel 3.2.** *Durch Rückwärtsanalyse lassen sich die Gewinn- (G) und Verluststellungen (V) von $S_{(1,2)}$ eindeutig ermitteln.*

| *Haufengröße* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Spielwert* | *G* | *G* | *V* | *G* | *G* | *V* | *G* | *G* | *V* | *G* | *G* | *V* |
| *optimaler Zug* | *1, 2* | *2* | – | *1* | *2* | – | *1* | *2* | – | *1* | *2* | – |

*Es ergibt sich das periodische Muster G G V. Zur besseren Übersichtlichkeit ist in den Verluststellungen kein „bester“ Zug angegeben (–).*

Im weiteren Verlauf des Kapitels wollen wir für dieses Spiel Monte-Carlo-Agenten (reine und asymmetrische Varianten) analytisch untersuchen. Die Methoden lassen sich z. T. auf Subtraktionsspiele mit anderen Zugmengen (auch mit mehr als zwei Zugmöglichkeiten) verallgemeinern.[1] Eine rein analytische Auswertung ist dann aber erheblich aufwändiger bzw. nicht geschlossen[2] möglich. Eine umfassende Untersuchung anderer Zugmengen, ggf. unter Zuhilfenahme numerischer Methoden, ist durchaus denkbar, würde aber den Rahmen dieses Teils der Arbeit sprengen.

## 3.2 Reiner Monte-Carlo-Agent

Wir untersuchen in diesem Kapitel stets den „asymptotischen“ Monte-Carlo-Agenten MC($\infty$). Dieser wählt immer denjenigen Zug, in dessen weiteren Spielverlauf eine Zufallspartie die höchste Siegwahrscheinlichkeit hat, vgl. Abschnitt 1.1. Bei gleichwertigen besten Zugkandidaten wird gleichverteilt einer dieser Züge ausgewählt.

Ein solcher Agent spielt trotz der (idealisiert) unendlichen Zahl von Simulationen nicht zwangsweise perfekt. Im Gegenteil, es gilt sogar folgendes Resultat:

**Satz 3.3.** *Für das Subtraktionsspiel $S_{(1,2)}$ existieren unendlich viele Haufengrößen $m$, die Gewinnstellungen entsprechen, d. h. $m \not\equiv 0 \pmod 3$, bei denen ein MC($\infty$)-Agent in eine Verluststellung ziehen würde.*

Den Beweis verschieben wir, da dieser Satz als Grenzfall in der nun folgenden Analyse asymmetrischer Monte-Carlo-Agenten ohnehin enthalten ist.

## 3.3 Statisch-asymmetrische Monte-Carlo-Agenten

In jeder Spielsituation gibt es bei $S_{(1,2)}$ genau zwei Zugmöglichkeiten. Während beim üblichen Monte-Carlo-Agenten beide Züge in den Zufallspartien gleichwahrscheinlich gewählt werden, ordnen wir nun beiden Varianten beliebige, aber feste Wahrscheinlichkeiten $p(1) = \frac{1-\varepsilon}{2}$ und $p(2) = \frac{1+\varepsilon}{2}$ mit $\varepsilon \in [-1, 1]$ zu. D. h. während der Zufallspartien werden mit Wahrscheinlichkeit $p(1)$ ein bzw. mit $p(2)$ zwei Hölzchen entnommen.

Statische Asymmetrie ist damit im weitesten Sinne ein Modell für (komplexere) Spiele, in denen eine A-priori-Gleichverteilung unter den Zugkandidaten nicht immer (einfach) realisierbar ist. Man denke z. B. an Wirtschaftssimulationsspiele bei denen die Züge typischerweise aus mehreren, z. T. abhängigen, Phasen (etwa Kaufen, Tauschen, Bauen) bestehen. Bei der Monte-Carlo-Simulation solcher Spiele greift man schon aus Performancegründen auf einfache Auswahlregeln (z. B. Gleichverteilung in jeder Phase) zurück, auch wenn dabei keine Gleichverteilung unter allen Zugoptionen realisiert wird.

[1] Wichtig ist dabei die Zusatzregel, dass ein Spieler auch dann gewinnt, wenn er mit dem gewählten Zug mehr als die noch vorhandene Hölzchenzahl nehmen würde.

[2] Da man die Nullstellen der charakteristischen Polynome der Rekursionsschemata nicht mehr explizit angeben kann.

### 3.3.1 Rekursionsgleichungen und charakteristisches Polynom

Mit $x(m)$ bezeichnen wir die Siegwahrscheinlichkeit der Zufallspartien mit Starthaufengröße $m$ aus Sicht des Startspielers. Diese Siegwahrscheinlichkeiten lassen sich rekursiv berechnen:

$$\begin{aligned} x(m) &= p(1)\cdot[1-x(m-1)] + p(2)\cdot[1-x(m-2)] \\ &= 1 - \frac{1-\varepsilon}{2}x(m-1) - \frac{1+\varepsilon}{2}x(m-2) \\ &\text{mit den Startwerten } x(0)=0 \text{ und } x(1)=1\,. \end{aligned}$$

Sowohl für analytische Betrachtungen als auch für die stabile numerische Auswertung ist es angebracht, die Inhomogenität durch die lineare Transformation

$$x(m) = \frac{1}{2} + y(m)$$

zu eliminieren. Dies führt auf

$$\begin{aligned} y(m) &= -\frac{1-\varepsilon}{2}y(m-1) - \frac{1+\varepsilon}{2}y(m-2) \\ &\text{mit den Startwerten } y(0) = -\frac{1}{2} \text{ und } y(1) = \frac{1}{2}\,. \end{aligned} \tag{3.1}$$

Diese lineare Differenzengleichung lässt sich durch den Exponentialansatz $y(m) = \lambda^m$ lösen. Das charakteristische Polynom

$$\lambda^2 + \frac{1-\varepsilon}{2}\lambda + \frac{1+\varepsilon}{2} = 0$$

hat die beiden Nullstellen

$$\begin{aligned} \lambda_1 &= \frac{\varepsilon - 1 + \sqrt{(5-\varepsilon)^2 - 32}}{4} \\ \lambda_2 &= \frac{\varepsilon - 1 - \sqrt{(5-\varepsilon)^2 - 32}}{4}\,. \end{aligned} \tag{3.2}$$

Die allgemeine Lösung ergibt sich, falls $\lambda_1 \neq \lambda_2$, aus einer Linearkombination der speziellen Lösungen $\lambda_i^m$. Unter Beachtung der Startwerte $y(0) = -\frac{1}{2}$ und $y(1) = \frac{1}{2}$ ergibt sich

$$y(m) = \alpha_1\lambda_1^m + \alpha_2\lambda_2^m \quad \text{mit} \quad \alpha_1 = \frac{1+\lambda_2}{2(\lambda_1-\lambda_2)},\ \alpha_2 = -\frac{1+\lambda_1}{2(\lambda_1-\lambda_2)}\,. \tag{3.3}$$

Im Falle einer doppelten Nullstelle des charakteristischen Polynoms, d. h. $\varepsilon = 5-\sqrt{32}$ und $\lambda_1 = \lambda_2 = \lambda = 1-\sqrt{2}$, ist die allgemeine Lösung eine Linearkombination der speziellen Lösungen $\lambda^m$ und $m\lambda^m$. Aus den Startwerten ergibt sich dann

$$y(m) = \beta_1\lambda^m + \beta_2 m\lambda^m \quad \text{mit} \quad \lambda = 1-\sqrt{2},\ \beta_1 = -\frac{1}{2},\ \beta_2 = -\frac{\sqrt{2}}{2}\,. \tag{3.4}$$

### 3.3.2 Reelle Fundamentallösungen

Für $\varepsilon \leq 5 - \sqrt{32} \approx -0.656854$ sind die $\lambda$-Werte aus (3.2) reell. In diesem Fall gilt folgendes Lemma:

**Lemma 3.4.** *Für jedes $-1 \leq \varepsilon \leq 5 - \sqrt{32}$ ist $y(m)$ aus Gleichung (3.3) eine alternierende Folge.*

**Beweis**. Zunächst gilt $-1 \leq \lambda_2 \leq 1 - \sqrt{2} \leq \lambda_1 \leq 0$. Dies lässt sich durch elementare Umformungen unter Beachtung von $-1 \leq \varepsilon \leq 5 - \sqrt{32}$ leicht zeigen. Daraus folgt speziell $|\lambda_1| \leq |\lambda_2|$.

Wegen $\alpha_1 + \alpha_2 = -\frac{1}{2}$ und $0 \leq \alpha_1$ gilt $\alpha_2 \leq -\frac{1}{2}$ und somit $|\alpha_1| < |\alpha_2|$.

Insgesamt gilt damit $|\alpha_1 \lambda_1^m| < |\alpha_2 \lambda_2^m|$. In der Folge $y(m)$ ist also der Summand $\alpha_2 \lambda_2^m$ absolut dominierend. D. h. mit $\alpha_2 \lambda_2^m$ ist auch die gesamte Folge $y(m)$ alternierend.

Im Grenzfall $\varepsilon = 5 - \sqrt{32}$ hat die Folge $y(m)$ die Form (3.4). Hier ist ebenso der alternierende Term $\beta_2 m \lambda^m$ dominierend. □

### 3.3.3 Komplexe Fundamentallösungen

Für $\varepsilon > 5 - \sqrt{32} \approx -0.656854$ sind die Lösungen (3.2) komplex, was aber kein grundsätzliches Problem darstellt. Da die Lösungen konjugiert zueinander sind, geht mit $\lambda = \lambda_1$ und $\overline{\lambda} = \lambda_2$ die allgemeine Lösung (3.3) in die Form

$$y(m) = \frac{1}{2(\lambda - \overline{\lambda})} \left[ (1 + \overline{\lambda})\lambda^m - (1 + \lambda)\overline{\lambda}^m \right]$$

über. Mit der kartesischen bzw. Polarkoordinatendarstellung

$$\lambda = a + ib = re^{i\varphi}$$

ergibt sich

$$y(m) = \frac{1}{4ib} \big[ \underbrace{(1 + a - ib)\lambda^m - (1 + a + ib)\overline{\lambda}^m}_{z} \big].$$

Wegen $\operatorname{Im}(z) = 2(1 + a)\operatorname{Im}(\lambda^m) - 2b\operatorname{Re}(\lambda^m)$ und $\lambda^m = r^m(\cos m\varphi + i \sin m\varphi)$ folgt

$$y(m) = \frac{r^m}{2b} \left[ (1 + a)\sin(m\varphi) - b\cos(m\varphi) \right]. \tag{3.5}$$

In Amplituden- und Phasendarstellung[3] mit nur einer trigonometrischen Funktion erhalten wir schließlich das folgende Ergebnis.

[3] Beachte $A \sin\alpha + B\cos\alpha = \sqrt{A^2 + B^2}\cos(\alpha - \arctan(A, B))$ mit $\arctan(y, x) = \arg(x + iy)$.

**Lemma 3.5.** *Für $5-\sqrt{32} < \varepsilon \leq 1$ gilt für die Siegquote statisch-asymmetrischer Monte-Carlo-Agenten*

$$\boxed{\begin{aligned} x(m) &= \frac{1}{2} + y(m) = \frac{1}{2} - \frac{\sqrt{1+\varepsilon}}{2b}\, r^m \cos(m\varphi + \varphi_0) \\ \textit{mit} \quad r &= \sqrt{\frac{1+\varepsilon}{2}} \\ a &= \frac{\varepsilon - 1}{4} \\ b &= \frac{\sqrt{32-(5-\varepsilon)^2}}{4} \\ \varphi &= \arctan(b,a) = \frac{\pi}{2} + \arctan\frac{-a}{b}, \qquad \textit{beachte } a \leq 0 \\ \varphi_0 &= \arctan\frac{1+a}{b}\,. \end{aligned}} \tag{3.6}$$

**Beispiel 3.6.** *Im Spezialfall $\varepsilon = 0$ des symmetrischen Zufallsspiels gilt*

$$x(m) = \frac{1}{2} + \frac{2^{1-m/2}}{\sqrt{7}} \sin\left[m \cdot \left(\pi - \arctan\sqrt{7}\right) - \arctan\frac{\sqrt{7}}{3}\right]. \tag{3.7}$$

*Diese Funktion ist in Abbildung 3.1 dargestellt.*

### 3.3.4 Ungünstige Züge des Monte-Carlo-Agenten

Im Fall $\varepsilon \leq 5-\sqrt{32}$ ist nach Lemma 3.4 klar, wie sich der Monte-Carlo-Agent verhält[4]:

| Haufengröße $m$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Spielwert | G | G | V | G | G | V | G | G | V | G | G | V |
| optimaler Zug | $1,2$ | 2 | – | 1 | 2 | – | 1 | 2 | – | 1 | 2 | – |
| $\varepsilon \leq 5-\sqrt{32}$: $\operatorname{sgn}(y(m))$ | $+$ | $-$ | $+$ | $-$ | $+$ | $-$ | $+$ | $-$ | $+$ | $-$ | $+$ | $-$ |
| MC($\infty$)-Agent | $1,2$ | 2 | 1 | $\boxed{2}$ | $\boxed{1}$ | 2 | 1 | 2 | 1 | $\boxed{2}$ | $\boxed{1}$ | 2 |

Periodisch schlägt also der MC($\infty$)-Agent für $m$ kongruent 4 bzw. 5 modulo 6 den falschen Zug vor (gekennzeichnet durch Kästchen).

Im Fall $\varepsilon > 5-\sqrt{32}$ ergibt sich nach Formel (3.6) eine gedämpfte Kosinus-Funktion, deren Frequenz $\varphi$ vom Asymmetrie-Parameter $\varepsilon$ abhängt, dabei gilt $\frac{\pi}{2} \leq \varphi < \pi$.

Einen typischen Verlauf zeigt Abbildung 3.1. Die durchgezogene Kurve entspricht der analytischen Darstellung aus Lemma 3.5. Durch das Subtraktionsspiel sind genau genommen nur die Werte $y(m)$ für die diskreten Stellen $m \in \mathbb{N}$ definiert. Die Anzahl

[4] Bei Haufengröße $m$ macht er den Zug $-\operatorname{argmin}_{i\in\{1,2\}} y(m-i)$.

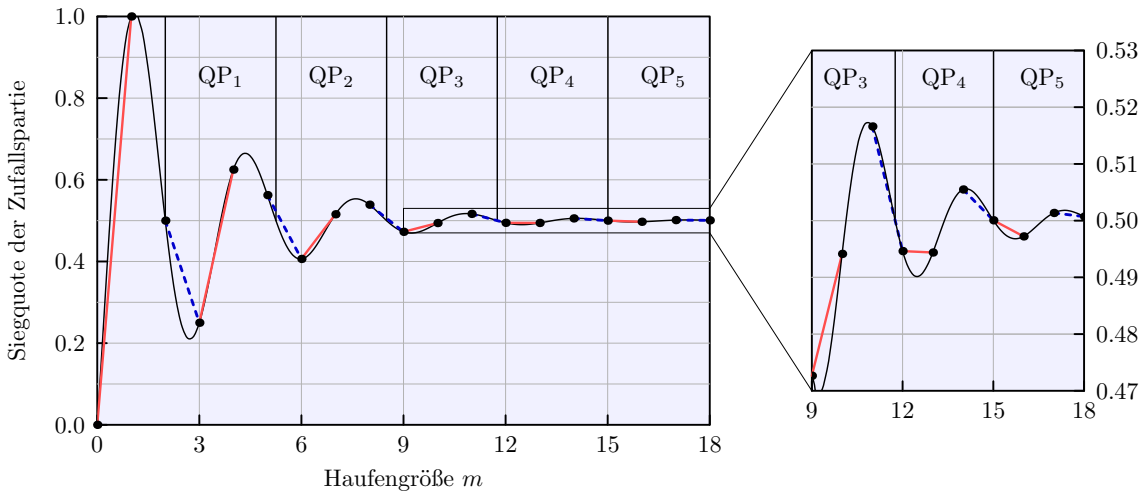



**Abbildung 3.1:** Siegquote $x(m) = \frac{1}{2} + y(m)$ symmetrischer Zufallspartien ($\varepsilon = 0$) gemäß Formel (3.7). Die roten (durchgezogenen) Segmente $(3k)$—$(3k+1)$ verschieben sich relativ zu den Quasi-Perioden (QP) nach links. Bei (12)—(13) wird das Segment erstmalig fallend, d. h. ein MC($\infty$)-Agent entscheidet sich bei Haufengröße 14 für den falschen Zug.
Die Stelle $m = 15$ liegt (ganz knapp) in QP$_4$, während $m = 16$ in QP$_5$ liegt. Damit ist das Segment (15)—(16) direkt (d. h. ohne Berechnung der Funktionswerte) als fallend identifizierbar.

dieser durch dickere Punkte dargestellten Werte pro Quasi-Periode (QP, charakterisiert durch die Nullstellen der kontinuierlichen Funktion $y(m)$) hängt von $\varphi$ ab. Für $\varphi = \pi$ sind es genau zwei und für $\varphi = \frac{\pi}{2}$ sind es genau vier.

Eine kritische Stelle im Intervall $\frac{\pi}{2} < \varphi < \pi$ wird bei $\varphi^* = \frac{2\pi}{3}$ erreicht, denn dann entspricht die Quasi-Periodizität des MC($\infty$)-Agenten genau der Periodenlänge drei der Gewinnn- und Verluststellungen.

**Beispiel 3.7.** *Im kritischen Fall* $\varphi^* = \frac{2\pi}{3}$ *gilt wegen* $\arctan \frac{-a}{b} = \frac{2\pi}{3} - \frac{\pi}{2} = \frac{\pi}{6}$ *für das kritische* $\varepsilon^*$

$$\frac{1-\varepsilon^*}{\sqrt{(5-\varepsilon^*)^2 - 32}} = \tan\frac{\pi}{6} = \frac{1}{\sqrt{3}}\,.$$

*Dies führt auf eine quadratische Gleichung, welche im zulässigen Intervall nur die Lösung* $\varepsilon^* = 2 - \sqrt{5}$ *hat. Damit berechnet man* $r = \frac{\sqrt{5}-1}{2}$, $a = \frac{1-\sqrt{5}}{4}$, $b = \frac{(\sqrt{5}-1)\sqrt{3}}{4}$ *und* $\frac{1+a}{b} = \sqrt{5/3}$. *Eingesetzt in Gleichung* (3.5) *ergibt sich*

$$\begin{aligned}
y(m) &= \frac{r^m}{2}\left(\frac{1+a}{b}\cdot\sin(m\varphi) - \cos(m\varphi)\right)\\
&= \frac{1}{2}\cdot\left(\frac{\sqrt{5}-1}{2}\right)^m\left(\sqrt{5/3}\cdot\sin\left(m\cdot\frac{2\pi}{3}\right) - \cos\left(m\cdot\frac{2\pi}{3}\right)\right)\\
&= \frac{1}{2}\cdot\left(\frac{\sqrt{5}-1}{2}\right)^m\cdot\begin{cases}-1 & \textit{falls} \quad m\equiv 0 \pmod 3\\ \frac{1+\sqrt{5}}{2} & \textit{falls} \quad m\equiv 1 \pmod 3\\ \frac{1-\sqrt{5}}{2} & \textit{falls} \quad m\equiv 2 \pmod 3\,.\end{cases}
\end{aligned}$$

*Insbesondere gilt hier*

$$\begin{aligned} y(3k+1) &= -y(3k)\,, \\ y(3k+2) &= -y(3k+1)\cdot(\sqrt{5}-2)\,, \\ y(3k+3) &= y(3k+2)\,. \end{aligned}$$

*Da $y(3k+3)$ und $y(3k+2)$ denselben Wert haben, entscheidet sich der Monte-Carlo-Agent bei $m$ kongruent 1 modulo 3 zu 50 % für den falschen Zug:*

| *Haufengröße $m$* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Spielwert* | *G* | *G* | *V* | *G* | *G* | *V* | *G* | *G* | *V* | *G* | *G* | *V* |
| *optimaler Zug* | *1, 2* | *2* | – | *1* | *2* | – | *1* | *2* | – | *1* | *2* | – |
| $\varepsilon = 2-\sqrt{5}$: $\operatorname{sgn}(y(m))$ | + | – | – | + | – | – | + | – | – | + | – | – |
| *MC($\infty$)-Agent* | *1, 2* | *2* | *1* | *1,*$\boxed{2}$ | *2* | *1* | *1,*$\boxed{2}$ | *2* | *1* | *1,*$\boxed{2}$ | *2* | *1* |

In den unkritischen Fällen gilt folgender Satz:

**Satz 3.8.** *Für jedes $5-\sqrt{32} < \varepsilon \leq 1$ mit $\varepsilon \neq 2-\sqrt{5}$ gibt es kongruent 1 und kongruent 2 modulo 3 jeweils unendlich viele Haufengrößen $m$, bei denen ein statisch-asymmetrischer MC($\infty$)-Agent einen Verlustzug vorschlägt.*

**Beweis**. Für $5-\sqrt{32} < \varepsilon < 2-\sqrt{5}$ gilt $\frac{2\pi}{3} < \varphi < \pi$. D. h. genau zwei oder drei aufeinanderfolgende ganzzahlige Stellen $m$ fallen in eine Quasi-Periode von $y(m)$. Unter einer Quasi-Periode (QP) ist hierbei das Intervall zwischen je zwei Nullstellen der Funktion $y(m)$ mit negativer Ableitung zu verstehen (jede zweite Nullstelle). Modulo 3 verschiebt sich von einer zur nächsten QP maximal ein diskreter Punkt über die QP-Grenze, wobei diese Verschiebung aufgrund der festen Frequenzdifferenz systematisch passiert. Damit liegt für jedes $t \in \{0,1,2\}$ regelmäßig eine QP-Grenze in den Intervallen $(3k+t, 3k+t+1)$. Da die Quasi-Halbperioden (Bereiche mit gleichem Vorzeichen) wegen $\varphi < \pi$ länger als eins sind, sind Segmente mit einer solchen Nullstelle fallend $(y(3k+t) > 0 > y(3k+t+1))$, vgl. Abbildung 3.1.

Analog betrachtet man Quasi-Perioden bzgl. der Nullstellen mit positiver Ableitung und findet modulo 3 regelmäßig ansteigende Segmente.

Für $2-\sqrt{5} < \varepsilon < 1$ gilt $\frac{\pi}{2} < \varphi < \frac{2\pi}{3}$. D. h. genau drei oder vier aufeinanderfolgende ganzzahlige Stellen $m$ fallen in eine QP und wir argumentieren analog.

Schließlich bleibt der Fall $\varepsilon = 1$. Hier gilt nach (3.1) einfach $y(m) = -y(m-2)$. Dies ergibt die Periode $(-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2})$ und damit insgesamt ein Monte-Carlo-Verhalten mit einer Periode der Länge 12:

| $m$ mod 3 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Spielwert | G | G | V | G | G | V | G | G | V | G | G | V |
| optimaler Zug | $1, 2$ | 2 | – | 1 | 2 | – | 1 | 2 | – | 1 | 2 | – |
| $\varepsilon = 1$: $y(m)$ | $-\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $-\frac{1}{2}$ |
| MC($\infty$)-Agent | $1, \boxed{2}$ | 2 | $1, 2$ | 1 | $\boxed{1}$,2 | $1, 2$ | 1,$\boxed{2}$ | $\boxed{1}$ | $1, 2$ | $\boxed{2}$ | $\boxed{1}$,2 | $1, 2$ |

□

Als Folgerung der Sätze 3.4 und 3.8 erhalten wir unmittelbar folgenden Satz.

**Satz 3.9.** *Für das Subtraktionsspiel $S_{(1,2)}(m)$ existieren unendlich viele Haufengrößen $m$, die Gewinnstellungen entsprechen, d. h. $m \not\equiv 0 \pmod 3$, bei denen ein statisch-asymmetrischer MC($\infty$)-Agent in eine Verluststellung zieht.*

Die Abbildungen 3.2 und 3.3 veranschaulichen den kompletten Sachverhalt anhand der numerisch ausgewerteten Rekursionsformel (3.1). Es ist zu erkennen, dass die beiden Fälle reeller und komplexer Fundamentallösungen direkt ineinander übergehen.

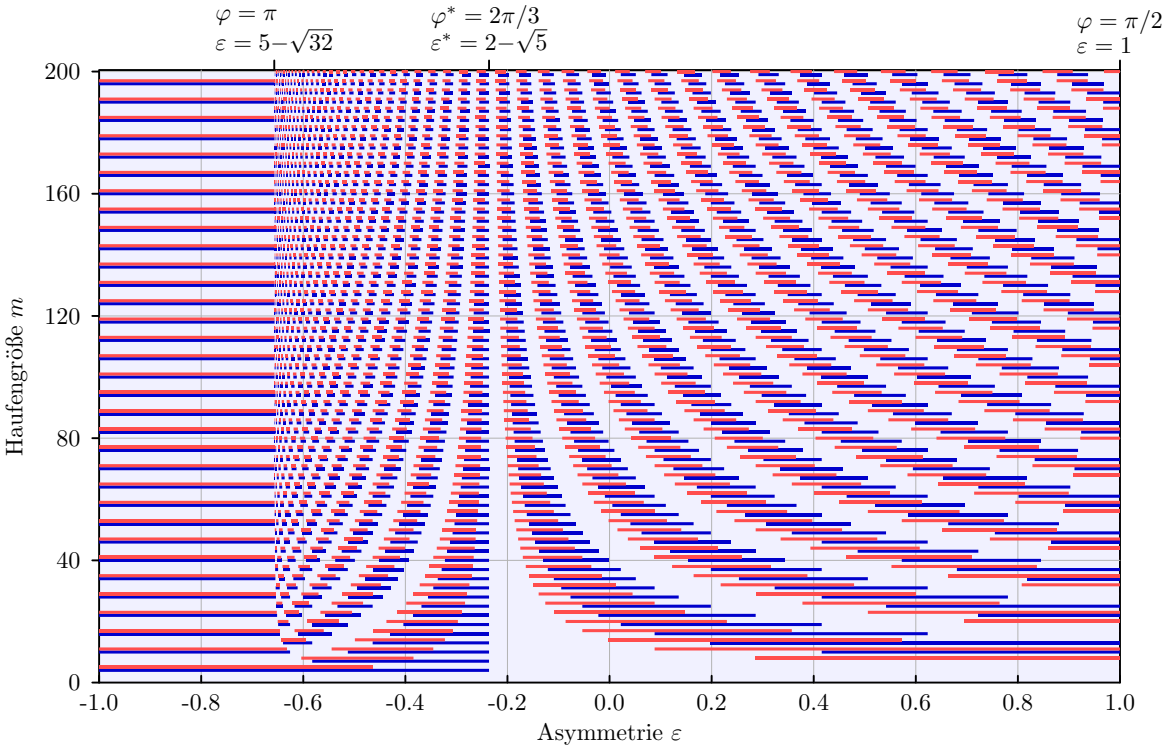



**Abbildung 3.2:** Ungünstige Züge statisch-asymmetrischer Monte-Carlo-Agenten im Subtraktionsspiel $S_{(1,2)}$ für Haufengrößen $m \leq 200$. Blau: $m \equiv 1 \pmod 3$, rot: $m \equiv 2 \pmod 3$. Der Grenzfall $\varepsilon = 1$ ist in der Abbildung nur zu erahnen (dort ist nochmal ein Sprung).

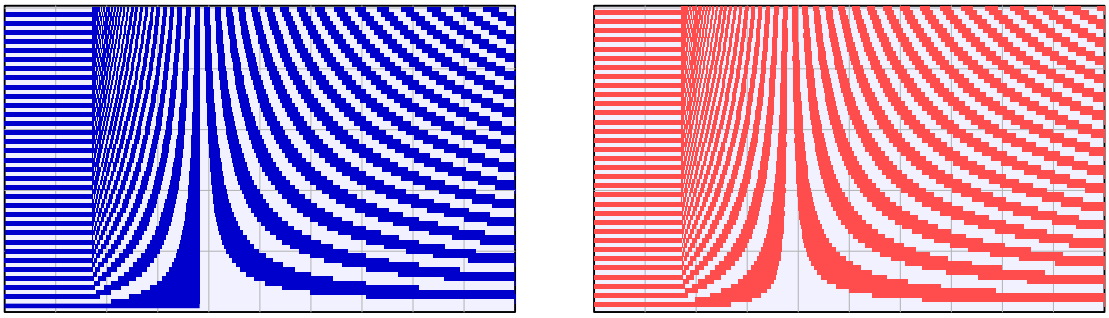


**Abbildung 3.3:** Ungünstige Züge statisch-asymmetrischer Monte-Carlo-Agenten für Haufengrößen $m \leq 200$, getrennt nach $m \equiv 1 \pmod 3$ (blau, links) und $m \equiv 2 \pmod 3$ (rot, rechts), vgl. Abbildung 3.2. Die zu den anderen $m$-Werten gehörenden Zeilen wurden weggelassen.

Das Entstehen einer Kante für $m$ kongruent 1 modulo 3 (blau) bei $\varepsilon^* = 2 - \sqrt{5}$ wird durch die Tabelle in Beispiel 3.7 plausibel.

Dieser kritische Wert $\varepsilon^*$ teilt das Intervall $[-1, 1]$ exakt im Goldenen Schnitt.

## 3.4 Intelligent-asymmetrische Monte-Carlo-Agenten

In diesem Abschnitt ordnen wir in den Zufallspartien den Zugmöglichkeiten Wahrscheinlichkeiten zu, die vom Spielwert der jeweiligen Stellung abhängen sollen. Dies ist ein stark vereinfachtes Modell, welches eine gewisse Intelligenz bei der Auswahl der Züge berücksichtigt, die z. B. durch eine einfache Heuristik realisiert sein könnte.

Wir beschränken die theoretischen Betrachtungen wieder auf das Subtraktionsspiel $\mathrm{S}_{(1,2)}$. Dieses Modell wurde schon in [40] behandelt. Durch eine analytische Untersuchung, ähnlich wie bei den statisch-asymmetrischen Monte-Carlo-Agenten, werden wir hier einige der in [40] geäußerten Vermutungen beweisen.

Die Siegquoten $x(m)$ von Zufallspartien mit Starthaufengröße $m$ eines intelligent-asymmetrischen Monte-Carlo-Agenten lassen sich wiederum rekursiv berechnen:

$$\begin{aligned} x(m) &= p_m \cdot [1 - x(m-1)] + (1 - p_m) \cdot [1 - x(m-2)] \\ &= 1 - p_m \cdot x(m-1) - (1 - p_m) \cdot x(m-2) \\ &\text{mit den Startwerten } x(0) = 0 \text{ und } x(1) = 1\,. \end{aligned}$$

Wiederum bekommen wir durch die lineare Transformation $x(m) = \frac{1}{2} + y(m)$ eine homogene Rekursionsgleichung:

$$\begin{aligned} y(m) &= -p_m \cdot y(m-1) - (1 - p_m) \cdot y(m-2) \\ &\text{mit den Startwerten } y(0) = -\frac{1}{2} \text{ und } y(1) = \frac{1}{2}\,. \end{aligned} \tag{3.8}$$

Die Koeffizienten $p_m$ und $(1-p_m)$ sind die Wahrscheinlichkeiten, bei Haufengröße $m$ ein bzw. zwei Elemente wegzunehmen.

Die Zugwahrscheinlichkeiten $p_m$ sollen vom Spielwert abhängen, welcher im Subtraktionsspiel $\mathrm{S}_{(1,2)}$ eine Periode der Länge drei aufweist (vgl. Beispiel 3.2). Deshalb fordern wir von den Zugwahrscheinlichkeiten $p_m$ dieselbe Periodizität ein, also

$$y(3k+j) = -p_j \cdot y(3k+j-1) - (1-p_j) \cdot y(3k+j-2), \quad j \in \{0,1,2\}\,. \tag{3.9}$$

Eliminiert man aus den fünf linearen Gleichungen

$$\begin{aligned}
y(3k+2) &= -p_2 \cdot y(3k+1) - (1-p_2) \cdot y(3k)\\
y(3k+3) &= -p_0 \cdot y(3k+2) - (1-p_0) \cdot y(3k+1)\\
y(3k+4) &= -p_1 \cdot y(3k+3) - (1-p_1) \cdot y(3k+2)\\
y(3k+5) &= -p_2 \cdot y(3k+4) - (1-p_2) \cdot y(3k+3)\\
y(3k+6) &= -p_0 \cdot y(3k+5) - (1-p_0) \cdot y(3k+4)
\end{aligned}$$

die Variablen $y(3k+1)$, $y(3k+2)$, $y(3k+4)$ und $y(3k+5)$, so erhält man eine Gleichung für $y(3k)$, $y(3k+3)$ und $y(3k+6)$:

$$\begin{aligned}
y(3k+6) = &-(p_0p_1p_2 + p_0p_1 + p_1p_2 + p_2p_0 - p_0 - p_1 - p_2) \cdot y(3k+3)\\
&- (1-p_0)(1-p_1)(1-p_2) \cdot y(3k)\,.
\end{aligned}$$

Da diese Formel invariant gegen zyklische Vertauschung von $p_0, p_1, p_2$ ist, gelten für die beiden anderen Restklassen modulo 3 dieselbe Beziehungen, zusammengefasst:

**Lemma 3.10.** *Für die Lösungen der periodischen Rekursion aus Gleichung (3.9) gilt*

$$\begin{aligned}
y(m+6) = &-(p_0p_1p_2 + p_0p_1 + p_1p_2 + p_2p_0 - p_0 - p_1 - p_2) \cdot y(m+3)\\
&- (1-p_0)(1-p_1)(1-p_2) \cdot y(m)\,.
\end{aligned} \tag{3.10}$$

An dieser Stelle wollen wir [40] folgen und die Zugwahrscheinlichkeiten konkretisieren:

$$p_0 = \frac{1}{2}, \quad p_1 = \frac{1+\varepsilon}{2}, \quad p_2 = \frac{1-\varepsilon}{2} \quad \text{mit } \varepsilon \in [-1,1]\,.$$

D. h. in den Verluststellungen werden beide Züge gleich behandelt und in den Gewinnstellungen werden die Gewinnzüge für $\varepsilon > 0$ bevorzugt. Die Bedingung $p_1 = 1-p_2$ ist zwar nicht unbedingt nötig, sie macht aber die folgenden Rechnung überschaubarer.

Eingesetzt in (3.10) ergibt sich

$$y(m+6) = \frac{1}{8}\left((5+3\varepsilon^2) \cdot y(m+3) - (1-\varepsilon^2) \cdot y(m)\right) \tag{3.11}$$

mit den Startwerten (gemäß (3.8))

$$\begin{aligned}
y(0) &= -\frac{1}{2} & y(3) &= -\frac{1+\varepsilon}{4}\\
y(1) &= \frac{1}{2} & y(4) &= \frac{1+\varepsilon^2}{8}\\
y(2) &= \frac{\varepsilon}{2} & y(5) &= \frac{1+5\varepsilon-\varepsilon^2+3\varepsilon^3}{16}\,.
\end{aligned}$$

Das zugehörige charakteristische Polynom hat die Lösungen[5]

$$\begin{aligned}\lambda_1 &= \frac{5+3\varepsilon^2+\sqrt{(\varepsilon^2+7)(9\varepsilon^2-1)}}{16}\,,\\ \lambda_2 &= \frac{5+3\varepsilon^2-\sqrt{(\varepsilon^2+7)(9\varepsilon^2-1)}}{16}\,.\end{aligned} \tag{3.12}$$

Bevor wir wiederum nach reellen und komplexen Lösungen unterscheiden, werden wir drei wichtige Formeln angeben, die uns beim weiteren Vorgehen nützlich sein werden. Die ersten beiden findet man in [40, Satz 2.4 bzw. Satz 2.5]:

**Lemma 3.11.** *Für $-1 < \varepsilon < 1$ gilt*

$$y(3k) - y(3k+1) = \frac{4}{1+\varepsilon} y(3k+3)\,, \tag{3.13}$$

$$y(3k) - y(3k-1) = \frac{3\varepsilon+1}{2} y(3k-3) + \frac{(1-3\varepsilon)(1-\varepsilon)}{8}(y(3k-3) - y(3k-4)) \tag{3.14}$$

$$= \frac{3\varepsilon-1}{1+\varepsilon} y(3k) + y(3k-3)\,. \tag{3.15}$$

D. h. ob der Monte-Carlo-Agent einen ungünstigen Zug macht, hängt in einfacher Weise von den Siegquoten der Verluststellungen bzw. in (3.14) von der Beurteilung einer kleineren Instanz ab.[6]

**Beweis des Lemmas**. Sukzessives Ersetzen der $y$-Terme mit höchstem Index, gemäß Rekursionsformel (3.9), zeigt durch elementare Rechnung die Richtigkeit der Behauptungen. □

### 3.4.1 Reelle Fundamentallösungen

Anhand der Diskriminante in (3.12) ist ersichtlich, dass die Lösungen genau für $|\varepsilon| \geq \frac{1}{3}$ reell sind.

**Lemma 3.12.** *Für $-1 < \varepsilon \leq -\frac{1}{3}$ bzw. $\frac{1}{3} \leq \varepsilon \leq 1$ gilt $y(3k) < 0$. Für $\varepsilon = -1$ und $k > 0$ gilt $y(3k) = 0$.*

**Beweis**. Zunächst gilt $0 \leq \lambda_2 \leq \frac{1}{3} \leq \lambda_1 \leq 1$, was sich wiederum durch elementare Umformungen unter Beachtung von $\frac{1}{9} \leq \varepsilon^2 \leq 1$ leicht zeigen lässt. Gleichheit tritt dabei nur an den Intervallgrenzen auf.

[5] Jeweils mit dem Ansatz $y(3k+j) = \lambda^k$, für $j \in \{0,1,2\}$.
[6] Details dazu folgen im Beweis von Satz 3.13.

1. Für $\frac{1}{3} < |\varepsilon| < 1$ führt der Ansatz $y(3k) = \alpha_1\lambda_1^k + \alpha_2\lambda_2^k$ mit den Startwerten $y(0) = -\frac{1}{2}$ und $y(3) = -\frac{1+\varepsilon}{4}$ auf

$$\alpha_1 = \frac{2\lambda_2 - 1 - \varepsilon}{4(\lambda_1 - \lambda_2)} \quad \text{und} \quad \alpha_2 = \frac{1 + \varepsilon - 2\lambda_1}{4(\lambda_1 - \lambda_2)}\,. \tag{3.16}$$

   Im Fall $\varepsilon > \frac{1}{3}$ gilt $\alpha_2 > 0$, was sich ebenso durch elementare Umformungen zeigen lässt, man beachte, dass der Nenner $(\lambda_1 - \lambda_2)$ positiv ist. Wegen $\alpha_1 + \alpha_2 = y(0) = -\frac{1}{2}$ gilt $\alpha_1 < -\frac{1}{2}$ und $|\alpha_2| < |\alpha_1|$. Die Folge $y(3k)$ wird also durch den negativen Term $\alpha_1\lambda_1^k$ absolut dominiert.

   Im Fall $\varepsilon < -\frac{1}{3}$ gilt $-\frac{1}{4} < \alpha_1 < 0$, was sich ebenso durch elementare Umformungen zeigen lässt. Folglich ist $-\frac{1}{2} < \alpha_2 < -\frac{1}{4}$, d. h. beide Koeffizienten sind negativ. Also sind auch alle Folgenglieder negativ.

2. Für $|\varepsilon| = \frac{1}{3}$ führt der Ansatz $y(3k) = \beta_1\lambda^k + \beta_2 k\lambda_2^k$ mit $\lambda = \frac{1}{3}$ und den Startwerten $y(0) = -\frac{1}{2}$ und $y(3) = -\frac{1+\varepsilon}{4}$ zu den expliziten Formeln

$$y(3k) = \begin{cases} -\frac{1}{2} \cdot \left(\frac{1}{3}\right)^k & \text{falls } \varepsilon = -\frac{1}{3} \\ -\frac{1}{2} \cdot (1+k) \cdot \left(\frac{1}{3}\right)^k & \text{falls } \varepsilon = \frac{1}{3}\,, \end{cases} \tag{3.17}$$

   also ist $y(3k) < 0$ erfüllt.

3. Falls $|\varepsilon| = 1$ folgt aus (3.11), dass $y(m+6) = y(m+3)$, $m \geq 0$ gilt. Mit dem Startwert $y(3) = -\frac{1+\varepsilon}{4}$ folgt $y(3k) = -\frac{1}{2}$ für $\varepsilon = 1$ und $y(3k) = 0$ für $\varepsilon = -1$ und $k > 0$. □

Zusammen mit Lemma 3.11 ergibt sich folgende Charakterisierung.

**Satz 3.13.** *Sei $\frac{1}{3} \leq |\varepsilon| \leq 1$. Dann gilt:*

1. *Für $\varepsilon \geq \frac{1}{3}$ macht der intelligent-asymmetrische Monte-Carlo-Agent stets einen optimalen Zug.*
2. *Für $\varepsilon \leq -\frac{1}{3}$ macht der intelligent-asymmetrische Monte-Carlo-Agent für Haufengrößen kongruent 2 modulo 3 stets einen optimalen Zug.*
3. *Für $\varepsilon < -\frac{1}{3}$ macht der intelligent-asymmetrische Monte-Carlo-Agent für Haufengrößen kongruent 1 modulo 3 stets einen ungünstigen Zug, mit Ausnahme der Haufengröße 1.*
4. *Für $\varepsilon = -\frac{1}{3}$ macht der intelligent-asymmetrische Monte-Carlo-Agent für Haufengrößen kongruent 1 modulo 3 mit Wahrscheinlichkeit $\frac{1}{2}$ einen ungünstigen Zug, mit Ausnahme der Haufengröße 1.*

**Beweis**. Wir unterscheiden modulo 3 die beiden Haufengrößen in denen ungünstige Züge auftreten können. Sei zunächst $\frac{1}{3} \leq |\varepsilon| < 1$.

Für Haufengrößen kongruent 2 modulo 3 folgt die Behauptung direkt aus Lemma 3.12, d. h. $y(3k) < 0$, und Gleichung (3.13). Denn $y(3k) - y(3k+1) < 0$ bedeutet, dass der Agent von $(3k+2)$ nach $(3k)$ zieht (optimaler Zug).

Für $\frac{1}{3} \leq \varepsilon < 1$ und Haufengrößen kongruent 1 modulo 3 folgt die Behauptung aus Gleichung (3.15), denn beide Summanden auf der rechten Seite sind negativ.

Für $-1 < \varepsilon < -\frac{1}{3}$ gilt $y(3) - y(2) = -\frac{\varepsilon+1}{4} - \frac{\varepsilon}{2} = \frac{-3\varepsilon-1}{4} > 0$ und damit rekursiv $y(3k) - y(3k-1) > 0$, denn in Gleichung (3.14) sind beide Summanden auf der rechten Seite positiv. D. h. für alle Haufengrößen $m$ kongruent 1 modulo 3 mit $m > 1$ macht der Agent den falschen Zug.

Für $\varepsilon = -\frac{1}{3}$ gilt $y(3k)-y(3k-1) = 0$ wegen (3.15) und (3.17). Gemäß Tie-Break-Regel macht der Agent dann in 50 % der Fälle den ungünstigen Zug.

Es verbleiben noch die (für das Modell eher uninteressanten) Fälle $\varepsilon = \pm 1$. Nach Gleichung (3.11) gilt $y(m+6) = y(m+3)$, $m \geq 0$, die Folge ist also periodisch. Es ergibt sich aus den Startwerten $y(0), \ldots, y(5)$ das folgende Bild

| | $y(0)$ | $y(3k+1)$ | $y(3k+2)$ | $y(3k+3)$ |
|---|---|---|---|---|
| $\varepsilon = 1$ | $-\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $-\frac{1}{2}$ |
| $\varepsilon = -1$ | $-\frac{1}{2}$ | $\frac{1}{2}$ | $-\frac{1}{2}$ | $0$ |

Für $\varepsilon = 1$ entspricht das natürlich genau dem Spielwert und der Agent zieht optimal. Für $\varepsilon = -1$ zieht der Agent von $(3k+2)$ nach $3k$, also optimal, und von $(3k+1)$ nach $(3k-1)$, also ungünstig. □

### 3.4.2 Komplexe Fundamentallösungen

Im Fall $|\varepsilon| < \frac{1}{3}$ sind die Lösungen (3.12) komplex und zueinander konjugiert. Aus (3.16) mit $\lambda = \lambda_1$ und $\overline{\lambda} = \lambda_2$ folgt

$$y(m) = \frac{1}{4(\lambda - \overline{\lambda})} \left( (2\overline{\lambda} - 1 - \varepsilon)\lambda^m - (2\lambda - 1 - \varepsilon)\overline{\lambda}^m \right).$$

Eine zum statischen Fall ganz analoge Rechnung führt zum folgenden Ergebnis:

**Lemma 3.14.** *Für jedes $|\varepsilon| < \frac{1}{3}$ gilt für die Siegquote intelligent-asymmetrischer Monte-Carlo-Agenten*

$$\boxed{\begin{aligned} y(3k) &= -\frac{\sqrt{(1-\varepsilon^2)(3\varepsilon+1)}}{8b}\, r^k \cos(k\varphi+\varphi_0) \\ \textit{mit}\quad r &= \sqrt{\frac{1-\varepsilon^2}{8}} \\ a &= \frac{5+3\varepsilon^2}{16} \\ b &= \frac{\sqrt{(\varepsilon^2+7)(1-9\varepsilon^2)}}{16} \\ \varphi &= \arctan\frac{b}{a} \\ \varphi_0 &= -\arctan\frac{1+\varepsilon-2a}{2b} = -\arctan\frac{(3-\varepsilon)(3\varepsilon+1)}{16b}\,. \end{aligned}} \tag{3.18}$$

Dies ist wiederum eine gedämpfte Kosinus-Funktion. Deren Frequenz $\varphi$ ist nach oben beschränkt und es gilt folgender Satz:

**Satz 3.15.** *Für jedes $|\varepsilon| < \frac{1}{3}$ hat die Funktion $y(3k)$ aus Lemma (3.14) an jeweils mindestens sechs aufeinanderfolgenden ganzzahligen Werten für $k$ dasselbe Vorzeichen (die erste Serie ab $y(0) = -\frac{1}{2}$ kann auch kürzer sein).*

**Beweis**. Wegen

$$\varphi = \arctan\frac{b}{a} = \arctan\frac{\sqrt{7-62\varepsilon^2-9\varepsilon^4}}{5+\varepsilon^2} < \arctan\frac{\sqrt{7}}{5} < \frac{\pi}{6}$$

entfallen auf jede Quasi-Halbperiode mindestens sechs ganzzahlige Stützstellen. □

Die Beurteilung der Züge durch Lemma 3.11 erlaubt in Verbindung mit Satz 3.15 eine recht genaue Charakterisierung ungünstiger Züge des Monte-Carlo-Agenten:

**Satz 3.16.** *Sei $-\frac{1}{3} < \varepsilon < \frac{1}{3}$. Dann gilt*

1. *Für Haufengrößen $m$ kongruent 2 modulo 3 gibt es quasi-periodisch Serien mit ungünstigen Zügen des intelligent-asymmetrischen Monte-Carlo-Agenten.*
2. *Ist $m = 3k+2$ die Haufengröße des ersten ungünstigen Zuges einer solchen Serie, so zieht der Agent bei Haufengröße $m = 3k+4$ richtig.*
3. *Ist $m = 3k+2$ die Haufengröße des letzten ungünstigen Zuges einer solchen Serie, so zieht der Agent bei Haufengröße $m = 3k+7$ auch ungünstig.*

**Beweis**. Die erste Aussage folgt direkt aus Satz 3.15 in Verbindung mit der Formel (3.13) aus Lemma 3.11.

Für die zweite und dritte Aussage bemühen wir ebenfalls das Lemma 3.11. Die Bezeichnung $Z(m)$ soll angeben, ob der MC($\infty$)-Agent bei Haufengröße $m$ einen

günstigen Zug vorschlägt, wobei wir etwaige Tie-Breaks unberücksichtigt lassen wollen. Dann gilt für das erste ungünstige $Z(3k+2)$:

$$\left.\begin{aligned} Z(3k+2) \text{ ungünstig} &\iff y(3k) > y(3k+1) \overset{(3.13)}{\iff} y(3k+3) > 0 \\ Z(3k-1) \text{ günstig} &\iff y(3k-3) > y(3k-2) \overset{(3.13)}{\iff} y(3k) < 0 \end{aligned}\right\}$$

$$\begin{aligned} &\implies \underbrace{\frac{3\varepsilon-1}{1+\varepsilon}}_{<0} y(3k+3) + y(3k) < 0 \\ &\overset{(3.15)}{\iff} y(3k+3) < y(3k+2) \\ &\iff Z(3k+4) \text{ günstig}. \end{aligned}$$

Analog argumentieren wir für das letzte ungünstige $Z(3k+2)$:

$$\left.\begin{aligned} Z(3k+2) \text{ ungünstig} &\iff y(3k) > y(3k+1) \overset{(3.13)}{\iff} y(3k+3) > 0 \\ Z(3k+5) \text{ günstig} &\iff y(3k+3) > y(3k+4) \overset{(3.13)}{\iff} y(3k+6) < 0 \end{aligned}\right\}$$

$$\begin{aligned} &\implies \underbrace{\frac{3\varepsilon-1}{1+\varepsilon}}_{<0} y(3k+6) + y(3k+3) > 0 \\ &\overset{(3.15)}{\iff} y(3k+6) > y(3k+5) \\ &\iff Z(3k+7) \text{ ungünstig}. \end{aligned}$$

□

Die Abbildung 3.4 veranschaulicht den kompletten Sachverhalt anhand der numerisch ausgewerteten Rekursionsformel (3.8).

Der Satz 3.16 besagt, dass sich für $-\frac{1}{3} < \varepsilon < \frac{1}{3}$ überlappende Serien ausbilden. Dabei befinden sich unterhalb des jeweils ersten blauen „Streifens“ mindestens zwei rote und oberhalb des letzten roten mindestens zwei blauen Streifen.

## 3.5 Diskussion

Für statisch-asymmetrische MC($\infty$)-Agenten ergab sich folgendes Bild (vgl. Abbildung 3.2):

- Für jedes $\varepsilon$ gibt es unendlich viele Haufengrößen, bei denen der Agent einen ungünstigen Zug vorschlägt.
- Bei $\varepsilon^* = 2 - \sqrt{5}$ gibt es nur ungünstige Züge bei Haufengrößen kongruent 1 modulo 3. Bei allen Haufengrößen $(3k+1)$, $k > 0$ liegen in diesem Fall Tie-Breaks vor, d. h. ungünstige Züge treten nur mit Wahrscheinlichkeit von 50 % auf.

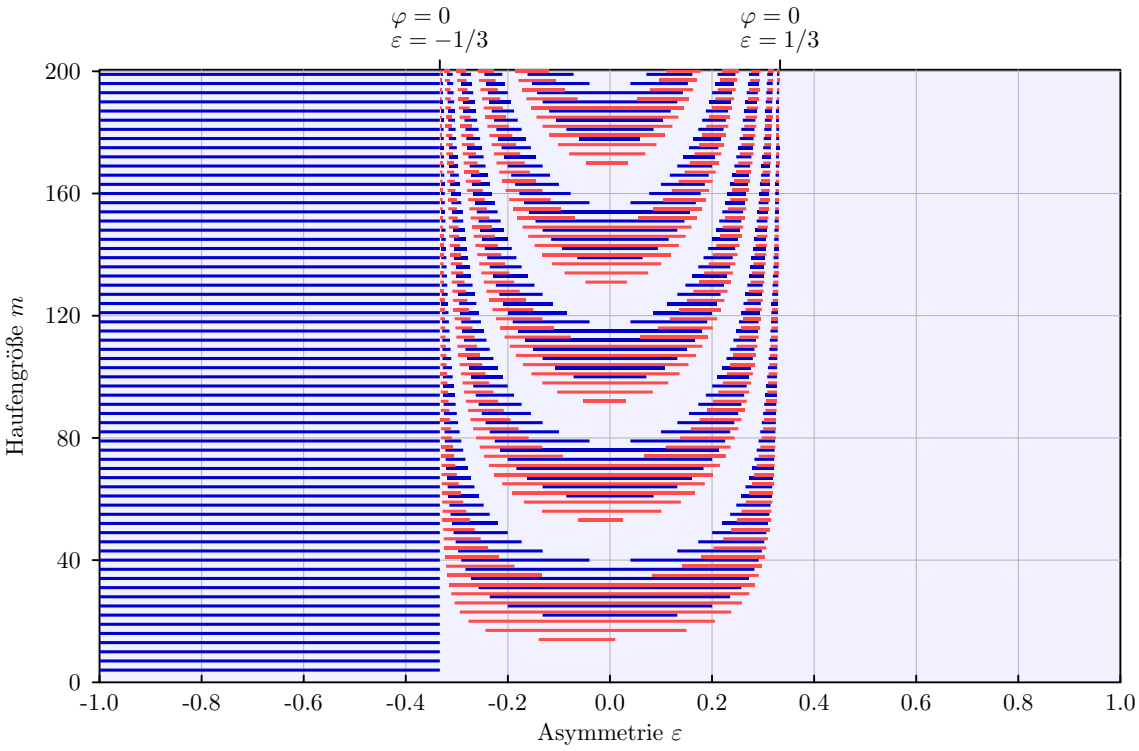



**Abbildung 3.4:** Ungünstige Züge intelligent-asymmetrischer Monte-Carlo-Agenten im Subtraktionsspiel $S_{(1,2)}$ für Haufengrößen $m \leq 200$. Blau: $m \equiv 1 \pmod 3$, rot: $m \equiv 2 \pmod 3$.

- Für $\varepsilon^* \neq 2 - \sqrt{5}$ gibt es für sowohl für Haufengrößen kongruent 1 als auch kongruent 2 modulo 3 jeweils unendlich viele Haufengrößen, bei denen der Agent einen ungünstigen Zug vorschlägt.
- Für $\varepsilon^* < 2 - \sqrt{5}$ wird stets schon bei Haufengröße 4 ungünstig gezogen, wie man aus dem Vergleich von $y(2) = \frac{\varepsilon}{2}$ mit $y(3) = \frac{\varepsilon^2 - 2\varepsilon - 1}{4}$ schnell erkennt.
- Für $\varepsilon^* > 2 - \sqrt{5}$ tritt der erste ungünstige Zug mit wachsendem $\varepsilon$ immer früher auf.

Eine kleine statische Asymmetrie verändert das Verhalten von MC($\infty$)-Agenten beim Subtraktionsspiel $S_{(1,2)}$ zunächst nur wenig. Für größere Asymmetrien ergeben sich jedoch für positive und negative Parameter $\varepsilon$ qualitativ unterschiedliche Phänomene.

Für intelligent-asymmetrische MC($\infty$)-Agenten ergab sich folgendes Bild:

- Für jedes $\varepsilon \geq \frac{1}{3}$ spielt der Agent optimal.
- Für jedes $\varepsilon < -\frac{1}{3}$ spielt der Agent für Haufengrößen kongruent 2 modulo 3 optimal. Kongruent 1 modulo 3 wird ab Haufengröße 4 immer ein ungünstiger Zug gespielt.
- Bei $\varepsilon = -\frac{1}{3}$ haben wir dieselbe Situation wie bei beim statisch-asymmetrischen Fall mit $\varepsilon^* = 2 - \sqrt{5}$.

- Für jedes $|\varepsilon| < \frac{1}{3}$ gibt es unendlich viele Haufengrößen, bei denen der Agent einen ungünstigen Zug vorschlägt.

  Die ungünstigen Züge für Haufengrößen kongruent 1 bzw. 2 modulo 3 bilden Serien mit mindestens 6 guten bzw. schlechten Zügen. Die Serie mit Haufengrößen kongruent 2 modulo 3 ist leicht nach unten verschoben.

  Die Serien mit Haufengrößen kongruent 1 modulo 3 ist symmetrisch bzgl. $\varepsilon = 0$.

Erstaunlicherweise zeigt sich sowohl für „positive“ als auch „negative“ Intelligenz eine Verbesserung des Agenten. Im Limes $\varepsilon \to \pm\frac{1}{3}$ (von Null kommend) wird der Agent sogar beliebig gut. Aufgrund der Asymmetrie der o. g. Serien mit Haufengrößen kongruent 2 modulo 3 ist die Verbesserung für positive $\varepsilon$ aber schneller und für $\varepsilon \geq \frac{1}{3}$ hat der Agent genug „Wissen“, um perfekt zu spielen.

Für den intelligent-asymmetrischen Fall gab es die Vorarbeit [40]. Diese behandelte nur den Fall $\varepsilon > 0$. Für $\varepsilon < \frac{1}{3}$ wurde die Existenz unendlich vieler Haufengrößen mit ungünstigen Zügen zwar vermutet, aber nicht bewiesen. Der Satz 2.5 aus [40] zeigte, dass die erste Haufengrößen mit ungünstigem Zug stets kongruent 2 modulo 3 ist. Beim Verallgemeinern dieses Satzes auf negative $\varepsilon$ stellte sich heraus, dass er für $\varepsilon \leq -\frac{1}{3}$ nicht mehr stimmt.[7]

Die explizite Herleitung für die Siegwahrscheinlichkeiten der Monte-Carlo-Partien erlaubt es, die Serien-Struktur von günstigen bzw. ungünstigen Zügen zu erklären und quantitative Aussagen abzuleiten. Dazu genügt es, die Frequenz $\varphi$ aus Lemma 3.5 bzw. 3.14 auszuwerten.

**Beispiel 3.17.** *Wir betrachten den symmetrischen Fall, d. h. $\varepsilon = 0$. Die Frequenz der gedämpften Kosinus-Funktion beträgt nach (3.7)*

$$\varphi = \frac{\pi}{2} + \arctan\frac{-a}{b} = \pi - \arctan\frac{b}{-a} = \pi - \arctan\sqrt{7} \approx 1.932\,.$$

*Demgegenüber hat die perfekte Strategie die Periodenlänge drei, also eine Frequenz von $\tilde{\varphi} = \frac{2}{3}\pi \approx 2.094$. Durch Überlagerung beider Frequenzen kommt es zu einer Schwebung mit der Periodendauer $\tau = \frac{2\pi}{\tilde{\varphi}-\varphi} = 38.730$.*

*Das ist genau die Periode, die man Abbildung 3.2 bzw. Abbildung 3.4 für $\varepsilon = 0$ zu sehen ist (vgl. auch Abb. 2.2 in [40]).*

Unter Berücksichtigung des Phasenwinkels $\varphi_0$ aus Lemma 3.5 bzw. 3.14 lässt sich die Dauer bis zum ersten ungünstigen Zug abschätzen.

[7] Interessanterweise liegt die Ursache dafür nur in dem in [40] nicht explizit betrachteten Induktionsanfang $y(3) < y(2)$.

**Beispiel 3.18.** *Wir betrachten die Asymptotik des intelligent-symmetrischen Falles für $\varepsilon \to \frac{1}{3}$. Nach (3.13) tritt der erste ungünstige Zug mit Haufengröße $m = 3k - 1$ dann auf, wenn $y(3k)$ negativ wird. Dies passiert, wenn das Argument $k\varphi + \varphi_0$ der Kosinus-Funktion in (3.18) den Wert $\frac{\pi}{2}$ überschreitet. Wir berechnen die kritische Stelle $m^*$:*

$$\frac{m^* - 1}{3}\varphi + \varphi_0 = \frac{\pi}{2}\,.$$

*Für die gesuchte Asymptotik setzen wir $\varepsilon = \frac{1}{3} - \delta$ und betrachten nur die höchste Ordnung bzgl. $\delta \to 0$. Wegen $b \to 0$ gilt $\varphi_0 \to -\frac{\pi}{2}$. Außerdem ist*

$$\varphi = \arctan\frac{b}{a} \approx \arctan\frac{\sqrt{\frac{1}{9} + 7} \cdot \sqrt{6\delta}}{5 + \frac{1}{3}} = \frac{1}{2}\sqrt{6\delta}\,,$$

*und damit*

$$m^* = \frac{3(\frac{\pi}{2} - \varphi_0)}{\varphi} - 1 \approx \frac{\pi\sqrt{6}}{\sqrt{\delta}} \approx 7.6953 \cdot \frac{1}{\sqrt{\delta}}\,.$$

*Diese Asymptote passt sehr gut zu den in [40] angegebenen tatsächlichen Werten $m^*(\delta = 10^{-2}) = 74$, $m^*(\delta = 10^{-4}) = 767$ und $m^*(\delta = 10^{-6}) = 7694$.*

## 3.6 Ausblick und offene Fragen

Mit der vorgestellten Methode in ggf. verallgemeinerter Form können weitere Fragestellungen untersucht werden.

Fragen, die für das Subtraktionsspiel $\mathrm{S}_{(1,2)}$ beantwortet werden konnten, stellen sich auch für andere Zwei- oder gar Mehrzug-Subtraktionsspiele. Zunächst könnte man den Fall $\mathrm{S}_{(2,3)}$ untersuchen, der auf ein kubisches charakteristisches Polynom führt und damit ebenfalls vollständig analytisch lösbar sein dürfte.

Für das Subtraktionsspiel $\mathrm{S}_{(4,7)}$ sind die experimentellen Befunde für den statisch-asymmetrischen Fall in Abbildung 3.5 dargestellt. $\mathrm{S}_{(4,7)}$ hat eine Grundperiode von 11 (Gewinnstellungen bei Haufengröße $m = 11k + t$, $t = 1, \ldots, 7$ ). Außer bei $t = 4$ treten wieder regelmäßig ungünstige Züge auf. Es gibt wieder ein kritisches $\varepsilon^*$, jedoch für $t = 2$ und $t = 6$ eine obere Schranke für die kritische Haufengröße.

Für dieses und weitere Subtraktionsspiele ergibt sich ein charakteristisches Polynom höheren Grades, dessen Nullstellen sich im Allgemeinen nur noch numerisch bestimmen lassen. Jedoch bleibt die Argumentation über die Dominanz des Terms mit dem betragsmäßig größten Eigenwert erhalten. Es erscheint sogar sinnvoll, statt mit Lemma 3.11 zu operieren, ausschließlich mit der expliziten Darstellung zu argumentieren.

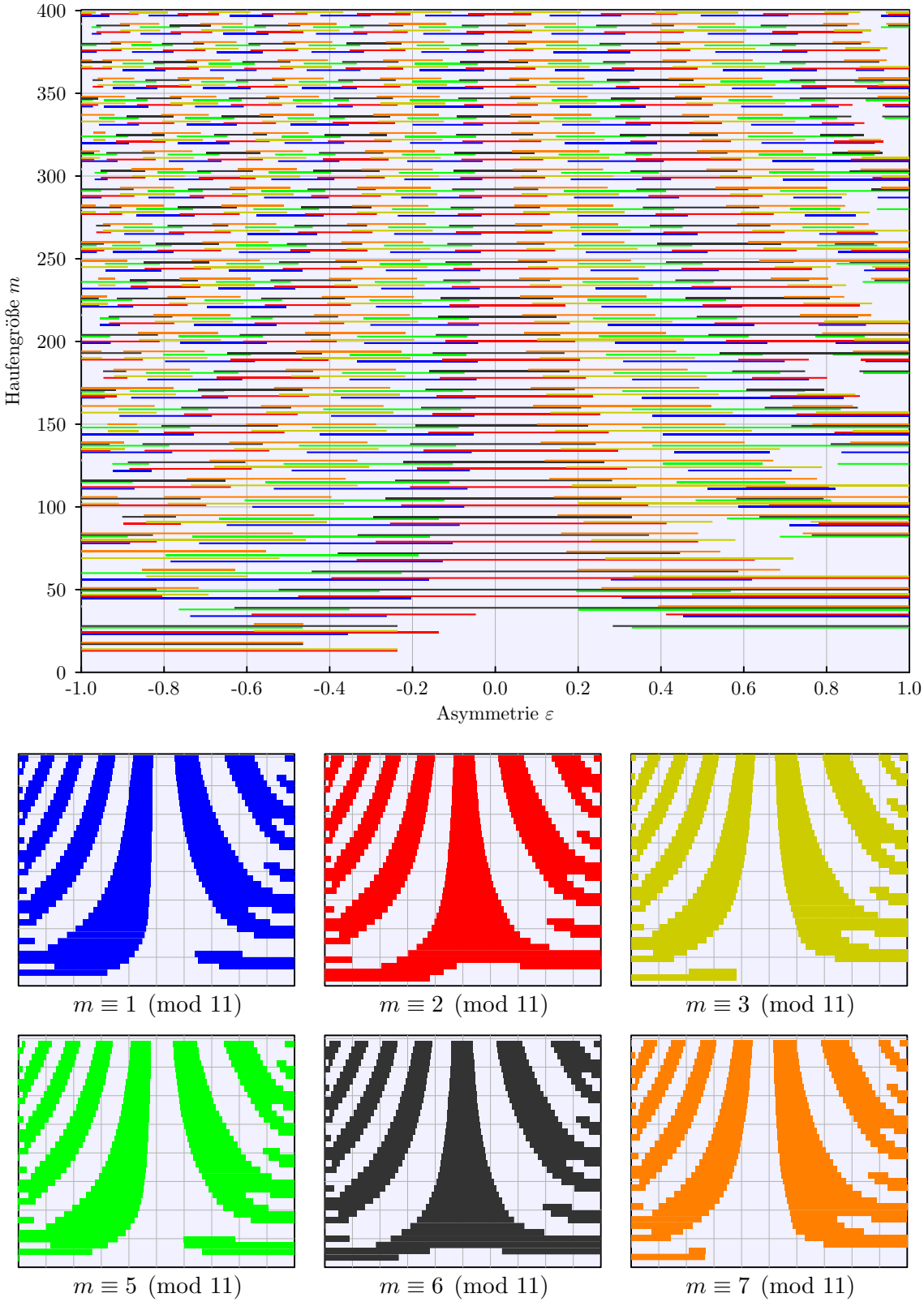



**Abbildung 3.5:** Ungünstige Züge statisch-asymmetrischer Monte-Carlo-Agenten im Subtraktionsspiel $S_{(4,7)}$ für Haufengrößen $m \leq 400$.

**Beispiel 3.19.** *Berechnet man analog zu Gleichung (3.16) auch die Koeffizienten der expliziten Darstellung von $y(3k+2)$, so folgt*

$$y(3k+3) - y(3k+2) = \frac{1+3\varepsilon}{32(\lambda_1 - \lambda_2)} \left[ (5 + 3\varepsilon^2 - 8\lambda_1)\lambda_2^k - (5 + 3\varepsilon^2 - 8\lambda_2)\lambda_1^k \right] .$$

*Da $\lambda_1$ und $\lambda_2$ nur von $\varepsilon^2$ abhängen, kann man das Symmetrieverhalten für Haufengrößen kongruent 1 modulo 3 direkt ablesen (vgl. Abbildung 3.4, nur die blauen Linien). Der Vorfaktor $(1+3\varepsilon)$ (dessen Betrag keinen Einfluss hat) ändert bei $\varepsilon = -\frac{1}{3}$ das Vorzeichen und damit ist das Bild für $|\varepsilon| < \frac{1}{3}$ symmetrisch und für $|\varepsilon| > \frac{1}{3}$ antisymmetrisch.*

Konkrete Fragestellungen für zukünftige Untersuchungen wären z. B.

- Aufgrund der quasi-periodischen Struktur treten ungünstige Züge bei $\mathrm{S}_{(1,2)}$ regelmäßig auf. Im statisch-asymmetrischen Fall sind für $\varepsilon \neq \varepsilon^* = 2 - \sqrt{5}$ asymptotisch die Hälfte aller Züge aus Gewinnstellungen ungünstig, während es im intelligent-asymmetrischen Fall von $\varepsilon$ abhängt.

  Welche Anteile von ungünstigen Zügen können bei Subtraktionsspielen auftreten? Wie hängt dies von der Zugmenge ab?

- Was passiert im intelligent-asymmetrischen Fall für $p_1 \neq 1 - p_2$? Kann man dies als Überlagerung statischer und intelligenter Asymmetrie interpretieren?

- Wie setzt sich die zyklische Koeffzientenstruktur in (3.10) für Rekurrenzen mit Grundperioden größer drei fort?

# Kapitel 4

# Deduktionsspiele und Rényi-Entropie-Strategien

*Wir untersuchen Agenten für Deduktionsspiele, welche auf der Maximierung der Rényi-Entropie, einer Verallgemeinerung der Shannon-Entropie, beruhen. Dazu formulieren wir die Deduktionsspiele als Suchprobleme. Als konkrete Untersuchungsobjekte dienen das bekannte Spiel Mastermind und das neue Spiel Beams, welches eine zweidimensionale Code-Struktur hat. Ziel ist es, Eigenschaften über die Eignung der Rényi-Entropie für Spiel-Agenten zu gewinnen und die Agenten mit perfekten Strategien zu vergleichen. Für Beams auf größeren Spielfeldern sind die perfekten Strategien unbekannt – wir versuchen aber, zumindest gute Vermutungen darüber anzustellen, wie nahe man diesen mit den Rényi-Entropie-Agenten kommt. Die Bewertung der Agenten erfolgt durch vollständige computergestützte Analyse, so dass wir hier stets exakte Ergebnisse präsentieren.*

## 4.1 Einleitung

Bei Deduktionsspielen ist es die Aufgabe des oder der Spieler einen geheimen Code durch Stellen von Fragen zu knacken. Das bekannteste und populärste Spiel dieser Art ist Mastermind, bei welchem es mehrstellige Farbcodes zu erraten gilt. Ein weiteres, zweidimensionales, Deduktionsspiel ist Beams, welches im weiteren Verlauf des Kapitels eingeführt und analysiert wird.

Fasst man diese Spiele als Suchproblem auf, werden durch die Spielregeln der Suchraum und die möglichen Anfragen einschließlich Antwortmöglichkeiten spezifiziert. Oft fällt es menschlichen Spielern nicht leicht, gute von weniger guten Anfragen zu unterscheiden. Demgegenüber ist die Entropie-Maximierung eine einfache Möglichkeit, um recht starke Computer-Agenten zu konstruieren. Voraussetzung ist, dass man die Entropie exakt berechnen kann, was je nach Suchraumgröße z. T. großen Rechenaufwand erfordert.

Im Rahmen dieses Kapitels interessieren wir uns stets für die durchschnittliche Anzahl von Suchanfragen (Spieldauer), die zum Auffinden des geheimen Codes nötig sind.

Dabei gehen wir davon aus, dass der geheime Code zufällig aus der Menge aller Codes gleichwahrscheinlich ausgewählt wird.

Theoretische Abschätzungen liefern in der Regel nur grobe untere Schranken für die minimale durchschnittliche Spieldauer, während nicht perfekte (heuristische) Agenten obere Schranken implizieren. Anhand von Beispielen, bei denen wir in der Lage sind, perfekte Suchstrategien zu bestimmen, zeigen wir, dass entropiebasierte Agenten nahezu perfekt spielen.

## 4.2 Rényi-Entropie und Heuristiken für Suchprobleme

### 4.2.1 Die Rényi-Entropie

Die Rényi-Entropie [41] ist eine Verallgemeinerung der bekannten Shannon-Entropie, welche eine bedeutenden Rolle in der Informationstheorie hat [20]. Das Konzept erlaubt es, eine Reihe klassischer Suchstrategien gemeinsam zu beschreiben, und außerdem interpolierende Strategien zu bilden.

**Definition 4.1.** *Die Rényi-Entropie der Ordnung $\alpha$, $\alpha \geq 0$, ist definiert durch*

$$H_\alpha(p) = \frac{1}{1-\alpha} \log_2 \left( \sum_{i=1}^{n} p_i^\alpha \right) ,$$

*wobei $p = (p_1, \ldots, p_n)$ ein Wahrscheinlichkeitsvektor ist, d. h. $0 \leq p_i$ und $\sum_{i=1}^{n} p_i = 1$.*

Diese Definition ist für die folgenden drei Grenzfälle zu präzisieren.

**1. Fall: $\boldsymbol{\alpha = 0}$.** Im Grenzfall $\alpha \to 0$ ergibt sich

$$H_0(p) = \log_2 \|p\|_0 ,$$

wobei die Null-Norm $\|p\|_0$ gleich der Anzahl der von Null verschiedenen Komponenten von $p$ ist.

**2. Fall: $\boldsymbol{\alpha = 1}$.** Der Grenzwert $\alpha \to 1$ ist über die L'Hôpitalsche Regel erklärt. Man erhält mit

$$H_1(p) = -\sum_{i=1}^{n} p_i \log_2 p_i ,$$

exakt die Shannon-Entropie.

**3. Fall: $\boldsymbol{\alpha = \infty}$.** Da der Grenzwert für $\alpha \to \infty$ existiert, definieren wir

$$H_\infty(p) = -\log_2 \|p\|_\infty$$

wobei $\|p\|_\infty$ die Maximum-Norm von $p$ bezeichnet.

Im Kontext der Suchprobleme ist die Maximierung der Rényi-Entropie direkt mit klassischen Suchstrategien verbunden, wie wir im nächsten Abschnitt zeigen werden. Der Vektor $p$ hat dabei folgende Bedeutung: Auf eine Anfrage mit $n$ Antwortmöglichkeiten ist $p_i$ die Wahrscheinlichkeit, die Antwort $i$ zu erhalten.

### 4.2.2 Suchprobleme

Beams ist genau wie Mastermind ein Suchproblem der folgender Struktur: Gegeben sei eine endliche Menge $C$, in der ein einzelnes ausgezeichnetes Element (geheimer Code) mittels Anfragen zu finden ist. In jeder Spielsituation gibt es eine endliche Zahl möglicher Anfragen, welche durch die Spielregeln festgelegt sind. Unter einer Anfrage ist dabei eine disjunkte Zerlegung von $C$ zu verstehen, wobei man als Antwort diejenige Teilmenge von $C$ erhält, in der das gesuchte Element liegt.

Durch sukzessive Anfragen kann diese Teilmenge immer weiter eingeschränkt werden, bis das gesuchte Element eindeutig identifiziert ist. Ziel ist es, möglichst schnell ans Ziel zu kommen. Je nach Spiel kann „möglichst schnell“ verschiedene Bedeutungen haben, beispielsweise:

- Die Suche soll im Durchschnitt möglichst wenige Anfrage benötigen.
- Die Suche soll (mit größtmöglicher Wahrscheinlichkeit) mit einer vorgegebenen Maximalzahl von Anfragen zum Ziel gelangen.
- Bei Mehrpersonenspielen: Jeder Spieler möchte mit größtmöglicher Wahrscheinlichkeit schneller als seine Gegner sein.

Weitere Optimierungskriterien sind denkbar. Wir wollen uns hier nur auf das erste Kriterium, d.h. die durchschnittliche Zahl der Anfragen, beschränken, wobei wir von der Gleichwahrscheinlichkeit aller Geheimcodes aus $C$ ausgehen.

Um Deduktionsspiele als Suchprobleme zu formalisieren, führen wir folgende Bezeichnungen ein:

| | |
|---|---|
| $C$ | Menge aller Codes (welche in der aktuellen Spielsituation noch möglich sind). |
| $Q$ | Menge aller möglichen Anfragen. Je nach Spielregeln kann $Q$ von der Spielsituation (d. h. von den vorangegangenen Anfragen und Antworten) abhängen. |
| $a_q$ | Anzahl aller möglichen Antworten der Anfrage $q \in Q$. Die Antwortmöglichkeiten $a$ seien dabei willkürlich von 1 bis $a_q$ durchnummeriert. |
| $q \in Q$ | Anfrage der Form $q = (q_1, q_2, \ldots, q_{a_q})$. Dabei seien $q_i : \mathrm{Pot}(C) \to \mathrm{Pot}(C)$ Filter, die genau diejenigen Elemente zurückliefern, die zur Antwort $i$ passen, d. h.: $q_i(C) = \{c \in C \mid \text{Anfrage } q \text{ hat Antwort } i\text{, falls } c \text{ der geheime Code ist.}\}$ |

Für jede Anfrage $q$ bilden die Mengen $(q_i(C))_{i=1}^{a_q}$ eine disjunkte Zerlegung der Menge $C$. Die Elemente von $q_i(C)$ lassen sich durch die Anfrage $q$ nicht unterscheiden. Da wir von der Gleichwahrscheinlichkeit aller geheimen Codes ausgehen, ist auch jeder Code aus $q_i(C)$ mit der gleichen Wahrscheinlichkeit der geheime Code, falls auf die Anfrage $q$ die Antwort $i$ gegeben wird. Die Wahrscheinlichkeit, die Antwort $i$ zu erhalten ist proportional zur Zahl der Elemente in $q_i(C)$. Es sei also

| | |
|---|---|
| $p_i(C,q) = \frac{\lvert q_i(C)\rvert}{\lvert C\rvert}$ | die Wahrscheinlichkeit, die Antwort $i$ zu erhalten und |
| $p(C,q) = (p_i(C,q))_{i=1}^{a_q}$ | der Vektor der Antwortwahrscheinlichkeiten. |

Oft dient es der Übersichtlichkeit, wenn wir die Argumente $C$ und $q$ weglassen, wir schreiben also $p = (p_1, \ldots, p_a)$ und $q = (q_1, \ldots, q_a)$. Dabei identifizieren wir die Filter $q_i$ mit den Mengen $q_i(C)$, d. h. $q$ steht auch für eine Partition von $C$.

### 4.2.3 Perfekte Strategien

Zur Ermittlung perfekter Strategien für Suchprobleme kann man rekursiv vorgehen, da sich durch eine sinnvolle Anfrage die Menge der verbleibenden Codes stets reduziert. Wie gehen hier davon aus, dass es stets mindestens eine solche Anfrage gibt[1].

**Satz 4.2.** *Sei $M \subseteq C$ eine beliebige nichtleere Teilmenge der Codemenge $C$ eines Suchspiels. Dann gilt für den Erwartungswert $\langle M\rangle$ der Suchdauer bei perfekter Suchstrategie die Rekursionsformel*

$$\langle M\rangle = \begin{cases} 0 & \text{falls } |M| = 1 \\ 1 + \min\limits_{q\in Q^*(M)} \sum\limits_{i=1}^{a_q} p_i(M,q)\langle q_i(M)\rangle & \text{falls } |M| > 1\,. \end{cases}$$

*Dabei bezeichnet $Q^*(M)$ die Menge sinnvoller Anfragen, d. h.*

$$Q^*(M) = \{q \in Q \mid |q_i(M)| < |M| \text{ für alle } i = 1, \ldots, a_q\}\,.$$

**Beweis**. Die Menge $Q^*(M)$ gewährleistet, dass $q(M)$ eine echte Partition der Menge $M$ ist, also die verbleibenden Codes $q_i(M)$ bei jeder Antwort $i$ weniger werden.

Für $|M| = 1$ ist der Geheimcode bekannt. Für $|M| > 1$ machen wir eine Anfrage. Die Anfrage wird so gewählt, dass im Erwartungswert für die verbleibenden Codes möglichst wenige Anfragen benötigt werden. $p_i(M,q)$ ist dabei die Wahrscheinlichkeit, dass bei Anfrage $q$ die Antwort $i$ gegeben wird. □

Bei der Anwendung der Rekursionsformel auf einem Computer stößt man aber recht schnell an Grenzen, da der Rechenaufwand exponentiell mit der Suchtiefe wächst.[2]

Um sich die Suche etwas zu erleichtern werden Branch-and-Bound Techniken verwendet: Mit lokalen Heuristiken (siehe Abschnitt 4.2.4) lassen sich schnell gute Anfragen finden. Die Auswertung der entsprechenden Teilbäume liefert gute ober Schranken für die perfekte Strategie. Umgekehrt lassen sich durch Abschätzungen wie in Abschnitt 4.3.2 untere Schranken finden, mit denen man unwichtige Teilbäume frühzeitig verwerfen kann.

[1] Beim Spiel Beams ist dies nicht immer der Fall – das Spiel kann mit einer Codemenge $|M|$ mit $|M| > 1$ enden. Durch die Vergabe von Strafpunkten (s. Abschnitt 4.4.1) ist aber eine, mit dem folgenden Satz kompatible Bewertung der Restmengen gegeben.

[2] Bei Mastermind kommt hinzu, dass die Anfragemenge $Q$ sehr groß ist. Deshalb wurde die perfekte Strategie für Mastermind erst 1993 publiziert [33].

### Freie Suchprobleme

**Definition 4.3** (Freies Suchproblem). *Ein freies Suchproblem mit Anfragebreite $a$ ist ein Suchproblem, in welchem alle Partitionen mit maximal $a$ ($a > 1$) Komponenten als Anfrage zulässig sind.*

Außer der Einschränkung, dass maximal $a$ verschiedene Antworten möglich sind, gibt es für die Anfragen bei freien Suchproblemen keine weiteren strukturellen Einschränkungen. D. h. jede Partition mit maximal $a$ Komponenten stellt eine zulässige Anfrage dar, bei deren Beantwortung diejenige Komponente ausgewählt wird, die den gesuchten Code enthält. Statt „freies Suchproblem mit Anfragebreite $a$“ schreiben wir aus Lesbarkeitsgründen auch kurz „freies Suchproblem“.

Man kann für freie Suchprobleme leicht die folgende Abschätzung zeigen.

**Satz 4.4.** *Sei $a \in \mathbb{N}_{>1}$. Sei $M \subseteq C$ eine beliebige nichtleere Teilmenge der Codemenge $C$ eines Suchspiels. Dann gilt für den Erwartungswert $\langle M \rangle$ der Suchdauer bei perfekter Suchstrategie mit in der Breite durch $a$ beschränkten Anfragen die Abschätzung*

$$\log_a |M| \leq \langle M \rangle < 1 + \log_a |M| \,.$$

**Beweis**. Für die untere Abschätzung gehen wir induktiv unter Benutzung der Rekursionsformel aus Satz 4.2 vor. Für $|M| = 1$ gilt die Aussage offenbar. Sei nun $|M| > 1$ und die Behauptung gelte für alle Mengen mit weniger als $|M|$ Elementen. Dann gilt für alle echten Partitionen $q$ der Menge $M$:

$$\begin{aligned}
\langle M \rangle &\geq 1 + \sum_{i=1}^{a} \frac{|q_i(M)|}{|M|} \langle q_i(M) \rangle \\
&\geq 1 + \sum_{i=1}^{a} \frac{|q_i(M)|}{|M|} \log_a |q_i(M)| && \text{(Induktionsvoraussetzung)} \\
&= 1 + \frac{a}{|M|} \sum_{i=1}^{a} \frac{|q_i(M)| \log_a |q_i(M)|}{a} \\
&\geq 1 + \frac{a}{|M|} \left( \sum_{i=1}^{a} \frac{|q_i(M)|}{a} \right) \log_a \left( \sum_{i=1}^{a} \frac{|q_i(M)|}{a} \right) && \text{(Jensensche Ungleichung)} \\
&= 1 + \frac{a}{|M|} \cdot \frac{|M|}{a} (\log_a |M| - \log_a a) \\
&= \log_a |M| \,,
\end{aligned}$$

wobei wir in der letzten Abschätzung die Jensensche Ungleichung für die konvexe Funktion $f(x) = x \cdot \log_a x$ benutzt haben.

Für die obere Abschätzung genügt folgende Überlegung: Wir füllen die Menge $M$ mit weiteren Codes bis zu einer Menge $M_a$ auf, deren Mächtigkeit $|M_a|$ die kleinste

Potenz von $a$ mit $|M_a| \geq |M|$ ist. Für diese gilt $|M_a| < a|M|$ und $\langle M_a \rangle = \log_a |M_a|$ und damit

$$\langle M \rangle \leq \langle M_a \rangle = \log_a |M_a| < \log_a(a|M|) = 1 + \log_a |M| \,.$$

□

Die untere Schranke gilt natürlich auch für beliebige Suchprobleme mit durch $a$ beschränkter Anfragebreite. Für freie Suchprobleme wird diese Schranke angenommen, falls $|M| = a^k$ für ein $k \in \mathbb{N}$ gilt, weil man stets eine gleichmäßige Partition bilden kann. In den anderen Fällen gilt stets $\langle M \rangle > \log_a |M|$, da nicht immer gleichmäßige Partitionen möglich sind und die Funktion $f(x) = x \cdot \log_a x$ streng konvex ist.

Um einen Eindruck davon zu bekommen wie groß die Abweichung für $|M| \neq a^k$ ist, müssen wir zunächst den Erwartungswert perfekter Strategien bestimmen. Der Weg über die Rekursionsformel aus Satz 4.2 ist sehr mühselig, da man jeweils alle (monotonen) Partitionen der Codemenge analysieren müsste. Den Aufwand kann man erheblich reduzieren, wenn man ausnutzt, dass nur die Anzahl der Elemente in jedem Teil der Partition relevant ist. Das Erzeugen der Partitionen kann dann ebenfalls rekursiv erfolgen. Möchte man eine Menge mit $m$ Elementen in $j$ Teile partitionieren, gibt es immer eine Komponente mit $m_1 \leq \lfloor \frac{m}{j} \rfloor$ Elementen. Der Rest ist eine Partition einer Menge mit $m - m_1$ Elementen in $j - 1$ Teile.

Diese Restmenge behandeln wir analog, müssen uns aber merken, in wie viele Teile (hier $j-1$) wir die Menge noch aufteilen dürfen. Diese Restriktion gilt aber nur für die aktuelle Anfrage, für spätere Anfragen steht wieder die volle Breite $a$ der Anfragen zur Verfügung.

Aus diesen Überlegungen heraus ergibt sich direkt eine Rekursionsformel für die erwarteten Suchdauern.

**Satz 4.5.** *Sei $a \in \mathbb{N}_{>1}$. Bezeichne $\langle m, j \rangle$ die Suchdauer für eine Menge mit $m \leq 1$ Elementen bei perfekter Suchstrategie, wobei die erste Suchanfrage die Breite $j$ mit $2 \leq j \leq a$ und alle weiteren Anfragen die Breite $a$ haben. Dann gilt*

$$\langle m, j \rangle = \begin{cases} 0 & \text{falls } m = 1 \\ 1 & \text{falls } 1 < m \leq j \\ \min\limits_{1 \leq m_1 \leq \lfloor \frac{m}{j} \rfloor} \left( \frac{m_1}{m}(1 + \langle m_1, a \rangle) + \frac{m - m_1}{m}(1 + \langle m - m_1, a \rangle) \right) & \text{f. } m > j,\ j = 2 \\ \min\limits_{1 \leq m_1 \leq \lfloor \frac{m}{j} \rfloor} \left( \frac{m_1}{m}(1 + \langle m_1, a \rangle) + \frac{m - m_1}{m} \langle m - m_1, j - 1 \rangle \right) & \text{f. } m > j,\ j > 2\,. \end{cases}$$

**Beweis**. Im Fall $m = 1$ ist die Lösung eindeutig und im Fall $1 < m \leq j$ kann man die $m$ Elemente mit einer Anfrage in $j$ Teile aufspalten, wobei wir in diesem Fall leere Komponenten zulassen.

Für $j = 2$ teilen wir die Menge in zwei Teile auf mit $m_1$ bzw. $m - m_1$ Elementen. Da wir nicht weiter aufspalten dürfen sind diese Teile Komponenten der tatsächlichen

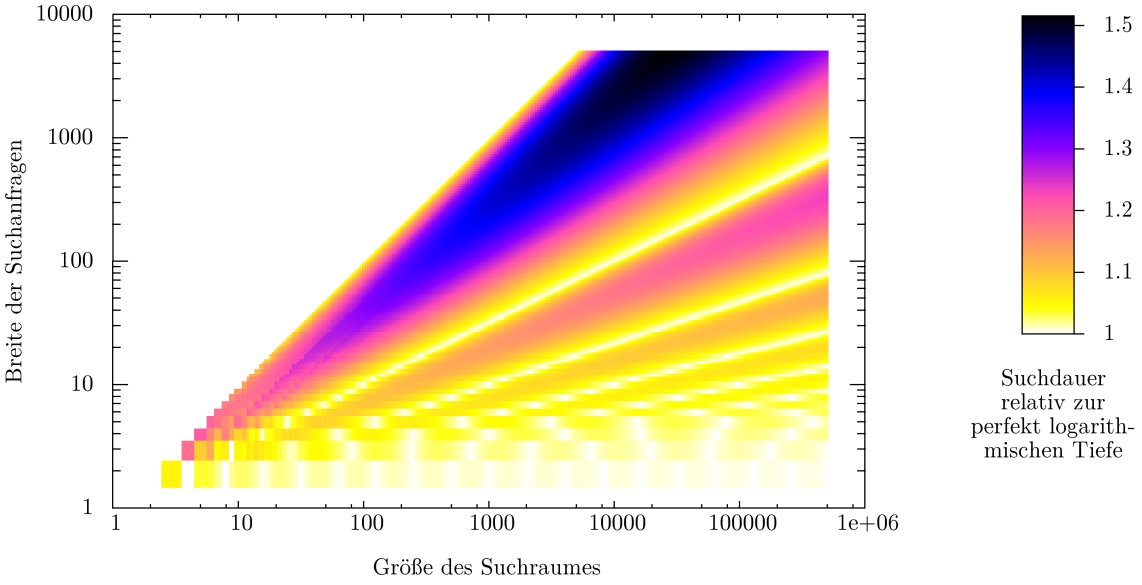



**Abbildung 4.1:** Suchtiefe für freie Suchprobleme mit $m$ Elementen und Anfragebreite $a$ relativ zur theoretischen Grenze $\log_a(m)$. In Abhängigkeit von $m$ und $a$ ist die Suchtiefe als Farbwert dargestellt. Es sind nur Werte für $a \leq m$ eingezeichnet.

Anfrage, d. h. der Erwartungswert hat die Struktur $1 + \langle m_1, a\rangle$ multipliziert mit der Eintrittswahrscheinlichkeit $\frac{m_1}{m}$. Analog für $m - m_1$.

Für $j > 2$ können wir die Komponente $m - m_1$ weiter aufspalten bevor wir die Anfrage tatsächlich ausführen. Der Erwartungswert hat die Struktur $\langle m - m_1, j - 1\rangle$ multipliziert mit der gesamten Eintrittswahrscheinlichkeit $\frac{m-m_1}{m}$. □

Die Abbildung 4.1 zeigt die relative Zunahme der Suchdauer bei perfekter Strategie im Vergleich zur theoretischen Grenze $\log_a(m)$. In der doppelt-logarithmischen Darstellung sind die Potenzen $|M| = a^k$ als helle Linien zu erkennen. Für feste Breite $a$ der Suchanfragen ist nach Satz 4.4 der absolute Unterschied durch 1 beschränkt. Entsprechend fällt der (maximale) relative Unterschied mit steigender Suchraumgröße wie $(\log_a(m))^{-1}$ und konvergiert somit gegen Null, jedoch sehr langsam.

Im Spezialfall der binären Suche, d. h. $a = 2$, entfällt in der Rekursionsformel der dritte Fall. Dadurch ist es möglich, eine explizite Formel für diesen Fall anzugeben ([2, Kap. III, §7]):

$$\langle m\rangle = \lceil \log_2 m\rceil - \frac{1}{m} 2^{\lceil \log_2 m\rceil} + 1\,.$$

### 4.2.4 Maximierung der Rényi-Entropie als lokale Heuristik

Die Maximierung der (Rényi-)Entropie der Antwortwahrscheinlichkeiten ist ein gebräuchliches Verfahren, gute Agenten für Deduktionsspiele zu entwickeln. Ein solcher

Agent denkt immer genau einen Schritt voraus, indem er für alle in der aktuellen Situation erlaubten Anfragen die Rényi-Entropie $H_\alpha$ der Antwort-Wahrscheinlichkeiten berechnet. Unter den Anfragen mit maximaler Rényi-Entropie wird ggf. unter Hinzunahme einer Tie-Break-Regel die Anfrage für den nächsten Zug ausgewählt. Eine solche lokale Heuristik nennen wir Rényi-Entropie-Strategie bzw. kurz $H_\alpha$-Strategie.

Tabelle 4.2 zeigt den Zusammenhang zwischen einigen klassischen lokalen Suchstrategien und den $H_\alpha$-Strategien der Ordnung $\alpha \in \{0, 1, 2, \infty\}$.

| Maximiere | Anschauliche Interpretation im Sinne des Suchprozess |
|---|---|
| $H_0(p)$ | Maximiere die Anzahl der möglichen Antworten (most-parts) |
| $H_1(p)$ | Maximiere die Shannon-Entropie |
| $H_2(p)$ | Minimiere die erwartete Anzahl verbleibender Codes, denn: $H_2(p) = -\log_2\left(\sum_{i=1}^n p_i^2\right) = -\log_2\left(\frac{1}{\lvert C\rvert} \cdot \sum_{i=1}^n p_i \cdot \lvert q_i\rvert\right)$ |
| $H_\infty(p)$ | Minimiere die maximale Anzahl verbleibender Codes |

**Tabelle 4.2:** Beispiele klassischer Strategien für Suchprobleme.

Es sei darauf hingewiesen, dass die exakte Berechnung der Entropien die Kenntnis der Antwort-Wahrscheinlichkeiten und somit der Code-Mengen voraussetzt. Dies kann im Falle vieler Codes und/oder vieler Anfragemöglichkeiten mit erheblichem Rechenaufwand verbunden sein.

### 4.2.5 Die besondere Rolle der Shannon-Entropie

Wir werden uns in diesem Abschnitt genauer der Shannon-Entropie zuwenden und zeigen, warum sie, unter gewissen Anforderungen an das Deduktionsspiel, sehr gute Heuristiken liefert. Die Schwierigkeit konkreter Abschätzungen liegt darin begründet, dass die Spielregeln die Menge der erlaubten Anfragen stark einschränken können. Außerdem kann es komplexe Abhängigkeiten der Anfragen während des Spielverlaufs geben, z. B. dadurch dass man bei Beams jede Diagonale nur einmal abfragen darf. Eine lokale Heuristik kann dies natürlich nicht vollständig erfassen.

Betrachten wir zunächst den Grenzfall, dass es außer der Breite keine weiteren Einschränkungen bei den Anfragen gibt, d. h. jede disjunkte Zerlegung der Codemenge in maximal $a$ Teile entspricht einer zulässigen Anfrage. In diesem Falle (also im Fall eines freien Suchproblems) zeigt sich, dass alle Heuristiken, die auf der Maximierung der Rényi-Entropie mit $0 < \alpha < \infty$ beruhen, dieselben Anfragen vorschlagen. Das sind genau diejenigen Anfragen, die zu einer möglichst gleichmäßigen Partition der Codemenge $C$ führen, wie der folgende Satz zeigt.

**Satz 4.6.** *Seien $a, m \in \mathbb{N}_{>0}$, $x \in \mathbb{N}^a$ und $0 < \alpha < \infty$. Ist $H_\alpha(x)$ maximal unter der Nebenbedingung $\sum_{i=1}^a x_i = m$, dann gilt $|x_i - x_j| \leq 1$ für alle Komponenten von $x$.*

**Beweis**. Es sei $x^* = \frac{m}{a}$ die durchschnittliche Größe der Elemente von $x$. Wir zeigen, dass für alle Komponenten von $x$ gilt $|x_i - x^*| < 1$, denn daraus ergibt sich die Behauptung.

Wir gehen indirekt vor. Zunächst untersuchen wir den Fall einer Abweichung nach oben, d. h. es gibt ein $i \in \{1, \ldots, a\}$ mit $x_i \geq x^* + 1$. Da $x_i$ insbesondere größer als der Durchschnitt $x^*$ ist, muss es auch ein $j \in \{1, \ldots, a\}$ mit $x_j < x^*$ geben. Weiterhin führen wir die positive ganze Zahl $\delta = x_i - 1 - x_j$ ein.

Sei $f : \mathbb{N} \to \mathbb{N}$ eine streng konvexe Funktion. Dann gilt

$$f(n + \delta + 1) - f(n + \delta) > f(n + 1) - f(n) \quad \text{für alle } n \in \mathbb{N}\,.$$

Wir setzen nun $n = x_j + 1$, d. h.

$$f(x_i) - f(x_i - 1) > f(x_j + 1) - f(x_j)\,.$$

Für $\alpha > 1$ setzen wir $f(n) = n^\alpha$, es ergibt sich

$$(x_i)^\alpha + (x_j)^\alpha > (x_j + 1)^\alpha + (x_i - 1)^\alpha\,.$$

Ersetzen wir also im Vektor $x$ die Elemente $x_i$ und $x_j$ durch $x_i - 1$ und $x_j + 1$, vergrößert sich der Wert von $H_\alpha(x)$. Dabei ist zu beachten dass in

$$H_\alpha(x) = \frac{1}{1-\alpha} \log_2 \left( \sum_{i=1}^{a} x_i^\alpha \right)$$

der Logarithmus streng monoton wachsend und der Vorfaktor für $\alpha > 1$ negativ ist. D. h. $x$ war bzgl. $H_\alpha$, im Widerspruch zur Annahme, nicht maximal.

Für $0 < \alpha < 1$ ist $f(n) = n^\alpha$ konkav und es drehen sich die Relationszeichen um, was der diesmal positive Vorfaktor kompensiert. Im Fall $\alpha = 1$ galt

$$H_1(x) = -\sum_{i=1}^{a} x_i \log_2 x_i\,,$$

wobei hier die Funktion $f(n) = -n \log(n)$ ebenfalls streng konkav ist.

Im Falle einer Abweichung nach unten, d. h. es gibt ein $i \in \{1, \ldots, a\}$ mit $x_i \leq x^* - 1$, geht man ganz analog vor. □

Für freie Suchprobleme haben damit alle Partitionen mit maximaler Rényi-Entropie mit $0 < \alpha < \infty$ dieselbe Struktur und sind gleichwertig, da es nur auf die Anzahl der Codes in den Teilmengen ankommt. Bezüglich $H_0$ und $H_\infty$ können noch weitere Partitionen optimal sein.

Die Vergrößerung der erwarteten Suchdauer zwischen diesen gleichwertigen $H_\alpha$-Strategien mit $0 < \alpha < \infty$ und der perfekten Strategie sind in Abbildung 4.3 dargestellt. Für Anfragebreiten $a > 2$ haben wir bei Suchraumgrößen $m \neq a^k$ meist Verluste gegenüber perfekter Suche. Die empirischen Daten zeigen, dass dieser Effekt in der

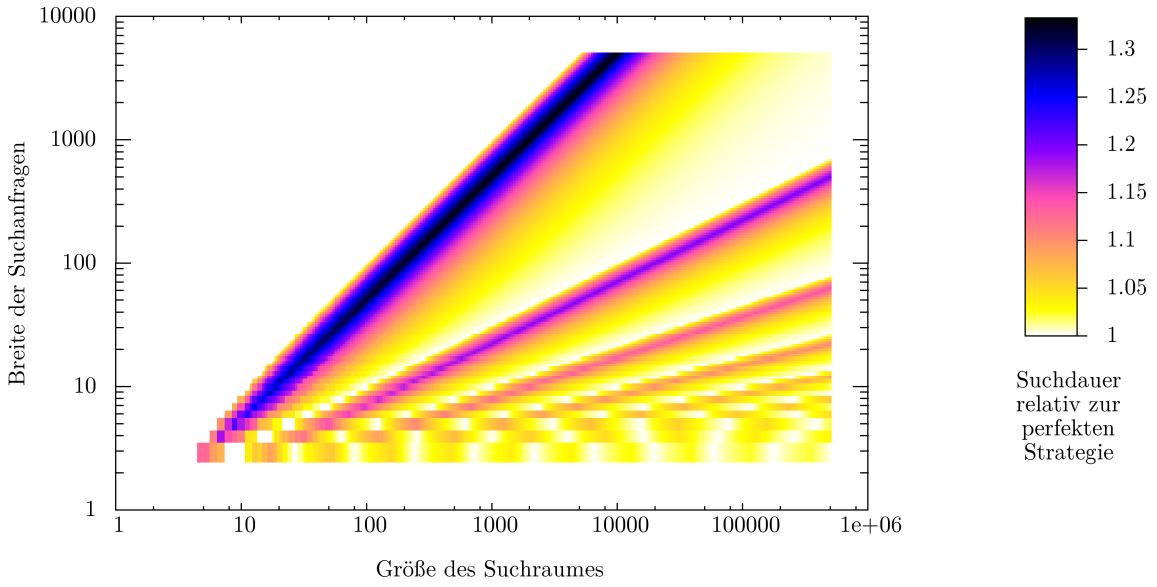



**Abbildung 4.3:** Suchtiefe für freie Suchprobleme mit $m$ Elementen und Anfragebreite $a$ für die $H_\alpha$-Strategie relativ zur perfekten Strategie.

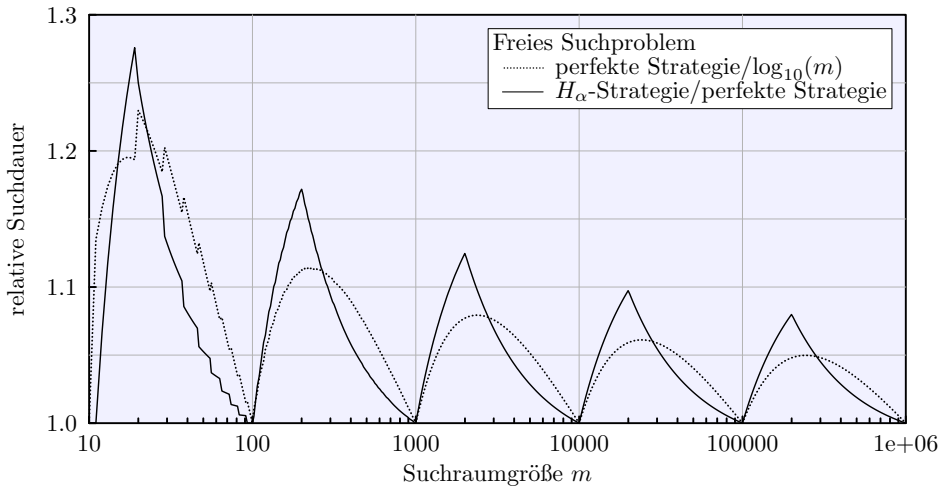



**Abbildung 4.4:** Freies Suchproblem mit Anfragebreite $a = 10$: Suchdauern für $H_\alpha$-Strategie, perfekte Strategie und logarithmische Grenze im Vergleich.

gleichen Größenordnung wie der Diskretisierungseffekt[3] liegt. Ebenfalls scheint dieser Unterschied mit größer werdender Suchraumgröße bei fester Breite $a$ der Suchanfragen gegen Null zu konvergieren. Abbildung 4.4 zeigt beide Effekte im Vergleich für eine feste Anfragebreite $a = 10$.

Im Falle eingeschränkter Suchprobleme, können sich die $H_\alpha$-Strategien mit $0 < \alpha < \infty$ unterscheiden. Bei konkreten Deduktionsspielen (vgl. Zahlenmaterial in den folgenden Abschnitten) zeichnet sich ab, dass $\alpha \approx 1$ in der Regel zu den besten lokalen Strategien führt. Der folgende Satz zeigt, dass für nicht zu chaotische Spiele mit logarithmischer Spieldauer die $H_1$-Heuristik lokal kaum schlechter ist, als eine perfekte Strategie.

**Satz 4.7.** *Wir betrachten ein Deduktionsspiel mit Codemenge $C$ und*

$$\gamma \log_2 |M| \leq \langle M \rangle \leq \Gamma \log_2 |M| \quad \text{für alle nicht-leeren } M \subseteq C\,,$$

*wobei $\langle M \rangle$ der Erwartungswert der Spieldauer bei perfektem Spiel mit Ausgangscodemenge $M$ ist. Dann vergrößert sich die Spieldauer bei einmaligem Anwenden der $H_1$-Strategie gegenüber perfekter Strategie maximal um den Faktor $\frac{\Gamma}{\gamma}$.*

**Beweis**. Sei $q$ eine perfekte Anfrage und $\tilde{q}$ die bzgl. $H_1$ optimale Anfrage. Außerdem bezeichne $\langle M, \tilde{q} \rangle$ die erwartete Spieldauer wenn zuerst $\tilde{q}$ gewählt und anschließend perfekt gespielt wird. Dann gilt

$$\begin{aligned}
\langle M, \tilde{q} \rangle = 1 + \sum_{i=1}^{a} \tilde{p}_i \langle \tilde{q}_i \rangle &= 1 + \sum_{i=1}^{a} \frac{|\tilde{q}_i|}{|M|} \langle \tilde{q}_i \rangle \\
&\leq 1 + \sum_{i=1}^{a} \frac{|\tilde{q}_i|}{|M|} \Gamma \log_2 |\tilde{q}_i| \\
&\leq 1 + \sum_{i=1}^{a} \frac{|q_i|}{|M|} \Gamma \log_2 |q_i| \qquad \text{(da } \tilde{q}_i \text{ optimal bzgl. } H_1\text{)} \\
&\leq 1 + \sum_{i=1}^{a} \frac{|q_i|}{|M|} \cdot \frac{\Gamma}{\gamma} \langle q_i \rangle \\
&\leq \frac{\Gamma}{\gamma} \langle M, q \rangle = \frac{\Gamma}{\gamma} \langle M \rangle\,.
\end{aligned}$$

□

Für reale Spiele ist dieser einfache Satz nicht sehr aussagekräftig, da ein Faktor $\frac{\Gamma}{\gamma}$ pro Anfrage ziemlich schlecht sein kann. Je dichter jedoch die beiden Abschätzungskonstanten $\gamma$ und $\Gamma$ beieinander liegen, desto perfekter wird die $H_1$-Strategie. Dabei ist zu beachten, dass nicht vorausgesetzt wurde, dass sich unter den Anfragen näherungsweise gleichmäßige Partitionen befinden müssen. Mit anderen Worten, das Prinzip der Entropie-Maximierung ist auch fernab vom Gleichgewicht anwendbar.

[3] Relativer Unterschied zwischen optimaler Suche und $\log_a m$, vgl. Abbildung 4.1

Für Spiele, die sich auf beliebig große Code-Mengen ausdehnen lassen, kann man versuchen, die Bedingung $\gamma \log_2 |M| \leq \langle M \rangle \leq \Gamma \log_2 |M|$ asymptotisch oder stochastisch zu interpretieren. Wir wollen dies im Rahmen der Arbeit nicht weiter vertiefen, jedoch im Abschnitt 4.5.1 zeigen, dass andere Asymptotiken dazu führen können, dass $H_1$ asymptotisch nicht die beste Rényi-Entropie-Strategie sein muss.

## 4.3 Mastermind

Mastermind ist eines der bekanntesten Deduktionsspiele weltweit. Es wurde 1970 von Mordechai Meirovitz erfunden und ab 1971 von Invicta Toys and Games Ltd. vermarktet. Jedoch gibt es zahlreiche Vorgänger und Varianten, beispielsweise „Bulls and Cows“ (mit Codes aus verschiedenen Ziffern von 0 bis 9) und Jotto (mit Wörtern als Codes). Jotto wurde von 1955 von Morton M. Rosenfeld erfunden und zunächst von seiner Firma Jotto Corp. vermarktet.

### 4.3.1 Spielregeln von Mastermind

Mastermind ist ein Zwei-Personen-Deduktionsspiel. Der erste Spieler (Codemaker) wählt einen Code, bestehend aus vier Stiften, wobei jeder Stift eine von sechs Farben hat. Die Reihenfolge ist relevant und mehrere gleichfarbige Stifte sind erlaubt. Der zweite Spieler (Codebreaker) versucht den Code zu knacken, indem er solange Codes vorschlägt bis der richtige Code getroffen wird. Jeder Vorschlag wird vom ersten Spieler mit zwei Zahlen (üblicherweise schwarze und weiße Marker) bewertet: Die erste Zahl gibt die Anzahl der korrekten Stifte (Farbe und Position), die zweite Zahl die Stifte mit richtiger Farbe, aber falscher Position an. Dabei wird jeder Stift nur einmal gezählt, d. h. der Vorschlag „AABB“ hat die Antwort $(1, 2)$, wenn der Geheimcode „ABAA“ lautet. Das Spiel endet mit dem korrekten Vorschlag, also der Bewertung $(4, 0)$ und der zweite Spieler (Codemaker) bekommt für jeden benötigten Versuch des ersten Spielers einen Punkt. Im Anschluss wird das Spiel mit vertauschten Rollen wiederholt.

### 4.3.2 Eigenschaften und Varianten von Mastermind

Insgesamt gibt es $6^4 = 1296$ verschiedene Codes. Jeder von ihnen ist gleichzeitig eine Anfrage im Sinne der oben eingeführten Suchprobleme. Es gibt 14 mögliche Antworten: $((0,0)$, $(0,1)$, $(0,2)$, $(0,3)$, $(0,4)$, $(1,0)$, $(1,1)$, $(1,2)$, $(1,3)$, $(2,0)$, $(2,1)$, $(2,2)$, $(3,0)$, $(4,0))$. Wegen $14^2 < 1296$ benötigt man (abgesehen von Zufallstreffern) mindestens drei Versuche um den Code zu identifizieren und (als Besonderheit bei Mastermind) eine vierte Anfrage in der der korrekte Code gelegt werden muss. Daraus ergibt sich eine einfache unter Schranke für durchschnittliche Zahl der benötigten Anfragen zu

$$\frac{1 \cdot 1 + 13 \cdot 2 + 13^2 \cdot 3 + (1296 - 1 - 13 - 13^2) \cdot 4}{1296} = \frac{4986}{1296} \approx 3.8472\,.$$

Tatsächlich sind die Antworten nicht gleichwahrscheinlich, so dass z. T. fünf oder mehr Anfragen notwendig sind[4], im Schnitt sind es bei perfektem Spiel $\frac{5625}{1296} \approx 4.3403$.

Diese (Standard-)Version von Mastermind sowie die oben eingeführten klassischen lokalen Suchstrategien wurden ausführlich in der Literatur diskutiert ([31], [29], [13], [32]). Einige dieser Ergebnisse wollen wir im Folgenden kurz zusammenstellen. Alle vorgestellten Resultate wurden im Rahmen dieser Arbeit verifiziert.

Bevor wir damit beginnen, wollen wir noch eine Variante von Mastermind einführen, um zusätzliche Vergleichsergebnisse zu bekommen. In dieser (Junior-)Variante sind nur verschiedenfarbige Codes (als Geheimcode und Anfragen) erlaubt. Es gibt damit $\binom{6}{4} \cdot 4! = 360$ Codes. Wir nennen diese Variante Mastermind-360 und das Original entsprechend Mastermind-1296. In Mastermind-360 benötigt die perfekte Strategie im Schnitt $\frac{1446}{360} \approx 4.0167$ Versuche. Eine unter Schranke ist analog zu oben durch $(1 \cdot 1 + 13 \cdot 2 + 13^2 \cdot 3 + (360 - 1 - 13 - 13^2) \cdot 4)/360 = \frac{1242}{360} = 3.45$ gegeben.

Anmerkung zur Bestimmung der perfekten Strategie: Bei Mastermind reicht es nicht aus, den Geheimcode zu identifizieren, sondern er muss auch gelegt werden. Bei der Benutzung der Rekursionsformel aus Satz 4.2 ist zu beachten dass im Falle einer von $(4, 0)$ verschiedenen Bewertung für verbleibende Einermengen $\langle M \rangle = 1$ gilt.

### 4.3.3 Analyse von Mastermind

In Tabelle 4.5 vergleichen wir die erwarteten Spieldauern für die $H_\alpha$-Strategien mit $\alpha \in \{0, 1, 2, \infty\}$ für beide Mastermind-Varianten. Im Falle gleicher Entropie-Werte für verschiedene Anfragen benötigen wir eine Tie-Break-Regel um eine dieser Anfragen auszuwählen. Die Regel LEX nimmt den ersten dieser Codes in lexikographischer Reihenfolge, während MIX zufällig mit gleichen Wahrscheinlichkeiten einen dieser Codes auswählt. In der Tabelle gibt „gesamt“ die Summe der Anfragen an, um alle Codes zu knacken. Im Falle der deterministischen Strategie LEX ist dies stets eine ganze Zahl. Im Falle von MIX ergeben sich Mittelwerte, welche durch vollständige Analyse exakt bestimmt wurden. Bei $H_1$ und $H_2$ ergeben sich (im Gegensatz zu $H_0$ und $H_\infty$) aufgrund der recht seltenen Tie-Breaks noch überschaubare Brüche. Als „mittel“ ist die mittlere Spieldauer, also „gesamt“ geteilt durch die Anzahl der Codes angegeben. „$\Delta$“ ist die relative Zunahme der Spieldauer im Vergleich zur perfekten Strategie.

Die Verwendung der Rényi-Entropie erlaubt die Betrachtung weitere intermediärer Strategien. Deren mittlere Spieldauern sind graphisch in den Diagrammen der Abbildung 4.6 dargestellt. Wie erwartet gibt es in der Nähe von $\alpha = 1$ Minima. Man beachte außerdem, dass die Funktionen nicht stetig sind. Da die Entscheidungsprozesse stets auf diskreten Mengen ablaufen, handelt es sich um stückweise konstante Funktionen. Dass die Funktionen im Limes $\alpha \to 0$ nicht stetig sind lässt sich so erklären: Zwei

[4] Donald E. Knuth hat 1976 eine Strategie vorgestellt, die im Worst-Case mit fünf Anfragen auskommt [31], sie ist aber im Erwartungswert nicht optimal.

| Strategie | | Standard Mastermind-1296 | | Vereinfachtes Mastermind-360 | |
|---|---|---|---|---|---|
| perfekt | gesamt | 5625 | | 1446 | |
| | mittel | 4.3403 | | 4.0167 | |
| | Tie-Break-Regel | LEX | MIX | LEX | MIX |
| $H_0$ | gesamt | 5668 | 5706.96 | 1457 | 1451.80 |
| | mittel | 4.3734 | 4.4035 | 4.0472 | 4.0328 |
| | $\Delta$ | 0.76% | 1.46% | 0.76% | 0.40% |
| $H_1$ | gesamt | 5722 | 5722.5 [a] | 1450 | 1450 |
| | mittel | 4.4151 | 4.4155 | 4.0278 | 4.0278 |
| | $\Delta$ | 1.72% | 1.73% | 0.28% | 0.28% |
| $H_2$ | gesamt | 5696 | 5688.75 [b] | 1459 | 1458.33 [c] |
| | mittel | 4.3951 | 4.3895 | 4.0528 | 4.0509 |
| | $\Delta$ | 1.26% | 1.13% | 0.90% | 0.85% |
| $H_\infty$ | gesamt | 5801 | 5791.42 | 1490 | 1483.67 |
| | mittel | 4.4761 | 4.4687 | 4.1389 | 4.1216 |
| | $\Delta$ | 3.13% | 2.96% | 3.04% | 2.61% |
| | Tie-Break-Regel | LEX | MIX | LEX | MIX |

[a] $= \frac{11445}{2}$ [b] $= \frac{5256409}{924}$ [c] $= \frac{4375}{3}$

**Tabelle 4.5:** $H_\alpha$-Strategien für Mastermind im Vergleich.

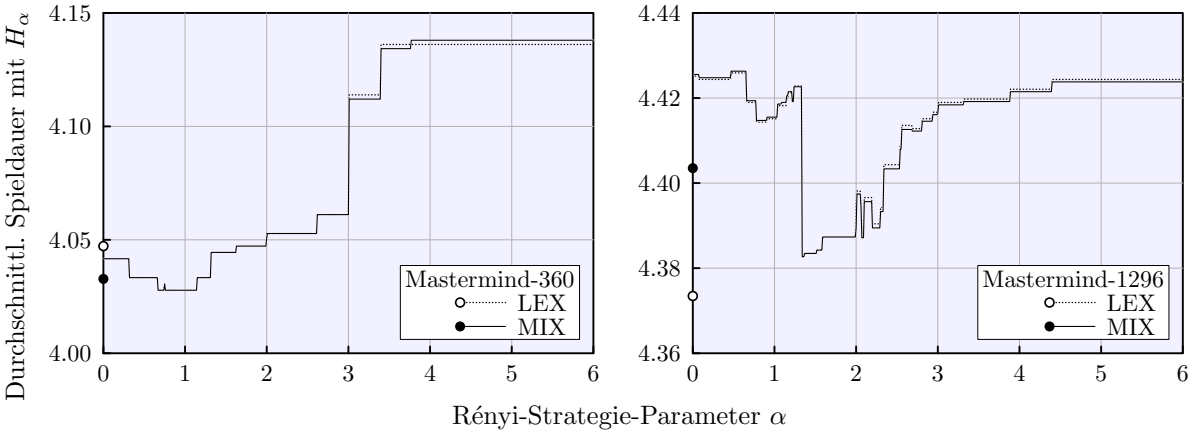



**Abbildung 4.6:** Durchschnittliche Spieldauer für Mastermind in Abhängigkeit des Rényi-Strategie-Parameters $\alpha$. Man beachte, dass diese Funktionen nicht stetig sind, insbesondere mit Sprüngen bei $\alpha = 0$.

verschiedene Vektoren mit gleicher Null-Norm (Anzahl nicht verschwindender Komponenten) haben typischerweise verschiedene $\alpha$-Norm für jedes $\alpha > 0$. Somit ist die Tie-Break-Menge für $\alpha = 0$ oft viel größer als für $\alpha > 0$.

Große Sprünge werden vor allem durch die Wahl der ersten Anfrage bestimmt. Besonders deutlich wird das bei Mastermind-1296. Für $\alpha \gtrsim 1.33789$ ist die erste Anfrage „AABC“ und für kleinere $\alpha$ ist es „ABCD“. Beide Anfragen haben denselben $H_0$-Wert, nämlich 14, so dass die LEX-Regel „AABC“ wählt. Überraschend ist, dass die $H_0$-LEX-Strategie die global beste Rényi-Entropie-Strategie ist. Dies wurde 2005 von Barteld Kooi gefunden [32], wobei nur die Strategien mit $\alpha \in \{0, 1, 2, \infty\}$ untersucht wurden.

Fragt man bei Mastermind-1296 unabhängig von den Entropiewerten im 1. Versuch stets „AABC“, so ähnelt die Kurve deutlich mehr der von Mastermind-360 und denen von Beams, wie in Abbildung 4.17 auf Seite 78 im Vergleich zu sehen ist.

## 4.4 Beams

Das Deduktionsspiel Beams[5] wurde 1995 von Michail Antonow erfunden [9]. Es handelt sich um ein Zwei-Personen-Spiel, in dem es gilt, einen vom Gegner gelegten Farbcode mit möglichst wenigen Fragen zu erraten. Während im Originalspiel beide Spieler gleichzeitig gegenseitig ihre Farbcodes entschlüsseln, wollen wir hier Beams analog zu Mastermind betrachten. D. h. ein Spieler versucht mit möglichst wenigen Fragen einen geheimen Farbcode zu knacken, während der zweite Spieler nur die Aufgabe hat, die Anfragen korrekt zu beantworten.[6] Wir interessieren uns wieder für die mittlere Anzahl benötigter Anfragen, wobei davon ausgegangen wird, dass jeder zulässige Code mit gleicher Wahrscheinlichkeit als Geheimcode auftritt.

### 4.4.1 Spielregeln von Beams

Beams wird in der Standard-Variante auf einem sechseckigen Brett mit 37 hexagonal angeordneten Feldern gespielt, wobei das Feld in der Mitte stets frei bleibt. Jedes der restlichen Felder hat eine von sechs Farben, wobei jede Farbe genau sechsmal vorkommt und jeweils gleichfarbige Felder eine zusammenhängende Gruppe (Puzzleteil) bilden müssen. Zwei Beispiele zeigt die Abbildung 4.7.

Im Spiel ist dieser Code zunächst nicht sichtbar, sondern wird durch Anfragen nach und nach aufgedeckt.

Für eine solche Anfrage wählt der Spieler eine Farbe und eine Diagonale aus. Diagonalen sind geradlinige Verbindungen zusammenhängender Felder, die von Rand zu Rand des Spielbrettes gehen. Diagonalen durch das Zentrum sind erlaubt. Es gibt $3 \cdot 7 = 21$ Diagonalen, die wir gemäß Abbildung A.1 durchnummerieren.

[5] Das Spiel „Beams“ wird unter dem Namen „Tu Ray“ die Ausstellung zum einhundertsten Geburtstag von Alan Turing im Jahre 2012 im Heinz-Nixdorf-Museums-Forum Paderborn begleiten [47].

[6] Wir gehen hier davon aus, dass bei der Beantwortung keine Fehler auftreten. Im realen Spiel können unabsichtliche oder gar absichtliche Fehler auftreten. Nach den Regeln von Antonow sind diese mit 10 Punkten (Punktvergabe siehe Spielregeln, Anhang A.3, Seite 108) zu bestrafen.

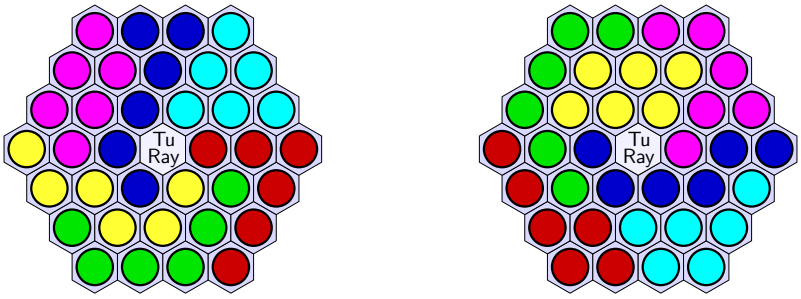



**Abbildung 4.7:** Zulässige Farbcodes.

Die gewählte Diagonale und Farbe stellen wir uns als Strahl vor (daher der Name Beams), der nur in dieser Farbe sensitiv ist. Alle getroffenen Felder dieser Farbe auf der Diagonalen werden nun aufgedeckt. Der Spieler kann nun weitere Anfragen stellen, wobei jede Diagonale höchstens einmal benutzt werden darf.

Das Spiel endet, wenn der ratende Spieler glaubt, den Code zu kennen bzw. keine ihm weiterhelfenden Diagonalen mehr frei sind. Er muss den Code vollständig legen, ggf. sind unklare Felder so auszufüllen, dass ein zulässiger Code entsteht. Für jede gebrauchte Frage und jedes Feld mit falscher Farbe gibt es jeweils einen Punkt für den Gegenspieler. Im Anschluss kann man das Spiel mit vertauschten Rollen wiederholen.

Ein ausführliches Beispiel für den Spielbeginn von Beams befindet sich im Anhang A.2 auf Seite 105.

### 4.4.2 Varianten von Beams

Analog zu Mastermind definieren wir weitere Varianten von Beams, um einerseits weiteres Vergleichsmaterial zu haben und andererseits eine hinreichend kleine Variante zu haben, deren perfekte Strategie zu ermitteln ist. Auf einem sechseckigen Brett mit Kantenlänge drei gibt es ohne das Mittelfeld 18 frei Felder. Diese füllen wir mit drei bzw. sechs Farben und Puzzleteilen aus sechs bzw. drei Feldern. Die Varianten nennen wir Beams(3c,6p) bzw. Beams(6c,3p). Dabei steht „c“ für Anzahl der Farben (color) und „p“ für die Anzahl der Felder (pieces) pro Farbe. Die Standard-Variante heißt in dieser Notation Beams(6c,6p).

Außerdem gibt es noch eine von Antonow eingeführte Experten-Version auf einem Feld mit Kantenlänge fünf. Die 60 Felder (ohne Mitte) werden mit sechs Teilen aus je zehn Feldern überdeckt. Als zusätzliche Forderung, wird verlangt, dass jede Farbe ein Randfeld hat (Randkontakt-Regel). Im Rahmen dieser Arbeit wird diese Variante, Beams(6c,10p) nicht weiter betrachtet, da die Größe der Codemenge die aktuellen Rechen- und Speicherkapazitäten weit übersteigt. Allein die Berechnung der 361 648 Puzzleteile dauerte gut einen Tag.

Die Tabelle 4.8 stellt die charakteristischen Zahlen der Beams-Varianten gegenüber. Die Zahl der Puzzleteile ist folgendermaßen zu verstehen. Das sechseckige Brett wird

| Variante | Farben | Felder | Puzzleteile | Muster | Codes | Diagonalen |
|---|---|---|---|---|---|---|
| Beams(3c,6p) | 3 | 18 | 751 | 310 | 1 860 | 15 |
| Beams(6c,3p) | 6 | 18 | 11 | 311 | 223 920 | 15 |
| Beams(6c,6p) | 6 | 36 | 814 | 4 234 462 | 3 048 812 640 | 21 |
| Beams(6c,10p) | 6 | 60 | 361 648 | viele | viele | 27 |

**Tabelle 4.8:** Beams-Varianten.

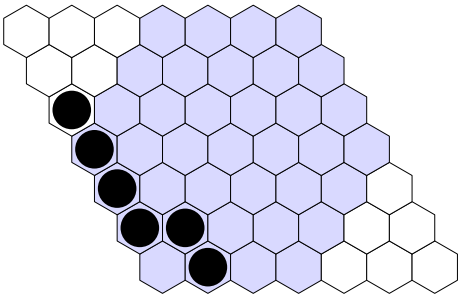


**Abbildung 4.9:** Rhombenförmig erweitertes Brett mit Puzzleteil in Normallage

in einen kleinstmöglichen Rhombus eingebettet und darin systematisch alle hineinpassenden Teile ermittelt, welche links und unten anstoßen (Normallage, siehe Abbildung 4.9). Damit werden Teile, die sich nur durch Verschiebung unterscheiden würden, nur einmal gezählt. Gedrehte bzw. gespiegelte Teile werden als unterschiedlich betrachtet und einzeln gezählt. Da sich die Rhomben in ihrer Größe unterscheiden, ist die Anzahl der Teile aus 6 Feldern bei Beams(3c,6p) und Beams(6c,6p) etwas verschieden. Einige (aber nur wenige) der Puzzleteile kommen in keinem Code vor, weil sie trotz Verschiebung nicht in das Sechseck passen oder aus sonstigen Gründen keinen zulässigen Code erlauben.

Als Muster zählen wir alle Möglichkeiten, mit den (ungefärbten) Puzzleteilen das Brett zu füllen. Auch hier werden gedrehte und gespiegelte Varianten einzeln gezählt. Die Codes entstehen durch Einfärbung, wobei es mit $c$ Farben $c!$ Möglichkeiten gibt.

Für Beams(3c,6p) welches mit 1860 Codes etwa die Komplexität von Mastermind-1296 hat, konnte die perfekte Strategie ermittelt werden. Durch verschiedene „Branch and Bound“-Techniken benötigt der Algorithmus zur Berechnung und Auswertung der perfekten Strategie nur gut eine Minute[7] und kommt zu dem Ergebnis, dass im Mittel genau $\frac{7347}{1860} = 3.95$ Anfragen benötigt werden.

[7] Ca. 75s auf einer Maschine mit Intel-E8200-CPU (2.66GHz).

### 4.4.3 Implementierung der Entropie-Heuristik für Beams(6c,6p)

#### Erzeugung der Datenbank

Ausgangspunkt der Arbeiten zu Beams war die Frage, ob es möglich ist, die Entropien für die Standard-Beams-Variante exakt zu ermitteln und die darauf basierende Strategie mit einer Vorarbeit von L. Schreiber [44] zu vergleichen, in welcher die Entropien relativ grob approximiert wurden.

Das Erzeugen der Puzzleteile (s. vorheriger Abschnitt) und Generieren aller 4 234 462 Muster war in wenigen Tagen programmiert. Die Berechnung dauerte ca. 15 Minuten. Codiert man jeweils 3 Felder in ein Byte ($6^3 = 216 < 256$) braucht man pro Code $36/3 = 12$ Byte. Das ergibt eine Datenbank von ca. 48 Megabyte.[8]

Mit den $6! = 720$ Einfärbungen ergeben sich jedoch 3 048 812 640 Codes, was eine Herausforderung darstellt. Zur schnellen Berechnung der Entropien sollten alle Codes (und beliebige Teilmengen) vollständig im RAM des Computers vorhanden sein. Das gelingt, indem man die Codes in einer Tabelle mit 4 234 462 Zeilen (für die Muster) und 720 Spalten (für die Färbungen) speichert. In jeder Zelle der Tabelle genügt dann ein Bit um anzuzeigen, ob ein Muster in der entsprechenden Färbung vorhanden ist. Die Tabelle benötigt insgesamt 3 048 812 640/8 Bytes, also ca. 360 MB, die sich im RAM heutiger Computer gut unterbringen lassen. Die Musterdatenbank (48 MB) ist natürlich ebenfalls im RAM vorzuhalten.

#### Agenten mit $H_\alpha$-Strategien

An sich ist es nicht schwer, Agenten mit Entropie-basierten $H_\alpha$-Strategien zu implementieren. Für die erste Anfrage sind o. B. d. A. aus Symmetriegründen nur vier Diagonalen und eine Farbe zu testen. Das Berechnen der Entropien für alle ca. $3 \cdot 10^9$ Codes bzgl. der in Frage kommenden Anfragen dauert wenige Minuten. Für die weiteren Anfragen muss man nur die bisher gefragten Farben und genau eine neue Farbe in Betracht ziehen. Im Laufe des Spiels wird mit zunehmender Reduktion der noch möglichen Codes die Bewertung schneller. Ein Computer-Agent benötigt so insgesamt ca. 10 Minuten (abhängig von den Treffern), um das Spiel Beams(6c,6p) lösen.

Weitere Optimierungen, z. B. eine Eröffnungsdatenbank mit den Entropien der ersten Anfragen sind denkbar. Dennoch muss in jedem Fall die Codemenge bzgl. der Antworten, welche vom tatsächlichen Geheimcode abhängen, ausgesiebt werden. Damit wird ein solcher Agent (je nach Hardware und maschinennaher Implementierung) im Sekunden- bis Minutenbereich bleiben.

Zwar eignen sich solche Agenten prinzipiell, um mit Monte-Carlo-Simulation mittlere Suchdauern zu bestimmen, jedoch sind für Genauigkeiten im Sub-Prozent-Bereich (vgl. Ergebnisse bei Mastermind) sehr viele Läufe nötig. Als Faustregel gilt, wegen

[8] Beachtet man noch Drehungen und Spiegelungen, reduziert sich die Datenbank nahezu um den Faktor 12, d. h. die Datenbank benötigt so ca. 4 MB Speicherplatz.

der Konvergenzordnung $\frac{1}{2}$, dass man dazu mindestens 10 000 Läufe braucht, was mehrere Tage dauern würde.

#### Vollständige datenbankbasierte Analyse der Heuristik

Wir verfolgen einen anderen Weg. Indem wir nicht einen konkreten Geheimcode betrachten, untersuchen wir die Aufspaltung der Codemenge bezüglich der verschiedenen Antwortmöglichkeiten. Da diese Aufspaltung disjunkt erfolgt, wächst die Gesamtmenge der Codes bei der Analyse der nächsten Anfrage nicht weiter an. Diese Aufspaltungen werden bis zum Entstehen von Einermengen analysiert. Aus der Suchtiefe lassen sich die Erwartungswerte für die Spieldauer exakt berechnen.

Der Rechenaufwand erhöht sich gegenüber einem Agenten etwas. Zum einen muss in jeder Tiefe (zusammengenommen über alle Antwortfälle) die gesamte Datenbank bewertet werden. Dies bedeutet ein Mehraufwand von maximal der Suchtiefe, bei Beams(6c,6p) ist das etwa ein Faktor von 10 (Zahlenmaterial zur Suchtiefe siehe nächster Abschnitt). Zum anderen sind in größerer Tiefe mehr Anfragen zu analysieren, da die Symmetrien weniger werden (als erste Anfrage mussten ja nur vier von $21 \cdot 6 = 126$ Anfragen getestet werden). Rechnet man mit durchschnittlich 100 Anfragen ergibt sich ein weiterer Faktor 25 im Rechenaufwand.

Tatsächlich ergaben sich so Rechenzeiten von ca. 30–40 Stunden, welche der Überschlagsrechung entspricht ($10 \cdot 25 \cdot 10\,\text{min}$). Das ist zwar nicht wesentlich schneller als die Monte-Carlo Simulation, wir bekommen aber in dieser Zeit exakte Ergebnisse.

Problematisch kann es aber werden, falls es zu Tie-Break-Situationen bei der Bewertung der Anfragen kommt. Bei einer deterministischen Tie-Break-Regel, wie lexikographischer Sortierung, ändert sich nichts. Bei der MIX-Regel, welche eine der am Besten bewerteten Anfragen auswählt, verzweigt sich die Analyse zusätzlich baumartig, da alle Möglichkeiten durchgerechnet werden müssen. Im Falle von $H_1$ und $H_2$ sind Tie-Breaks selten und die Rechenzeit betrug ca. eine bzw. zwei Wochen für die Komplettanalyse aller Codes. Bei den Strategien $H_0$ und $H_\infty$ sind Tie-Breaks viel häufiger, so dass eine exakte Analyse nicht möglich war. Ausnahmsweise wurde hier auf Monte-Carlo-Simulationen zurückgegriffen.

### 4.4.4 Ergebnisse der Analyse von Beams

Analog zu Mastermind präsentieren wir hier die Ergebnisse zu den mittleren Spieldauern für das Spiel Beams. Da die Anfragen Paare $(c, d)$ aus Farbe ($c$olor) und Diagonale ($d$) sind, gibt es zwei mögliche lexikographische Anordnungen. LEX-c wählt zunächst die niedrigste Farbe und LEX-d zuerst die niedrigste Diagonale. Abgesehen von $H_0$ und $H_\infty$ unterscheiden sich die Ergebnisse zu beiden Regeln jedoch so wenig, dass sie in den Diagrammen in Abbildung 4.11 optisch nicht zu unterscheiden sind.

| Strategie | | Beams(3c,6p) | | | Beams(6c,3p) | | | Beams(6c,6p) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| perfekt | mittel | 3.95 | | | unbekannt | | | unbekannt | | |
| | Tie-Break | LEX-c | LEX-d | MIX | LEX-c | LEX-d | MIX | LEX-c | LEX-d | MIX |
| $H_0$ | mittel | 4.0392 | 4.0328 | 4.0460 | 7.8675 | 7.8826 | 8.080* | 10.6639 | 10.6729 | 10.70* |
| | $\Delta$ | 2.26% | 2.10% | 2.43% | | | | | | |
| $H_1$ | mittel | 3.9898 | 3.9919 | 3.9875 | 7.6982 | 7.6978 | 7.6969 | 10.2829 | 10.2830 | 10.2827 |
| | $\Delta$ | 1.01% | 1.06% | 0.95% | | | | | | |
| $H_2$ | mittel | 4.0043 | 4.0032 | 4.0125 | 7.8553 | 7.8538 | 7.8361 | 10.4646 | 10.4641 | 10.4628 |
| | $\Delta$ | 1.37% | 1.35% | 1.58% | | | | | | |
| $H_\infty$ | mittel | 4.2602 | 4.2532 | 4.2677 | 8.3770 | 8.3463 | 8.3026 | 10.9718 | 10.9530 | 10.91* |
| | $\Delta$ | 7.85% | 7.68% | 8.04% | | | | | | |
| | Tie-Break | LEX-c | LEX-d | MIX | LEX-c | LEX-d | MIX | LEX-c | LEX-d | MIX |

* Werte durch Monte-Carlo-Simulation bestimmt (siehe Text)

**Tabelle 4.10:** $H_\alpha$-Strategien für Beams im Vergleich.

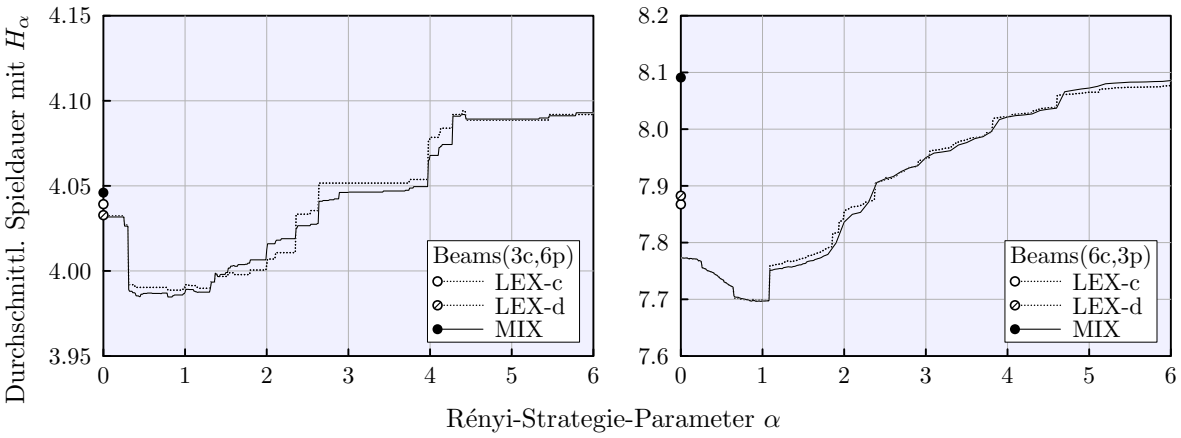



**Abbildung 4.11:** Durchschnittliche Spieldauer für Beams(3c,6p) und Beams(6c,3p) in Abhängigkeit des Rényi-Strategie-Parameters $\alpha$. Man beachte, dass diese Funktionen nicht stetig sind, insbesondere mit Sprüngen bei $\alpha = 0$. Die beiden lexikographischen Tie-Break-Regeln liefern fast identische Resultate, so dass beide Graphen im selben Stil gezeichnet sind.

In Tabelle 4.10 sind Zahlenwerte für $H_\alpha$ mit $\alpha \in \{0, 1, 2, \infty\}$ zusammengefasst. Für Beams(6c,3p) wurde $H_0$-MIX durch Simulation aller 223 920 Codes bestimmt, wobei jeweils nur eine Tie-Break-Instanz simuliert wurde. Für Beams(6c,6p) mit $H_0$-MIX bzw. $H_\infty$-MIX wurden 4000 bzw. 3000 Codes zufällig ausgewählt. Die Fehlerabschätzung bzgl. $2\sigma$-Signifikanz[9] beträgt dabei jeweils 0.05.

Abbildung 4.12 zeigt die Ergebnisse für Beams(6c,6p). In dieser Graphik stecken ca. 3000 Stunden Rechenzeit, da jeder Datenpunkt 30–40 Stunden benötigte. Die

[9] siehe Abschnitt 4.4.5

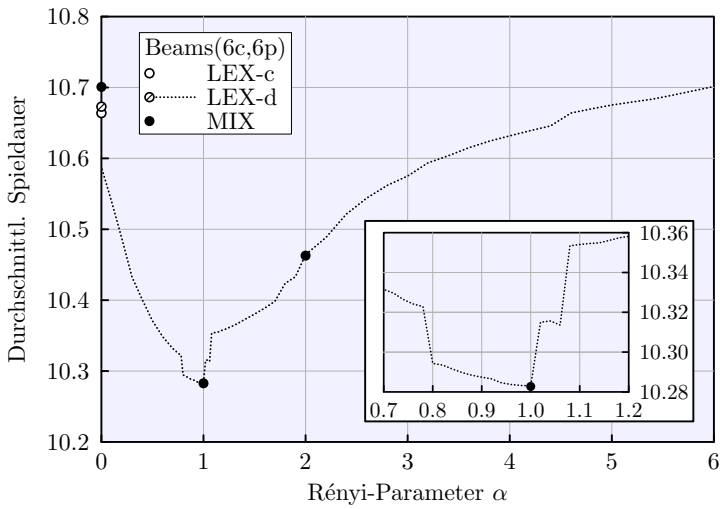



**Abbildung 4.12:** Durchschnittliche Spieldauer für Beams(6c,6p) in Abhängigkeit des Rényi-Strategie-Parameters $\alpha$. Die Auflösung beträgt $\Delta\alpha = 0.1$ bzw. im Bereich der Vergrößerung $\Delta\alpha = 0.02$.

MIX-Regel wurde nur für $H_1$ und $H_2$ analysiert, wobei die Abweichungen zu den LEX-Regeln minimal sind.

### 4.4.5 Verifikation der Erwartungswerte durch Simulationen

Zur Verifikation der Ergebnisses wurde für Beams(6c,6p) die $H_1$-MIX-Strategie simuliert. Dazu wurden $N = 5000$ zufällig gewählte Instanzen vom Agenten gelöst. Die Abbildung 4.13 zeigt das Histogramm. Als Mittelwert der Punkte (Anfragen plus ggf. Strafpunkte) ergibt sich $E = 10.258$ (zum Vergleich: $E_{H_1}^{\mathrm{MIX}} = 10.2827$ war das Ergebnis der vollständigen Analyse). Darin enthalten sind auch die Strafpunkte für unvollständig gelöste und anschließend falsch geratene Codes. In ca. 0.4 % der Fälle gab es zwei solche Strafpunkte.

Die Varianz[10] beträgt 1.836, woraus sich mit der Standardabweichung $\sigma$ ein Toleranzbereich von

$$\Delta_{2\sigma} = \frac{2\sigma}{\sqrt{N}} = 0.038$$

ergibt. Das $2\sigma$-Konfidenzintervall entspricht (für Normalverteilungen) einem Signifikanzniveau von 4.6 %. Wie im Histogramm in Abbildung 4.13 zu sehen ist, sind die Punktzahlen näherungsweise normalverteilt. D. h. mit gut 95 %-iger Wahrscheinlichkeit liegt der wahre Wert von $E$ im Intervall $[10.220, 10.296]$. Das deckt sich mit dem ermittelten exakten Wert 10.2827 (vgl. Tabelle 4.10).

[10] genauer: die erwartungstreu geschätzte empirische Varianz

| Punkte | 0 RF | 2 RF | Gesamt |
|---|---|---|---|
| 6 | 6 | 0 | 6 |
| 7 | 78 | 0 | 78 |
| 8 | 375 | 0 | 375 |
| 9 | 973 | 2 | 975 |
| 10 | 1446 | 4 | 1450 |
| 11 | 1225 | 10 | 1235 |
| 12 | 637 | 8 | 645 |
| 13 | 192 | 9 | 201 |
| 14 | 25 | 4 | 29 |
| 15 | 5 | 1 | 6 |
| Summe | 4962 | 38 | 5000 |

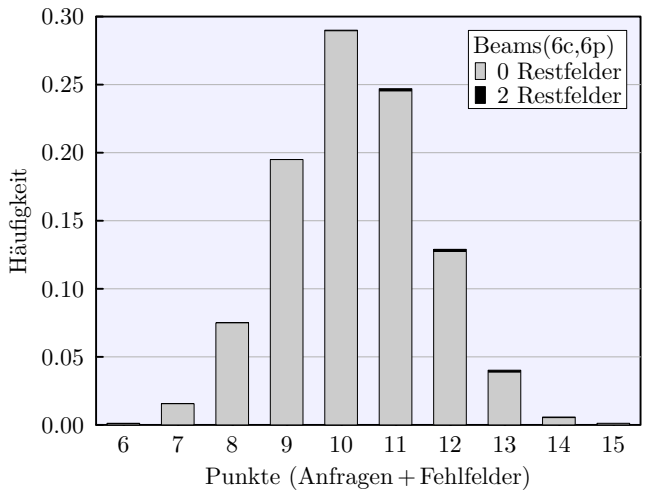



**Abbildung 4.13:** Histogramm der Simulationen zu Beams(6c,6p) mit $H_1$-Strategie und 5000 Instanzen. Die Spalte „2 RF“ gibt an, wie oft es zu zwei Restfeldern kam, d. h. keine differenzierenden Anfragen mehr möglich waren. Statistisch kommt es in der Hälfte dieser Fälle zu zwei Zusatzpunkten durch falsch geratene Felder. Mehr als zwei Restfelder traten nie auf.

Anmerkung: Nutzt man die Simulationsergebnisse, um Aussagen über anderweitig nicht exakt bekannte Größen zu machen, sollte man eine Fehlerabschätzung z. B. über die Hoeffding-Ungleichung vornehmen. Diese Abschätzungen sind jedoch gröber, da nur obere und untere Schranken der Verteilung eingehen. In [44] sind entsprechende Konfidenzintervalle angegeben.[11]

## 4.5 Versagen der Entropie-Heuristik

Die Shannon-Entropie-Heuristik ist nicht immer die beste Rényi-Entropie-Strategie. Dass die Wahl der richtigen Heuristik durchaus von den Eigenheiten eines Suchspiels abhängen kann, wird in diesem Abschnitt gezeigt. Bevor wir dazu ein konkretes, genau zu diesem Zweck konstruiertes Spiel vorstellen, greifen wir die wichtigsten theoretischen Überlegungen der vorherigen Abschnitte auf.

Es sei wieder $C$ die Menge der aktuell noch zulässigen Codes in einem Suchspiel. Bei der Ermittlung einer perfekten Strategie ist gemäß der Rekursionsformel aus Satz 4.2 der Ausdruck

$$\sum_{i=1}^{a_q} |q_i(C)| \langle q_i(C) \rangle \tag{4.1}$$

über alle zulässigen Anfragen $q$ zu minimieren. Dabei haben wir die Wahrscheinlichkeit $p_i(C, q)$ des Auftretens der Antwort $i$ durch die Mächtigkeit der zugehörigen

[11] Bei Berücksichtigung der unteren Schranke (man braucht mindestens fünf Anfragen) ließen sich die in [44] angegebenen Intervalle noch etwas verkleinern.

Antwortmenge ersetzt, denn es gilt $p_i(C,q) = \frac{1}{|C|}|q_i(C)|$. Ist die erwartete mittlere Suchdauer $\langle q_i(C)\rangle$ logarithmisch in $|q_i(C)|$, entspricht die Minimierung von (4.1) der Maximierung der Shannon-Entropie. Genaueres haben wir in Satz 4.7 gezeigt.

Es soll daher ein Suchspiel konstruiert werden, in welchem die erwartete Suchtiefe nicht logarithmisch von der Zahl der verbleibenden Möglichkeiten abhängt. Aufgrund der Abschätzung aus Satz 4.4

$$\langle C\rangle \geq \log_a |C|$$

für die erwartete Suchdauer mit Anfragen der Breite $a$, kommen dafür nur langsame Spiele in Betracht, d. h. die Spieldauer muss stärker als logarithmisch mit der Suchraumgröße wachsen.

Der Einfachheit halber soll es wie bei freien Suchproblemen nur auf die Größe des Suchraumes ankommen, d. h. die Elemente sind gleichberechtigt. Im Unterschied zu freien Suchproblemen muss die Anfragenmenge $Q$ eingeschränkt werden. Dies soll so geschehen, dass wegen der Gleichberechtigung der Elemente, nur Bedingungen an die Mächtigkeit der Komponenten der Partitionen gestellt werden sollen.

Da die mittlere Suchtiefe stärker als logarithmisch mit der Suchraumgröße wachsen soll, muss dies erst recht für die maximale Suchtiefe gelten. Daher muss es asymptotisch stets eine Antwort geben, deren Trefferwahrscheinlichkeit beliebig dicht an eins liegt, formal:

$$\lim_{|C|\to\infty} \overline{C} = 1$$

mit

$$\overline{C} = \min_{q\in Q(C)} \|p(C,q)\|_\infty \,.$$

Umgekehrt bedeutet dies, dass die Summe der Trefferwahrscheinlichkeiten aller anderen Antworten beliebig klein werden muss.

Bevor wir im nächsten Abschnitt ein konkretes Modell-Spiel vorstellen und Laufzeitabschätzungen beweisen werden, soll die Betrachtung solcher langsamen Suchspiele ein wenig motiviert werden.

Dass die Anfragemenge $Q(C)$ von der Menge $C$ der noch verbleibenden Codes abhängen soll, ist zwar eine künstliche Bedingung, sie spiegelt aber typische Phänomene bei realen Deduktionsspielen wider: Betrachtet man beispielsweise Mastermind-Varianten mit sehr vielen Farben, so ist tatsächlich die Trefferwahrscheinlichkeit anfangs klein. Umgekehrt nimmt mit kleiner werdendem Fundus möglicher Codes die Zahl der sinnvollen Anfragen ab, während sie aber besser differenzieren. Wir werden dies in der Diskussion noch einmal aufgreifen.

### 4.5.1 Das Wurzel-Modell

Hier ist das konkrete Beispiel, bei dem sich die $H_1$-Strategie asymptotisch nicht als die optimale Rényi-Strategie erweist. Wir werden diese Aussage zwar nur durch empirisches Zahlenmaterial belegen, uns aber bei der Konstruktion des Spiels von den Vorüberlegungen leiten lassen.

**Definition 4.8** (Wurzel-Modell)**.** *Das Wurzel-Modell ist ein Suchspiel mit einer endlichen Code-Menge, bei dem alle Partitionen $q = (q_1, \ldots, q_{a_q})$ mit*

$$\sum_{i=1}^{a_q-1} (1 + |q_i(C)|) \leq \lceil \sqrt{|C|} \rceil$$

*als Anfragen zulässig sind. Dabei ist $C$ die Menge der aktuell zulässigen Codes.*

Den Ausdruck

$$K(C, q) = \sum_{i=1}^{a_q-1} (1 + |q_i(C)|) = (a_q - 1) + \sum_{i=1}^{a_q-1} |q_i(C)|$$

sollte man sich dabei wie folgt vorstellen. Im Wurzel-Modell sind nur Anfragen $q$ zulässig, deren Kosten[12] $K(C, q)$ durch $\lceil \sqrt{|C|} \rceil$ beschränkt sind. Die Kosten in unserem Modell sind also die Summe aus der Anzahl und der Mächtigkeiten der Komponenten der Partitionen, wobei jedoch eine Komponente unabhängig von ihre Größe „gratis“ ist. Diese „größte“ Komponente hat also

$$|a_{a_q}| = |C| - \sum_{i=1}^{a_q-1} |q_i(C)| = |C| - (K(C, q) - (a_q - 1)) \geq |C| - \lceil \sqrt{|C|} \rceil + a_q - 1$$

Elemente. Die „kleinen“ Komponenten $q_1, \ldots, a_{q_a-1}$ kann man sich als Ausschlusstests vorstellen, welche Fixkosten von Eins und variable Kosten entsprechend der Anzahl der gemeinsam getesteten Elemente verursachen. Das Wurzel-Modell ist also eine Art Group-Testing-Problem, wobei die Anfragen durch die Kostenbeschränkung stark eingeschränkt sind.

**Beispiel 4.9.** *In der Menge $C$ tatverdächtiger Personen soll per DNA-Test eine Person identifiziert werden. Da die Haushaltsmittel knapp sind, können nicht alle Personen auf einmal und einzeln getestet werden. Dazu wird in einigen Personengruppen $q_i$ von jeder Person eine Probe genommen, das Material innerhalb jeder Gruppe gemischt und auf die Referenz-DNA getestet.*[13] *Die Kosten dafür seien $|q_i| + 1$. Insgesamt dürfen die Kosten maximal $L$ betragen, wobei die bewilligten Ausgaben $L$ von der Gesamtzahl der noch Tatverdächtigen abhängen sollen. Bei Misserfolg müssen die Kosten für eine weitere Testrunde bewilligt werden.*

[12] Jeder Anfrage wird per obigem Ausdruck eine Maßzahl zugeordnet, die wir Kosten dieser Anfrage nennen.

[13] Die technische Machbarkeit soll hier keine Rolle spielen.

Im Wurzel-Modell haben wird als Kostenschranke $L$ die Funktion

$$L(C) = \lceil \sqrt{|C|} \rceil$$

gewählt, daher der Name des Modells. Wichtig ist hierbei, dass $L$ langsamer als $|C|$ wächst. Der Summand $|q_i(C)|$ bei den Kosten $K(C, q)$ sorgt dafür, dass die „kleinen“ Teilmengen zusammen höchsten $\sqrt{|C|}$ Elemente haben, dementsprechend hat die Komponente $q_{a_q}(C)$ mindestens $|C| - \sqrt{|C|}$ Elemente und somit ist

$$\overline{C} \geq \frac{|C| - \sqrt{|C|}}{|C|} = 1 - \frac{1}{\sqrt{|C|}}\,.$$

Damit gilt, wie in der Vorüberlegung geforderte, $\lim\limits_{|C|\to\infty} \overline{C} = 1$.

Der Summand $(a_q - 1)$ in der Kostenfunktion zählt die Anzahl der abgespaltenen „kleinen“ Teilmengen und erlaubt damit eine strategische Variabilität. Je breiter eine Anfrage wird (mehr abgespaltene Gruppen), umso mehr vergrößert sich der Restmenge $q_{a_q}(C)$. Es gilt also den richtigen Mittelweg zwischen vielen sehr kleinen oder wenigen etwas größeren abgespaltenen Teilmengen zu finden.

### 4.5.2 Abschätzung der Suchtiefe des Wurzel-Modells

**Satz 4.10.** *Im Wurzel-Modell mit Codemenge $C$ (mit $|C| > 1$) gilt für die erwartete Suchdauer*

$$\frac{1}{2} \cdot \sqrt{|C|} \leq \langle C \rangle \leq 2 \cdot \sqrt{|C|}\,.$$

**Beweis**. Wir beginnen mit der unteren Schranke, d. h. der linken Seite der Ungleichung.

Aufgrund der Kostenbeschränkung zerfällt jede zulässige Anfrage $q = (q_1, \ldots, q_{a_q})$ in eine „große“ Komponente $q_{a_q}$ und eine oder mehrere „kleine“ Komponenten $q_1, \ldots,$ $q_{a_q-1}$, welche wir als abgespaltenen Block bezeichnen (gekennzeichnet durch geschweifte Klammern in Abbildung 4.14). Für die erste Anfrage ist die Gesamtzahl der Elemente des abgespaltenen Blocks durch

$$L_1 := \lfloor \sqrt{|C|} \rfloor$$

nach oben beschränkt. Den weiteren Suchaufwand innerhalb abgespaltener Blöcke ignorieren wir. Für die nächste Anfrage in der Restmenge $q_{a_q}$ ist die Gesamtzahl der Elemente des abgespaltenen Blocks durch $L_2 := \lfloor \sqrt{|q_{a_q}|} \rfloor \leq \lfloor \sqrt{|C|} \rfloor = L_1$ beschränkt. Dies setzt sich so fort.

Jedes Mal werden höchstens $L_1$ Elemente aus der größten Komponente entfernt. Wir gehen nun (in Sinne einer unteren Abschätzung für die Suchtiefe) davon aus, dass

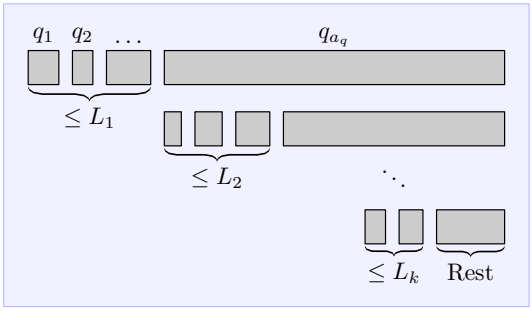



**Abbildung 4.14:** Untere Abschätzung der Suchtiefe beim Wurzel-Modell.

stets ein Block von genau $L_1$ Elementen abgespalten wird und ignorieren den weiteren Suchaufwand in diesem Block. Dann benötigt man mindestens

$$k := \left\lfloor \frac{|C|}{L_1} \right\rfloor$$

Anfragen um die größte Komponente vollständig aufzulösen. Aufgrund der Rundungen enthält die Restkomponente nach $k$ Anfragen noch $|C| - kL_1$ Elemente. Der Suchaufwand bis zur Tiefe $k$ lässt sich nun folgendermaßen abschätzen:

$$\langle C \rangle \geq \frac{1}{|C|} \left( k \cdot (|C| - kL_1) + \sum_{j=1}^{k} j \cdot L_1 \right) .$$

Der Index $j$ gibt dabei an, bei der wievielten Anfrage ein Block mit $L_1$ Elementen abgespalten wurde. Der Summand $k \cdot (|C| - kL_1)$ kommt von der Restmenge. Mit der Gaußschen Summenformel folgt

$$\begin{aligned}
\langle C \rangle &\geq \frac{1}{|C|} \left( k|C| - k^2 L_1 + \frac{k(k+1)}{2} \cdot L_1 \right) \\
&= k - \frac{1}{|C|} k L_1 \frac{k-1}{2} \\
&\geq k - \frac{k-1}{2} && \text{(wegen } kL_1 \leq |C|\text{)} \\
&= \frac{1}{2} \cdot (k+1) \\
&\geq \frac{1}{2} \cdot \frac{|C|}{L_1} \\
&= \frac{1}{2} \cdot \frac{|C|}{\lfloor \sqrt{|C|} \rfloor} \\
&\geq \frac{1}{2} \sqrt{|C|} .
\end{aligned}$$

Kommen wir nun zur oberen Schranke. Es genügt, eine Strategie anzugeben, die der Laufzeitschranke $\langle C \rangle \leq 2 \cdot \sqrt{|C|}$ genügt. Dazu betrachten wir die binäre Strategie

$q = (q_1, q_2)$ mit maximaler Komponente $q_1$. Gemäß Definition 4.8 muss dabei $1+q_1 \leq \lceil\sqrt{|C|}\rceil$ gelten.

Wir führen den Beweis induktiv. Sei $N = |C| > 1$ und die Behauptung gelte für alle Mengen mit weniger als $N$ Elementen. Sei $x = |q_1| = \lceil\sqrt{N}\rceil - 1$. Dann gilt zunächst

$$\frac{1}{2}\sqrt{N} \leq x \leq \sqrt{N}\,.$$

Bezüglich einer binären Anfrage mit $x$ bzw. $N - x$ Elementen gilt nun

$$\begin{aligned}\langle C\rangle &\leq 1 + \frac{x}{N}\langle x\rangle + \frac{N-x}{N}\langle N-x\rangle \\ &\leq 1 + \frac{2}{N}\left(x\cdot\sqrt{x} + (N-x)\sqrt{N-x}\right) \qquad \text{(Induktionsvoraussetzung)} \\ &\leq 2\sqrt{N}\,.\end{aligned}$$

Die letzte Abschätzung ist etwas technisch und gilt für $N \geq 16$ und $\frac{1}{2}\sqrt{N} \leq x \leq \sqrt{N}$, wie in Anhang A.1 gezeigt wird. Für $N \leq 15$ müssen wir – im Sinne eines Induktionsanfangs – die Behauptung explizit zeigen.

| $\lvert C\rvert$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\langle C\rangle \cdot \lvert C\rvert$ | 0 | 2 | 5 | 9 | 12 | 17 | 21 | 27 | 32 | 36 | 43 | 48 | 54 | 62 | 68 |

**Tabelle 4.15:** Erwartete Suchdauer im Wurzel-Modell bei perfekter Strategie. Angegeben ist dabei die Gesamtzahl der Anfragen über alle Codes, d. h. der Quotient aus zweiter und erster Zeile ist der Erwartungswert für die Suchdauer.

In der Tabelle 4.15 sind die erwarteten Suchtiefen für $|C| \leq 15$ aufgeführt. Man überzeugt sich schnell, dass die Behauptung $\langle C\rangle \leq 2\cdot\sqrt{|C|}$ gilt. □

### Bemerkungen

1. Die Tabelle 4.15 hätte man auch mit der binären Strategie aus dem Beweis erstellen können. Lediglich bei $|C| = 12 = 3 + 9$ hätte sich der Gesamtwert 49 ergeben und folglich der Gesamtwert 69 bei $|C| = 15 = 3 + 12$.

2. Die Abschätzungen im Beweis sind teilweise recht grob. Es ist sicher möglich, $\gamma\cdot\sqrt{|C|} \leq \langle C\rangle \leq \Gamma\cdot\sqrt{|C|}$ mit schärferen Konstanten $\gamma$ und $\Gamma$ zu zeigen.

3. Der Ausdruck (4.1), welcher zur Ermittlung der perfekten Strategie zu minimieren ist, verhält sich wie

$$\sum_{i=1}^{a_q} |q_i(C)|\sqrt{|q_i(C)|} = \sum_{i=1}^{a_q} |q_i(C)|^{3/2}\,.$$

D. h. es ist die Norm $||q||_{3/2}$ zu minimieren, was genau der Maximierung der Rényi-Entropie mit $\alpha = \frac{3}{2}$ entspricht.

Die Argumentation im Beweis von Satz 4.7 lässt sich direkt auf andere Laufzeitschranken verallgemeinern. Gilt

$$\gamma \cdot f(|C|) \leq \langle C \rangle \leq \Gamma \cdot f(|C|)$$

so ist die lokale Strategie, welche

$$\sum_{i=1}^{a_q} |q_i(C)| f(|q_i(C)|)\,, \tag{4.2}$$

minimiert, in jeder Anfrage maximal um den Faktor $\frac{\Gamma}{\gamma}$ schlechter als eine perfekte Strategie.

Ist $\frac{\Gamma}{\gamma} \approx 1$, so ist die so gewählte lokale Strategie also nahezu perfekt.

### 4.5.3 Ergebnisse für das Wurzel-Modell

Für das Wurzel-Modell wurden mit Computerunterstützung bis $|C| = 10000$ alle zulässigen Partitionen $q$ mit $K(C, q) \leq \lceil \sqrt{|C|} \rceil$ berechnet und damit sukzessive die perfekte Strategie ermittelt (Tabelle 4.15 zeigt die Werte bis $|C| = 15$).

In Abbildung 4.16 sind für $\alpha \in \{1, \frac{3}{2}, 2\}$ die $H_\alpha$-Strategien normiert bzgl. der perfekten Strategie dargestellt. Vor allem die Rundung in der Kostenbeschränkung verursacht Sprünge die durch die Normierung besonders sichtbar werden. Jedoch werden diese Sprünge immer kleiner, da mit größerer Suchraumgröße die Diskretisierungseffekte weniger bedeutsam werden.

Ab $|C| = 100$ zeigt sich, dass die Shannon-Entropie-Strategie ($H_1$) immer schlechter wird. Demgegenüber ist die $H_{3/2}$-Strategie deutlich überlegen. Ob sie sich der perfekten Strategie annähert ist nicht ganz klar zu erkennen, die Daten im Bereich $6000 \leq |C| \leq 10000$ sprechen eher dagegen.

In der Gegenüberstellung im Abschnitt 4.6.1 (Abbildung 4.17) ist auch für das Wurzel-Modell die $\alpha$-Abhängigkeit für $0 \leq \alpha \leq 6$ für feste Größe der Startmenge ($|C| = 1000$) dargestellt. Ab $\alpha \gtrsim 1.6$ sind dort die $H_\alpha$-Strategien identisch mit der $H_\infty$-Strategie, d. h. es wird jeweils nur eine Gruppe abgespaltet ($a_q = 2$). Für größere Startmengen $C$ verschiebt sich dieser Übergang zu größeren $\alpha$-Werten.

## 4.6 Auswertung und Diskussion

### 4.6.1 Deduktionsspiele im Vergleich

In Abbildung 4.17 sind die Ergebnisse der in diesem Kapitel betrachteten Spiele gegenübergestellt. Die erwartete Spieldauer mit der $H_\alpha$-Strategie hat jeweils einen bassinförmigen Verlauf. Bei Mastermind-360 und allen Beams-Varianten gibt es im

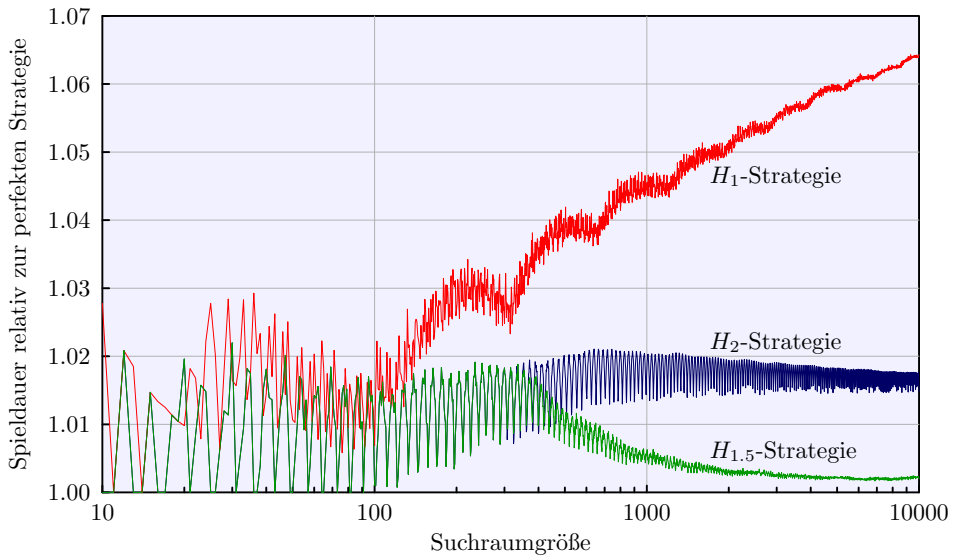



**Abbildung 4.16:** Erwartete Suchtiefe beim Wurzel-Modell für verschiedene Heuristiken (normiert bzgl. der perfekten Strategie).

Bereich $\alpha \approx 1$ ein (lokales) Optimum. Bei Mastermind-1296 gibt es zwar auch ein kleines lokales Minimum bei $\alpha \approx 0.85$, dieses wird aber durch die nicht so günstige erste Anfrage „ABCD“ für $\alpha \lesssim 1.34$ überlagert. Nimmt man dagegen stets die optimale Anfrage „AABC“ als erste Anfrage, so ergibt sich ein zu Mastermind-360 analoges Bild.

Beim Wurzel-Modell liegt das Optimum im Bereich $\alpha \approx 1.35$. Dieses Spiel wurde ja absichtlich so konstruiert, dass die Shannon-Entropie ($\alpha = 1$) nicht optimal ist. Vermutlich verschiebt sich das Minimum für einen größeren Suchraum weiter in Richtung $\alpha = \frac{3}{2}$, was der theoretisch optimalen Heuristik entsprechen würde (vgl. Abschnitt 4.5.2).

Beim Deduktionsspiel Beams sehen die Kurven alle sehr ähnlich aus, wobei sie mit zunehmender Suchraumgröße „glatter“ werden. Die $H_1$-Strategie ist in allen drei Varianten nahezu die beste $H_\alpha$-Strategie.

Die Funktionen sind aufgrund der diskreten Entscheidungsmöglichkeiten stückweise konstant. Große Sprünge treten insbesondere im Limes $\alpha \to 0$ auf. Das Phänomen, dass die $H_0$-Strategien bei Mastermind-1296 besonders gut sind [32], ist bei Beams nicht zu beobachten. Insgesamt ist Mastermind-1296 (man beachte auch das Verhalten bei $\alpha \approx 2$) das außergewöhnlichste der betrachteten Spiele in Bezug auf Rényi-Entropie-Strategien.

Bei Mastermind und Beams spielen die Tie-Break-Regeln (außer bei $H_0$) eine sehr untergeordnete Rolle. Insbesondere sind bei Beams(6c,6p) durch weitere (sehr rechenintensive) Berechnungen zur MIX-Regel keine neuen Erkenntnisse zu erwarten.

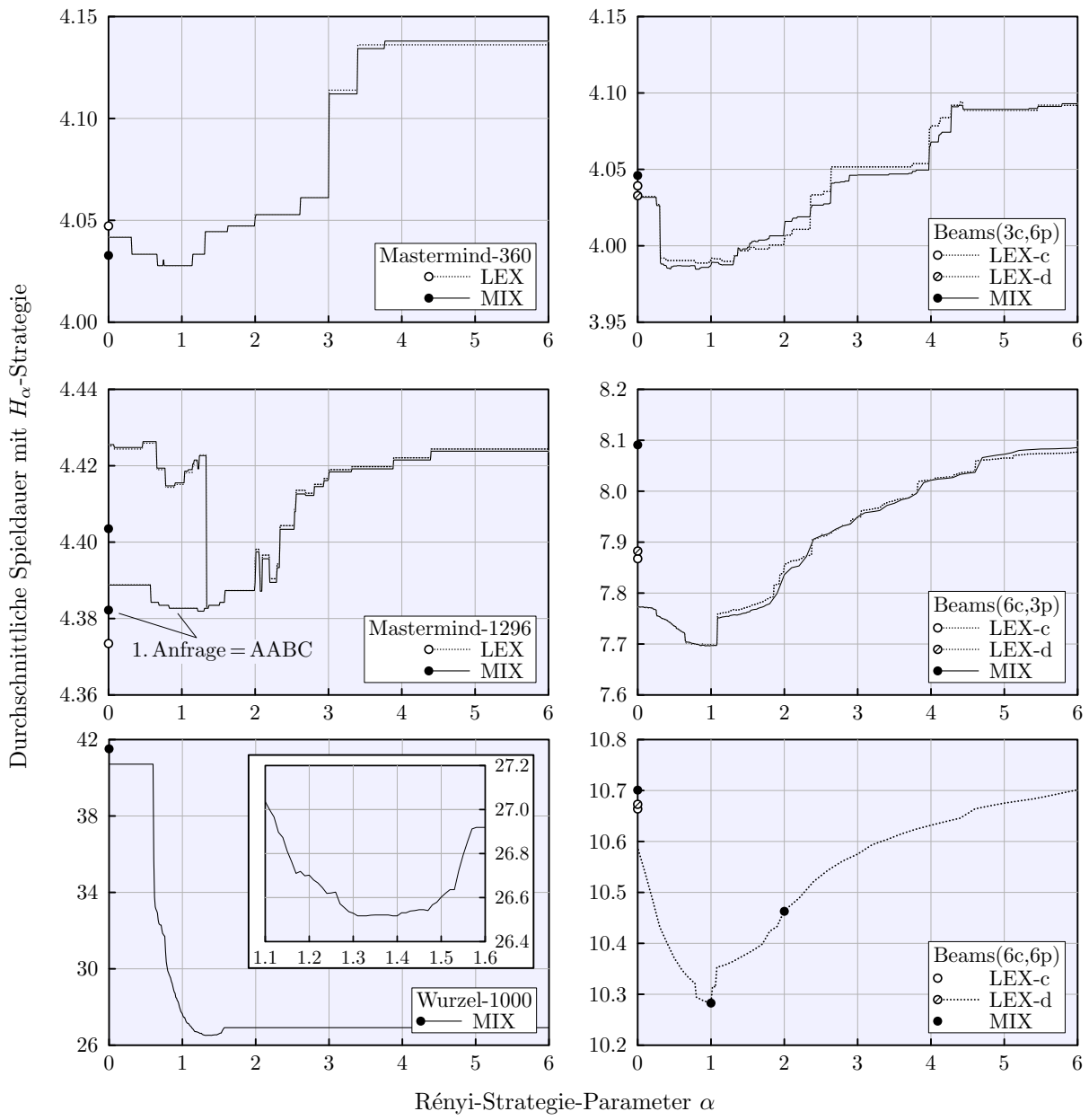



**Abbildung 4.17:** Durchschnittliche Spieldauer für Mastermind, Beams und das Wurzel-Modell (mit $|C| = 1000$) in Abhängigkeit des Rényi-Strategie-Parameters $\alpha$.

### 4.6.2 Die besten Rényi-Strategien

Ein Ausgangspunkt für dieses Kapitel 4 der Arbeit war die Frage, ob es möglich ist, für die Standard-Variante Beams(6c,6p) die $H_1$-Strategie zu implementieren und diese mit anderen Heuristiken [44] zu vergleichen.

In der Tabelle 4.18 sind die erwarteten Spieldauern der besten, auf Maximierung der

| Spiel | Mmind-360 | Mmind-1296 | Beams(3c,6p) | Beams(6c,3p) | Beams(6c,6p) |
|---|---|---|---|---|---|
| perfekte Strategie | 4.0167 | 4.3403 | 3.9500 | unbekannt | unbekannt |
| beste Rényi-Strat. | 4.0278 | 4.3827 | 3.9847 | 7.6963 | 10.2827 |
| $\Delta$ | 0.28 % | 0.98 % | 0.88 % | | |

**Tabelle 4.18:** Die besten Rényi-Entropie-Strategien. Sofern perfekte Strategien bekannt sind, ist die relative Abweichung zu diesen angegeben.

Rényi-Entropie basierenden, Strategien angegeben. Bei Mastermind und Beams(3c,6p) sind diese weniger als 1 % schlechter als die perfekten Strategien. Das Verhalten der Spieldauern für die $H_\alpha$-Strategien in Abhängigkeit von $\alpha$ ist in allen Beams-Varianten recht ähnlich. Wir vermuten daher, dass für Beams(6c,3p) und Beams(6c,6p) die besten Rényi-Strategien (fast identisch mit $H_1$) ähnlich gut sind. Die Ergebnisse bei der freien Suche (Abschnitt 4.2.3) und beim Wurzel-Modell (Abschnitt 4.5.3) legen nahe, dass der relative Unterschied mit zunehmender Suchraumgröße eher abnimmt.

### 4.6.3 Vergleich mit approximierten Entropien bei Beams

In der Diplomarbeit von L. Schreiber [44] wurde das Konzept der Entropie-Maximierung schon einmal für das Spiel Beams angewendet. Im Rahmen der Diplomarbeit wurde als Entropie nur die Shannon-Entropie untersucht. Außerdem konnte diese nur für Beams(3c,6p) berechnet werden. Um für Beams(6c,6p) und sogar Beams(6c,10p) Agenten zu konstruieren, wurde versucht die Entropien ohne vollständige Datenbank zu schätzen. Dazu wurden durch verschiedene kombinatorische Analysen der bisherigen Antworten die noch zulässigen Farben für jedes Feld eingeschränkt. Im Anschluss wurden daraus die Entropie von Farb-Diagonal-Kombinationen (=Anfragen) unter Annahme der Unabhängigkeit der Felder und Gleichverteilung der zulässigen Farben auf jedem Feld folgendermaßen approximiert:

$$H'(c, D) = -\sum_{d \in D} p_{d,c} \log_2 p_{d,c} + (1 - p_{d,c}) \log_2 (1 - p_{d,c}) \,.$$

Dabei ist $c$ eine Farbe und $D$ eine Diagonale (als Menge der Felder der Diagonalen). $p_{d,c}$ ist die geschätzte Wahrscheinlichkeit, dass das Feld $d$ die Farbe $c$ hat. Also $p_{d,c} = 0$, wenn die Farbe $c$ durch die kombinatorische Analyse ausgeschlossen wurde bzw. $p_{d,c} = \frac{1}{n_d}$, wenn $c$ eine der $n_d$ noch zulässigen Farben ist.

Aufgrund der Näherungen

- unvollständige kombinatorische Analyse (zu wenig Farben ausgeschlossen),
- Annahme der Gleichverteilung der (vermuteten) zulässigen Farben und
- Annahme der Unabhängigkeit der Felder einer Diagonalen (unrealistisch, da gleichfarbige Felder verbunden sein müssen)

| Strategie | durchschnittliche Suchdauer | | |
|---|---|---|---|
| | Beams(3c,6p) | Beam(6c,6p) | Beams(6c,10p) |
| perfekt | 3.95 | unbekannt | unbekannt |
| $H_1$ (Shannon-Entropie) | 3.9875 | 10.2827 | unbekannt |
| $H_\alpha$ (Rényi-Entropien) | $\approx 4.0\ldots4.3$ | $\approx 10.3\ldots10.9$ | unbekannt |
| Ergebnisse von L. Schreiber [44]: | | | |
| $H_1$ (Shannon-Entropie) | 3.93±0.11 | — | — |
| $H'$ (zulässige Farben gleichverteilt) | 4.66±0.11 | 13.04±0.16 | 17.94±0.20 |
| $U$ (unsichere Felder) | 4.77±0.11 | 13.31±0.16 | 18.29±0.20 |

**Tabelle 4.19:** Vergleich der mittlere Suchdauern der $H_\alpha$-Strategien mit den Ergebnissen aus [44] für approximierte Entropien.

kann man von dieser Heuristik keine sehr guten Agenten erwarten. Tatsächlich liefert sogar die Heuristik

$$U(c, D) = |\{d \in D \mid \text{ Farbe } c \text{ ist für } d \text{ zulässig}\}|\,,$$

welche nur die unsicheren Felder zählt kaum schlechtere Agenten. Im Sinne einer Most-Parts-Strategie kann man $U$ als Approximation der Rényi-Entropie $H_0$ interpretieren.

Tabelle 4.19 stellt die Ergebnisse unserer exakten Analyse (für Beams(3c,6p) und Beams(6c,6p)) und die Ergebnisse von L. Schreiber für die approximierten Entropien gegenüber. Die Mittelwerte von L. Schreiber entstanden durch Monte-Carlo-Simulation, wobei die Gleichverteilung ebenfalls nur approximiert wurde[14]. Die angegebenen Konfidenzintervalle zum Niveau 0.05 könnte man gemäß Fußnote 11 auf Seite 70 noch etwas verkleinern.

Insgesamt zeigt sich, dass die approximierten Entropien $H'$ und $U$ deutlich schlechtere Agenten liefern als sämtliche $H_\alpha$-Strategien. Fazit: Eine gute Approximation der (Rényi-)Entropie ist wichtiger als die Wahl des optimalen Wertes für $\alpha$.

### 4.6.4 Zum Wurzel-Modell

Wir konnten zeigen, dass man durch geeignete Konstruktion von Suchspielen erreichen kann, dass die Shannon-Entropie nicht die beste Rényi-Entropie ist. Durch den Zusammenhang (4.2) zwischen erwarteter Suchdauer und einer lokalen Heuristik, kann man im Prinzip für jede Heuristik ein passendes Suchspiel konstruieren. Übliche Suchprobleme haben jedoch eine logarithmische erwartete Suchdauer, so dass die $H_1$-Strategie zu präferieren ist. Endliche Probleme, z. B. konkrete Deduktionsspiele, können jedoch

[14] Bei Beam(6c,6p) zeigte sich, dass die von L. Schreiber verwendete Methode zur Erzeugung zufälliger Codes gut funktioniert. Im Vergleichstest unter Verwendung der Codes aus der vollständigen Datenbank (vgl. Abschnitt 4.4.3) ergaben sich keine wesentlichen Unterschiede in den Ergebnissen.

auch Charakteristiken aufweisen, die zu anderen besseren Heuristiken führen (vgl. Mastermind-1296).

Ein ähnliches Phänomen konnte Reams [39] beobachten. Er untersuchte das Buchstabenspiel Hangman (dt. Galgenmännchen, Galgenraten) bei welchem durch Abfragen einzelner Buchstaben ein Wort (fester Länge aus einer vorgegebenen Wortliste) zu erraten ist. Experimentelle Ergebnisse zeigten, dass die Entropie-Heuristik mit zunehmender Wortlänge im Vergleich zur optimalen Strategie immer schlechter wird. Ab einer Wortlänge von sechs wird sie von der simplen Strategie, welche einfach den häufigsten Buchstaben der verbleibenden Wortliste wählt, dominiert und diese simple Strategie scheint sich sogar der optimalen Strategie anzunähern.

## 4.7 Ausblick und offene Fragen

Bei den exemplarisch betrachteten Spielen Beams und Mastermind sind die $H_\alpha$-Strategien offenbar nahezu perfekt. Jedoch sind für Beams – bis auf Beams(3c,6p) – die perfekten Strategien unbekannt. Aufgrund der Komplexität der Suchräume wird es in absehbarer Zeit kaum möglich sein diese exakt zu ermitteln. Jedoch sind durch folgende Ansätze Fortschritte bei der Bestimmung unterer Schranken für perfekte Strategien denkbar:

- Durch unvollständige Baumsuche lassen sich untere Schranken bestimmen. Dazu bricht man die Suche in einer gewissen Tiefe ab und schätzt den Rest durch logarithmische Suchdauer ab.

  Bei der Ermittlung der perfekten Strategie für Mastermind und Beams(3c,6p) wurde dies erfolgreich in Branch-and-Bound-Techniken angewendet.

- Ausgehend von freien Suchproblemen (vgl. Abbildung 4.4, Seite 58) sind perfekte Strategien und lokale Heuristiken genauer zu studieren. Verallgemeinerungen auf andere Suchprobleme – mit geeigneten Voraussetzungen – scheinen möglich. Für Spiele/Suchprobleme mit beliebig groß wählbarem Suchraum sind experimentelle und theoretische Ergebnisse für das asymptotische Verhalten (z. B. für die $H_1$-Strategie) zu erwarten.

  Als Spiel könnte man wieder Mastermind-Varianten, jedoch mit $p$ Stiften und $c$ Farben studieren.

Bei kleineren Suchproblemen (bzw. im Endspiel) kann man die lokalen Heuristiken sicher noch etwas verbessern, indem man entweder mehrere Schritte vorausrechnet, oder ab einer genügend kleinen Restmenge die perfekte Strategie bestimmt.

Für Beams(6c,10p) ist der Suchraum so groß, dass die Berechnung der Entropie nicht exakt möglich ist. In diesen Fall kann man versuchen, die Entropien zu approximieren, beispielsweise wie in [44]. Wie bei Beams(6c,6p) zu sehen (vgl. Abschnitt 4.6.3), haben diese recht einfachen Approximationen deutliches Verbesserungspotential. In wie weit sich dieses Potential mit einfachen Mitteln, z. B. einem genetischen Ansatz [11], nutzen

lässt, muss sich zeigen. Jedenfalls ist durch die hier exakt analysierte $H_1$-Strategie für Beams(6c,6p) eine wichtige Vergleichsmarke (mit 10.28 Anfragen) gegeben.

Die Gesamtrechenzeit für die vollständigen Analysen der $H_\alpha$-Strategien, die Monte-Carlo-Simulationen und Ermittlung der perfekten Strategien betrug ca. 10 000 CPU-Stunden. Obwohl diese Ressourcen bei Verwendung von Mehr-Kern-Computern zur Verfügung standen und die Codes durchaus effizient programmiert wurden, gibt es punktuell Verbesserungsmöglichkeiten:

- Zur Implementierung eines schnellen $H_1$-Agenten für Beams(6c,6p) sollte man mit einer Eröffnungsdatenbank arbeiten und das Aussieben der Codemenge durch hardware-nähere Programmierung beschleunigen.
- Für die eingehendere Untersuchung des Wurzel-Modells für Suchraumgrößen $m > 10\,000$ kann man nicht mehr alle zulässigen Partitionen berechnen. Die Idee der effizienteren Rekursion aus Satz 4.5 lässt sich auch auf das Wurzel-Modell übertragen, da die Kostenbeschränkung additiv ist bzgl. der Komponenten der Partition.

Weitere offene Fragestellungen ergeben sich bzgl. der Laufzeitabschätzungen für Suchprobleme:

- Wie kann man die Überlegungen aus Satz 4.7 so verallgemeinern, dass man $\langle M \rangle = \gamma f(|M|)$ in einem asymptotischen oder stochastischen Sinne voraussetzen kann.
- Gilt im Wurzel-Modell asymptotisch $\gamma = \Gamma$, d. h. existiert $\lim\limits_{|C|\to\infty} \frac{\langle C \rangle}{\sqrt{|C|}}$ ?
- Der Ausdruck (4.2) gibt an, welche Heuristik der perfekten Strategie (lokal) am nächsten kommt. Für Spiele mit logarithmischer Dauer ist dies die $H_1$-Strategie, jedoch ist nicht klar, wie stark sich andere $H_\alpha$-Strategien davon unterscheiden. Die Ergebnisse zu Beams(6c,6p), vgl. Abbildung 4.17, lassen vermuten, dass das Optimum bei $\alpha = 1$ mit zunehmender Suchraumgröße schärfer ausgeprägt ist.
- Um das Verhalten der $H_\alpha$-Strategien weiter zu studieren kann man weitere intermediäre Instanzen (z. B. von Beams) betrachten, bei denen der Coderaum durch zufällige Vorauswahl künstlich reduziert wird. Diese, das ursprüngliche Spiel zwar stark verfremdende Modifikation, dürfte für die Analyse von Computer-Agenten unproblematisch sein.

Speziell für Beams sind auch weitere spielstrategische Überlegungen interessant:

- Minimierung der maximalen Suchtiefe, vgl. [31], [28], [19]
- Minimax-Spielanalyse, vgl. [13, Kap. 3.13], d. h. der Codemaker wählt den Code nicht zufällig mit gleicher Wahrscheinlichkeit, sondern bevorzugt „schwerere“ Codes. Auch hier gibt es Vorarbeiten von L. Schreiber [44].

# Kapitel 5

# Analyse von Shannons Analog-Heuristik für das Switching-Game

*In diesem Kapitel untersuchen wir eine analoge Heuristik für das Spiel Bridg-It (auch Shannon-Switching-Game, Gale, Bird-Cage). Eine spieltheoretische Analyse dieses Spiels ist schon länger bekannt [35], wir werden deshalb darauf nur kurz eingehen und uns hauptsächlich mit der Analyse der Heuristik befassen.*

*Nach einigen graphentheoretischen Vorüberlegungen führen wir Shannons analoge Heuristik (Widerstandsmodell) ein und untersuchen, wie gut ein darauf basierender Agent spielt. Als Gegner kommen insbesondere auch Monte-Carlo-Agenten zu Einsatz. Das auffällige Verhalten dieser Agenten gibt Anlass zur weiteren Untersuchung von Zufallseinflüssen auf Heuristiken. Daraus wird schließlich ein Modell zur künstlichen Randomisierung von Heuristiken abgeleitet.*

## 5.1 Einführung

1951 realisierte Claude Shannon einen Automaten (siehe Abbildung 5.1), der das Verbindungsspiel Bridg-It, auch bekannt als Gale oder Bird-Cage, gegen menschliche Gegner spielen konnte.

Obwohl dieser Automat nur aus einer einfachen Schaltung aus Schaltern, Lampen und Widerständen besteht, berichtete Shannon, dass der Automat recht stark spielte. Sofern er den ersten Zug machte, siegte er fast immer gegen menschliche Spieler (vgl. Zitat auf Seite 93).

### 5.1.1 Das Shannon-Switching-Game

Das Shannon-Switching-Game[1] ist ein Nullsummen-Strategie-Spiel mit voller Information für für zwei Spieler, die wir **Cut** und **Short** nennen.

[1] Das Shannon-Switching-Game, benannt nach seinem Erfinder Claude Shannon, ist eine Verallgemeinerung von Bridg-It. Zumindest in spezieller Form wurde es unabhängig auch von anderen (z. B. David Gale) erfunden und ist unter den Namen Gale bzw. Bird-Cage bekannt.

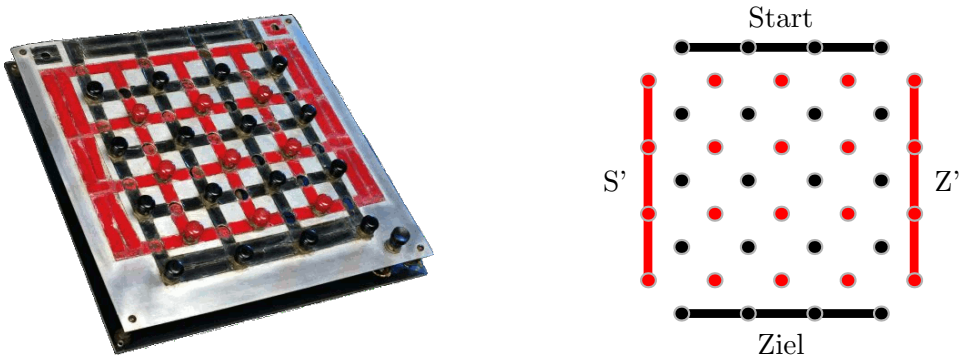



**Abbildung 5.1:** Shannons Bridg-It-Automat. Links: Foto des Original-Automaten [46], rechts: schematische Darstellung.

Das Spiel wird auf einem endlichen, zusammenhängenden, ungerichteten Graphen gespielt, wobei zwei Knoten **Start** und **Ziel** ausgezeichnet sind.

Beide Spieler ziehen abwechselnd. Welcher Spieler beginnt, muss vor Spielbeginn festgelegt werden.[2] Für Cut besteht ein Zug darin, eine (unmarkierte) Kante des Graphen zu entfernen. Ist Short am Zug, markiert er eine (noch vorhandene) Kante.

Short gewinnt, sobald er eine durchgängige Verbindung vom Start- zum Ziel-Knoten herstellt. Ist dies nicht mehr möglich, gewinnt Cut. Ein unentschieden ist nicht möglich.

Die Knoten des Graphen entsprechen dabei genau den schwarzen Punkten in Abbildung 5.1. Entfernte Kanten entsprechen dabei Verbindungen von roten Punkten. Eine solche „rote Brücke“ schließt dann „schwarze Brücke“ aus.

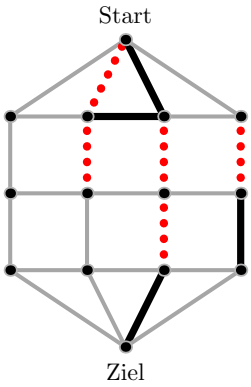



**Abbildung 5.2:** Shannon-Switching-Game.

### 5.1.2 Perfekte Strategien

Das Spiel ist endlich, da jede Kante nur einmal markiert bzw. entfernt werden kann. Wurden alle Kanten markiert bzw. entfernt, so lässt sich eindeutig ein Sieger ermitteln. Short hat gewonnen, falls Start- und Ziel-Knoten durch markierte Kanten verbunden sind. Da es kein Unentschieden gibt, muss es für einen der beiden Spieler eine Gewinnstrategie geben. Diese wird im Allgemeinen von der Struktur des Graphen $G$ abhängen, aber auch davon welcher der beiden Spieler beginnt.

Das Problem lässt sich graphentheoretisch analysieren. Lehman [35] präsentierte eine vollständige Analyse unter Verwendung der Theorie der Matroide. Eine erste darauf

[2] Später werden wir ausschließlich selbstduale Graphen betrachten und können ohne Einschränkung fordern, das stets Cut beginnt.

basierende Computerimplementation realisierte 1969 Stephen M. Chase vom IBM Watson Research Center [18]. Es gibt weitere Publikationen, die z. T. mit elementaren graphentheoretischen Mitteln auskommen, z. B. [37] und [36].

Grundidee vieler dieser Analysen ist es, nach unabhängigen überspannenden Bäumen im Graphen $G$ zu suchen. Der Arbeit von Mansfield [37] folgend, sind Graphen mit zwei solchen Bäumen zu betrachten.

**Definition 5.1.** *Ein Graph heißt* ***positiv****, falls er zwei überspannende Bäume enthält, die keine gemeinsamen Kanten haben.*

Dann gilt folgender Satz:

**Satz 5.2.** *$G$ enthält genau dann einen positiven Teilgraphen der die Knoten Start und Ziel enthält, wenn der Spieler Short eine Gewinnstrategie als nachziehender Spieler hat.*

#### Bemerkungen

- Einen schönen, nicht sehr langen Beweis findet man in [37].
- Hat Short als nachziehender Spieler eine Gewinnstrategie, so auch als anziehender Spieler.
- Hat Short als nachziehender Spieler keine Gewinnstrategie, so lässt sich die Frage, ob Short als anziehender Spieler eine Gewinnstrategie hat mit Suchtiefe eins (indem man alle Züge von Short durchprobiert) auf den Fall als nachziehender Spieler zurückführen.
- Die Argumentation mit den zwei überspannenden Bäume ist konstruktiv. Man kann daraus Algorithmen ableiten, die in polynomieller Zeit entscheiden, ob ein Graph **positiv** ist und ggf. diese Spannbäume angeben.
- In [35] sind für die Klasse der Bridg-It-Graphen (vgl. Definition 5.5) die überspannenden Bäume explizit angegeben.

### 5.1.3 Planare Graphen und Dualität

Den zum Switching-Game gehörenden Graphen bezeichnen wir mit $G$ und die zwei ausgezeichneten Knoten (Start und Ziel) mit $A$ und $B$. Das Switching-Game wird also durch das Tripel $S(G, A, B)$ charakterisiert. Wenn nichts anderes angegeben ist, so soll Cut der Startspieler sein. Cut färbe die Kanten von $G$ rot und Short färbe sie schwarz. D. h. es werden keine Kanten entfernt, sondern nur entsprechend eingefärbt, wobei einmal gefärbte Kanten nicht umgefärbt werden dürfen.

Schränkt man das Switching-Game auf planare Graphen $G$ ein, so lassen sich enge Beziehungen zwischen den zunächst sehr unterschiedlich erscheinenden Spielern Cut und Short finden.

Wir fordern also nun, dass sich $G$ planar in die Ebene einbetten lässt, und zwar so dass die Punkte $A$ und $B$ beide auf dem äußeren Rand liegen. Gleichzeitig machen wir die kleine Verallgemeinerung, nämlich dass $G$ auch Mehrfachkanten enthalten darf. Auf die Planarität hat dies keinen Einfluss. Im weiteren Verlauf, insbesondere bei der Implementierung der Heuristik, werden diese Mehrfachkanten jedoch benötigt.

Ein planarer Graph hat die schöne Eigenschaft, dass man einen (ebenfalls planaren) dualen Graphen konstruieren kann, indem man gewissermaßen Ecken und Flächen vertauscht. Aus $G = (V, E)$ wird $G = (V', E')$, wobei $V'$ mit den Flächen von $G$ (einschließlich der äußeren) identifiziert wird. Für jede Kante $e \in E$ enthält $E'$ das Paar aus den beiden an $e$ angrenzenden Flächen als duale Kante.

**Bemerkungen**

- Mehrere gemeinsame Kanten zwischen zwei Flächen führen zu Mehrfachkanten in $G'$.
- $G$ und $G'$ haben dieselbe Kantenzahl und es gibt nach Konstruktion eine kanonische Bijektion zwischen den Kanten.
- Der zu $G'$ duale Graph ist wieder $G$.

Nun können wir das duale Shannon-Spiel definieren:

**Definition 5.3.** *Das Shannon-Spiel $S' = (G', A', B')$ heißt dual zum Shannon-Spiel $S = (G, A, B)$, wenn die Graphen $G + (A, B)$ und $G' + (A', B')$ dual sind, wobei sich die Kanten $(A, B)$ und $(A', B')$ entsprechen. Dabei ist $G + (A, B)$ der Graph, der aus $G$ durch Hinzufügen der Kante $(A, B)$ entsteht.*

**Bemerkungen**

- Wegen der Forderung, dass $A$ und $B$ am Rand liegen sollen, ist $G + (A, B)$ ebenfalls planar.
- Bis auf Vertauschung sind (bei gegebener Einbettung von $G$) die Knoten $A'$ und $B'$ eindeutig bestimmt. Da wir ungerichtete Graphen betrachten ist diese Vertauschung unerheblich.
- Aufgrund der kanonischen Bijektion zwischen den Kanten können wir die Farben von $G$ übernehmen, wobei wir aber rot und schwarz vertauschen. Ungefärbte Kanten bleiben ungefärbt.

Es gilt folgender Satz:

**Satz 5.4.** *Sind in $S = (G, A, B)$ alle Kanten gefärbt, so existiert genau dann in $G$ ein schwarzer Kantenzug von $A$ nach $B$, falls es in $S' = (G', A', B')$ keinen schwarzen Kantenzug von $A'$ nach $B'$ gibt.*

**Beweis**. Dies ist eine einfache elementare Eigenschaft planarer Graphen, deren Beweis wir hier nur kurz skizzieren wollen.

Wir betten beide Graphen planar ein, so dass sich genau die dualen Kanten überkreuzen, und $A$, $A'$, $B$ und $B'$ (in dieser Reihenfolge) am Rand liegen.

Gäbe es schwarze Kantenzüge von $A$ nach $B$ und von $A'$ nach $B'$, so müssten sie sich überkreuzen. Die dualen Kanten an einer solchen Kreuzung müssen aber nach Konstruktion verschiedenfarbig sein.

Gibt es keine schwarzen Kantenzüge von $A$ nach $B$, so betrachten wir die schwarze Zusammenhangskomponente von $A$. Betrachtet man nun alle von außen diese Zusammenhangskomponente berührenden Flächen, so definieren diese einen zusammenhängenden schwarzen Teilgraphen des dualen Graphen, der $A'$ und $B'$ verbindet. □

Nun sehen wir, dass die Rollen beider Spieler analog sind: Während Short versucht, auf $G$ eine (schwarze) Verbindung herzustellen, versucht Cut dasselbe auf $G'$. Hat ein nachziehender Short-Spieler auf dem dualen Spiel eine Gewinnstrategie, so hat ein anziehender Short-Spieler auf $G$ keine Gewinnstrategie.[3]

Besonders interessant ist der Fall selbstdualer Spiele, d. h. wenn der Graph $G'$ des dualen Spiels isomorph zu $G$ ist, wobei die Knoten $A$ und $B$ den Knoten $A'$ und $B'$ entsprechen. Dann hat nämlich der anziehende Spieler eine Gewinnstrategie.

Für konkrete Untersuchungen betrachten wir später die folgende einparametrige Klasse selbstdualer Graphen.

**Definition 5.5** (Bridg-It-Graph). *Ein Bridg-It-Graph der Ordnung $M$ besteht aus einem $(M-1) \times M$ Gitter dessen Knoten der langen äußeren Seiten mit zwei weiteren Knoten $A$ bzw. $B$ verbunden sind.*

Beispiele für diese Graphen sind in Abbildung 5.3 dargestellt.

## 5.2 Shannons analoge Heuristik

### 5.2.1 Das Widerstandsmodell

Shannon betrachtete den Graphen $G$ als elektrischen Schaltkreis, wobei zwischen Start- und Ziel-Knoten eine Spannung angelegt wird. Die Kanten sollen alle denselben elektrischen Widerstand $R$ haben, mit Ausnahme markierter (schwarzer) Kanten (sehr kleiner Widerstand $R_s$, Kurzschluss) und gelöschter (roter) Kanten (sehr großer Widerstand $R_c$, kein Stromfluss). Den Grenzübergang zu $R_s = 0$ und $R_c = \infty$ machen wir später.

[3] Damit lässt sich zumindest bei planaren Graphen die in den Bemerkungen zu Satz 5.2 erwähnte die Suchtiefe von eins für Short als anziehenden Spieler umgehen.

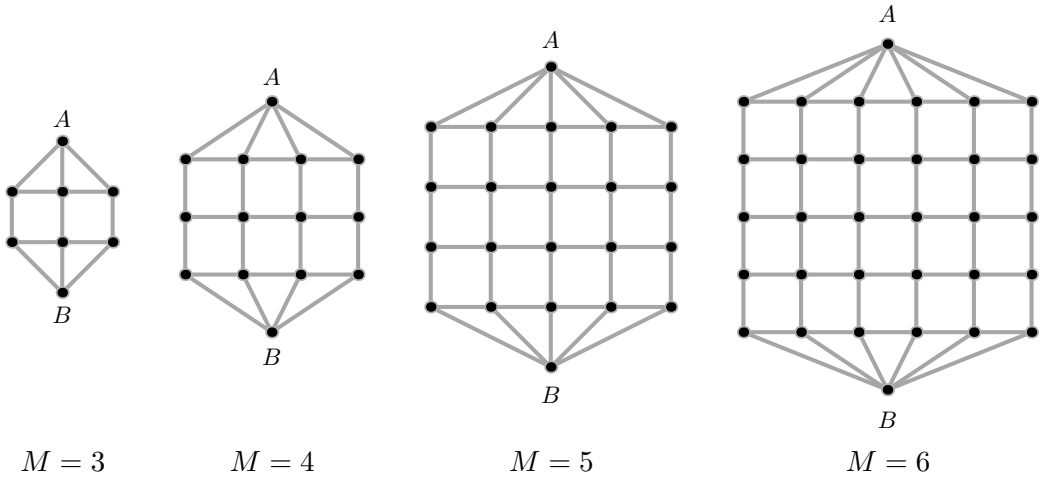



**Abbildung 5.3:** Beispiele selbstdualer Bridg-It-Graphen. Der Graph der Ordnung $M$ hat $M(M-1)+2$ Knoten und $M^2+(M-1)^2$ Kanten.

Die von Shannon eingeführte Heuristik besagt nun:

**Wähle die Kante mit dem größten Stromfluss für den nächsten Zug.**

Diese Heuristik ist durchaus einleuchtend. Kanten, durch die viel Strom fließt, liegen in Bereichen, an denen der Graph nur durch wenige Kanten zusammenhängt. Andererseits bilden Ketten aus Kanten eine Reihenschaltung mit hohem Widerstand, so dass der Stromfluss eher gering ist. Ketten sind strategisch nicht so wichtig, das sie für Short verloren sind und Cut nur reagieren muss.

Shannon verwendete für die technische Umsetzung (vgl. Abbildung 5.1) der Heuristik einfach (eine Kombination von Widerständen und) Lampen als Kanten. Beim Aufdrehen der Spannung zeigte die am hellsten leuchtende Lampe (=größter Stromfluss[4]) den nächsten Zug des Automaten an, vgl. Abbildung 5.4.

### 5.2.2 Das Netzwerk als Gleichungsystem

Die Widerstands-Heuristik führt zusammen mit der Kirchhoffschen Knotenregel[5] auf das Gleichungssystem

$$\sum_{i \in N_j} (x_i - x_j) g_{ij} = 0 \quad \forall j \neq A, B\,, \tag{5.1}$$

[4] Man beachte, das dieser Automat das Widerstands-Modell nur näherungsweise abbildet, weil der Widerstand einer Lampe aus physikalischen Gründen nicht konstant ist, sondern erheblich vom Stromfluss abhängt. An Energiespar-Lampen mag man gar nicht erst denken...

[5] In jedem Knoten ist die Summe aller Ströme gleich Null, mit Ausnahme der Knoten $A$ und $B$, an die ja eine zusätzliche Spannungsquelle angelegt wird.

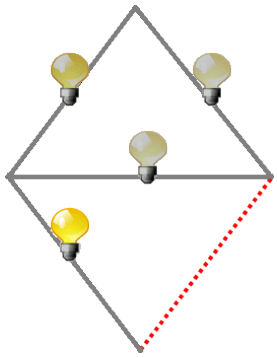


**Abbildung 5.4:** Analog-Realisierung des Switching-Game durch Glühlampen.

wobei $N_j$ alle mit $j$ verbundenen Knoten bezeichnet. Der Variablen $x_i$ entspricht das elektrische Potential im Knoten $i$ und die $g_{ij}$ sind die Leitwerte, also die reziproken Widerstände. Wir setzen $x_A = 1$ und $x_B = 0$ (äußere Spannungsquelle). Stellt man die Gleichungen des Systems (5.1) nach $x_j$ um, folgt

$$x_j = \sum_{i \in N_j} \frac{g_{ij}}{\sum_{k \in N_j} g_{kj}} x_i \quad \forall j \neq A, B\,, \quad x_A = 1\,,\ x_B = 0\,. \tag{5.2}$$

Da die Summe der Koeffizienten auf der rechten Seite gleich Eins ist bzw. im Falle $A \in N_j$ oder $B \in N_j$ kleiner als Eins[6] ist, handelt es sich um ein strikt diagonaldominantes Gleichungssystem und ist damit eindeutig lösbar. Insbesondere bieten sich hierfür auch numerische Iterationsverfahren an.

### 5.2.3 Dualität der Heuristik

Es stellt sich die Frage, ob die Widerstands-Netzwerk-Heuristik nicht einen der beiden Spieler benachteiligt. Sollte dies so sein, müsste man diesem Spieler raten, die Heuristik auf dem dualen Spiel anzuwenden. Da die Heuristik in jedem Schritt auf dem aktuell eingefärbten Graphen beruht, sind auch bei anfänglich selbstdualem Startgraphen, die Zwischenstände nicht selbstdual.

Erfreulicherweise sind jedoch die Ergebnisse der Heuristik in beiden Spielen kompatibel, zumindest bei geeigneter Wahl der Widerstandswerte. Wir können das Ganze zunächst etwas allgemeiner formulieren.

Es sei $(G, A, B, g)$ ein zusammenhängender planarer Graph mit zwei ausgezeichneten Knoten $A$ und $B$ auf dem äußeren Rand. Außerdem habe jede Kante ein nichtnegatives Gewicht $g_{ij}$. Zu diesem Graphen existiert ein eindeutig bestimmtes Knoten-

[6] Im Fall $i = A, B$ ist $x_i$ keine Variable, sondern führt auf eine Inhomogenität des Gleichungssystems. Die Summe der Koeffizienten ist also kleiner als Eins, falls $j$ mit $A$ oder $B$ verbunden ist.

potential $x$ mit $x_A = 1$ und $x_B = 0$, so dass gilt

$$\sum_{i \in N_j} (x_i - x_j) g_{ij} = 0 \quad \forall j \neq A, B\,,$$

wobei $N_j$ alle mit $i$ verbundenen Knoten enthält.

Es sei $(G', A', B', g')$ die dazu duale Konfiguration (im Sinne der Definition 5.3) mit $g'_{i'j'} = 1/g_{ij}$, wobei sich die Kanten $(i, j)$ und $(i', j')$ entsprechen sollen.

Dann gilt:

**Satz 5.6.** *Es gibt auf $G'$ ein eindeutig bestimmtes Potential $x'$ mit $x'_{A'} = 1$, $x'_{B'} = 0$ und*

$$\sum_{i' \in N_{j'}} (x'_{i'} - x'_{j'}) g'_{i'j'} = 0 \quad \forall j' \neq A', B'.$$

*Für dieses Potential sind die Potentialdifferenzen $(x'_{i'} - x'_{j'})$ und $(x_i - x_j)$ über den dualen Kanten proportional zueinander.*

**Beweis**. Analog zum Graphen $G$ existiert wegen der Struktur des Gleichungssystems auch das Potential $x'$ auf $G'$ und ist eindeutig bestimmt. Um die Proportionalität zu zeigen, machen wir den Ansatz

$$x'_{i'} - x'_{j'} = (x_i - x_j) \cdot g_{ij} \cdot \lambda$$

und zeigen, dass dieses Potential den geforderten Bedingungen genügt.

Es sei $Z_j$ der Rand derjenigen Fläche in $G'$, die dem Knoten $j$ aus $G$ entspricht. Dann kann man die behauptete Formel des Satzes leicht nachrechnen:

$$\sum_{i' \in N_{j'}} (x'_{i'} - x'_{j'}) g'_{i'j'} = \sum_{(i,k) \in Z_{j'}} \lambda (x_i - x_k) g_{ij} g'_{i'j'} = \lambda \cdot \sum_{(i,k) \in Z_{j'}} (x_i - x_k) = 0 \quad \forall j' \neq A', B'.$$

Die Konstante $\lambda$ ist so zu bestimmen, dass $x'_{A'} = 1$ und $x'_{B'} = 0$ erfüllt werden kann.[7] □

[7] Ausgehend von $x'_{B'} = 0$ kann man für festes $\lambda$ das Potential $x'$ aus dem Ansatz eindeutig rekonstruieren, wobei noch sicherzustellen ist, dass dies wegunabhängig ist. Dazu genügt es zu zeigen dass die Summe der so angesetzten Potentialdifferenzen auf dem Rand jeder Fläche von $G'$ verschwindet:

$$\sum_{(i',k') \in Z_j} (x'_{i'} - x'_{k'}) = \sum_{i \in N_j} (x_i - x_j) \cdot g_{ij} \cdot \lambda = 0 \quad \forall j \neq A, B\,.$$

Die Konstante $\lambda$ kann schließlich so gewählt werden, dass $x'_{A'} = 1$ gilt.

**Bemerkungen**

- Die Idee des Satzes und Beweises besteht darin, dass Knoten- und Maschenregeln (Kirchhoffschen Gesetze) zueinander dual sind.
- Die Knoten $A$, $B$, $A'$ und $B'$ sind nicht von geschlossenen Kantenzügen des jeweils dualen Graphen umgeben und deshalb von den Gleichungen auszuschließen. Stattdessen gibt es für diese Knoten eine Potentialvorgabe.
- Bei den sich entsprechenden Kanten $(i, j)$ und $(i', j')$ ist die Orientierung zu beachten.
- Der Faktor $\lambda$ spielt für die Shannon-Heuristik keine Rolle.
- Die Shannon-Heuristik betrachtet den Stromfluss im Widerstandsnetzwerk, wobei die Kanten als Widerstände mit $R_{ij} = 1/g_{ij}$ betrachtet werden ($g = 1/R$ wird Leitwert genannt). Der Stromfluss ist damit $f_{ij} = (x_i - x_j)g_{ij}$ bzw. $f'_{i'j'} = (x'_{i'} - x'_{j'})g_{i'j'} = (x_i - x_j) \cdot \lambda$. D. h. es gilt

  $$f'_{i'j'} = f_{ij} \cdot \lambda / g_{ij}$$

  D. h. wenn alle Kanten denselben Widerstandswert haben[8], liefert die Shannon-Heuristik auf beiden Graphen dieselbe Bewertungsreihenfolge der Kanten.
- Verwendet man dagegen endliche Leitwerte $(g_c, g, g_s)$ für rote, ungefärbte und schwarze Kanten, dann liefert die Shannon-Heuristik im dualen Graphen dieselbe Bewertungsreihenfolge der Kanten, falls $g_c g_s = g^2$ gilt ($g_c$ und $g_s$ müssen dual sein!).
- Im Grenzfall $g_c \to 0$, $g_s \to \infty$, sind gelöschte und kurzgeschlossenen Kanten ebenfalls dual zueinander.

### 5.2.4 Direkte und Iterative Löser

Das Gleichungssystem (5.1) kann man direkt lösen. Für nicht zu große Graphen ist das mit einem Eliminationsverfahren realisierbar. Die Zeitkomplexität beträgt dabei $O(N^3)$, wobei $N$ die Anzahl aller Knoten ist. Unter Beachtung, dass die Adjazenzmatrizen eher dünn besetzt sind, lässt sich sogar noch etwas Rechenzeit einsparen.

Alternativ bietet sich die iterative Lösung des Systems (5.2) mit dem Gauß-Seidel-Verfahren an. Aufgrund der Diagonaldominanz konvergiert das Verfahren stets gegen eine Lösung. Die Zeitkomplexität pro Iteration beträgt $O(\sum N_i)$ und entspricht also der Kantenzahl des Graphen. Durch Überrelaxation gemäß

$$x^{(t+1)} = F(x^{(t)}) \quad \Longrightarrow \quad x_{\text{SOR}}^{(t+1)} = (1-\omega)x_{\text{SOR}}^{(t)} + \omega F(x_{\text{SOR}}^{(t)})$$

[8] Im Switching-Game sind dazu rote Kanten zu eliminieren und mit schwarzen Kanten verbundene Knoten zusammenzufassen.

lässt sich die benötigt Iterationszahl reduzieren. Der optimale Überrelaxationsparameter $\omega$ liegt im Intervall $(1, 2)$ und kann experimentell geschätzt oder dynamisch angepasst werden.

Als Startlösung kann man einfach $x_i = \frac{1}{2}$ wählen[9]. Während des Spiels ist es möglich, die Lösung vor dem letzten Zug als Startlösung zu verwenden.

### 5.2.5 Implementierung von Computer-Agenten

Wir wollen stets das ideale Modell mit unendlichem Leitwert für markierte Kanten und Leitwert Null für gelöschte Kanten verwenden. In diesem Falle sind während des Spiels von Cut gelöschte Kanten aus der Adjazenzmatrix zu entfernen. Werden dabei Teile des Graphen abgespaltet (die Heuristik schlägt solche Züge nicht vor), braucht man die zugehörigen Variablen nicht mehr zu betrachten. Umgekehrt sind durch Short verbundene Knoten zu einer Variablen zusammenzufassen. Dabei können Mehrfachkanten entstehen.

Zum Prüfen der Siegbedingung muss man sich die Zusammenhangskomponenten des Graphen bzgl. ungefärbter bzw. markierter Kanten ansehen. Bei (hinreichend) exakter Lösung des Gleichungssystems, kann man sogar einfacher argumentieren: Short gewinnt, wenn der eine Kante mit Potentialdifferenz $|x_i - x_j| = 1$ spielt und Cut hat gewonnen, wenn seine gespielte Kante nach dem Zug die Potentialdifferenz eins hätte.

## 5.3 Ist die Shannon Heuristik optimal?

Wir untersuchen diese Frage anhand der selbstdualen Graphen Bridg-It-Graphen (siehe Abbildung 5.3). D. h. wir benötigen den dualen Graphen nicht und berechnen die Heuristik, egal ob für Cut oder Short, auf dem primalen Spielgraphen.

Außerdem betrachten wir ausschließlich das ideale Modell mit Widerstand Null für markierte Kanten und Widerstand unendlich für gelöschte Kanten.

### 5.3.1 Gewinnen gegen die Heuristik

Obwohl Cut als anziehender Spieler eine Gewinnstrategie hat, ist es schon auf recht kleinen Graphen möglich, als nachziehender Spieler gegen die Heuristik zu gewinnen. Schon bei $M = 3$, siehe Abbildung 5.5, verliert Cut, wenn er im ersten Zug A1, E1, A5 oder E5 wählt und der Gegenspieler das ausnutzt (er spielt nicht nach Heuristik). Wählt Cut jedoch einen gemäß Heuristik gleichwertigen Eröffnungszug (C1, A3, C3, E3 oder C5), ist er nicht zu schlagen.

[9] Im Fall der selbstdualen Bridg-It-Graphen (siehe Abbildung 5.3) kann man für den ersten Schritt die exakte Lösung direkt angeben: Das Potential ist in jeder waagerechten Schicht konstant und fällt linear von Schicht zu Schicht.

In Abbildung 5.5, Teil c), sehen wir, dass die Shannon-Heuristik im dritten Zug ungünstig spielt. Die Kanten A3 und B4 haben beide eine Spannungsdifferenz (entspricht Stromfluss) von 21, während E3 nur den Wert 20 hat. A3 and B4 bilden aber ein Kette und für Cut besteht keine Notwendigkeit, eine dieser Kanten zu entfernen, bevor Short die andere markiert hat.

Bei der nächstgrößeren Instanz, die dem Originalspiel von Shannon (siehe Abbildung 5.1) entspricht, gibt es für jeden (heuristik-konformen) Anfangszug von Cut eine Gewinnstrategie für Short. Aus Symmetriegründen sind dabei nur die vier Eröffnungszüge A1, C1, A3 bzw. C3 zu untersuchen. In Abbildung 5.6 ist für jeden dieser Züge ein Beispiel für eine siegreiche Gegenstrategie angegeben. Bis auf das Auftreten von Zweier-Ketten und trivialen Endspielsituationen sind alle Züge der Shannon-Heuristik eindeutig bestimmt, d. h. mehrere Kanten mit maximalem Fluss treten nicht auf. Damit ist Abbildung 5.6 eine vollständige Aufstellung die beweist, dass die Shannon-Heuristik bei omnipotentem[10] Gegenspieler immer verliert.

Trotzdem ist es für einen menschlichen Spieler nicht so einfach zu erkennen, wie die Heuristik zu schlagen ist. Gardner [26] berichtete über Shannon:

> „When his machine has first move, it almost always wins. Out of hundreds of games played, the machine has had only two losses when it had the first move, and they may have been due to circuit failures or improper playing of the game. If the human player has first move, it is not difficult to beat the machine, but the machine wins if a gross error is made“.

### 5.3.2 Shannon vs. Shannon

Was passiert, wenn beide Spieler gemäß Shannon-Heuristik spielen? Durch Computergestützte kombinatorische Analyse konnten wir zeigen, dass bis zu einer bestimmten Graphgröße, nämlich $M = 8$ (d. h. ein $7\times8$-Gitter plus Start und Ziel), immer der anziehende Spieler gewinnt und zwar unabhängig von der Kantenwahl beim Auftreten gleichwertiger Kanten.

Bei größeren Graphen (dieser selbstdualen Bauart) kann auch der nachziehende Spieler gewinnen. Insbesondere hängen die Ergebnisse dann von der Kantenwahl beim Auftreten gleichwertiger Kanten ab. Dabei spielt vor allem der erste Zug eine große Rolle.

Tatsächlich sind, abgesehen vom ersten Zug, gleichwertige Kanten selten, so dass es möglich ist, alle Fälle vollständig zu analysieren. Leider steigt die Komplexität stark mit $M$ an, denn in jedem Zug ist ein Gleichungssystem mit $O(M^2)$ Variablen zu lösen. Dazu kommt noch die Partiedauer in der Größenordnung $O(M)$ bis $O(M^2)$. Schließlich sind alle $O(M)$ Eröffnungszüge zu analysieren. Je nachdem, wie man das Gleichungssystem löst (direkt, sparse-Solver oder iterativ), bekommt man eine nicht zu verachtende Zeitkomplexität im Bereich von $O(M^5)$ bis $O(M^9)$.

[10] Der Spieler weiß, dass sein Gegner nach Shannon-Heuristik spielt und nutzt dies aus.

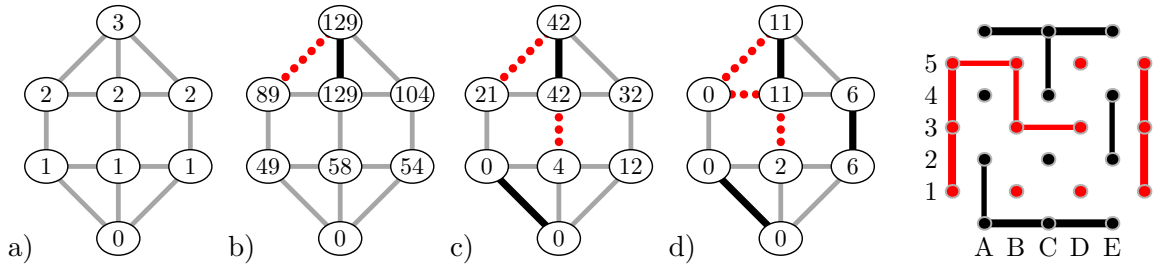
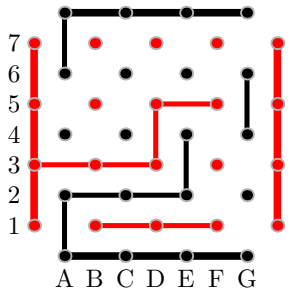



**Abbildung 5.5:** Kleinste Verlust-Instanz für die Shannon-Heuristik.
Links: Graphen und Potentiale (normiert auf ganzzahlige Werte) jeweils mit Cut am Zug. Dieser wählt eine Kante mit größter Potentialdifferenz.
Rechts: Spiel nach 6 Zügen im Bridg-It-Diagramm (Züge werden durch die Koordinaten des Mittelpunktes der gewählten Kante notiert): 1. A5, 2. C5, 3. C3, 4. A1, 5. B4, 6. E3. Es handelt sich um eine direkte Gewinnsituation für Short, weiterer Spielverlauf: 7. E1, 8. D2, 9. C1, 10. B2, 11. E5, 12. D4.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. A1 | 2. C1 | 1. C1 | 2. A1 | 1. A3 | 2. C3 | 1. C3 | 2. E3 |
| 3. C3 | 4. E3 | 3. A3 | 4. C3 | 3. C1 | 4. A1 | 3. E1 | 4. D2 |
| 5. E5 | 6. A7 | 5. B2 | 6. E1 | 5. siehe links [b] | | 5. C1 | 6. A1 |
| 7. A5 | 8. D4 | 7. D2 | 8. E3 | | | 7. A3 | 8. B2 |
| 9. C5 | 10. G5 | 9. E5 | 10. G5 | | | 9. E5 | 10. A7 |
| 11. G7 | 12. F6 | 11. G7 | 12. D4 | | | 11. D4 | 12. G5 [c] |
| 13. E7 | 14. D6 | 13. C5 | 14. A7 | | | | |
| 15. F4 | 16. G3 | 15. A5 [a] | 16. F6 | | | | |
| 17. G1 | 18. F2 | 17. E7 | 18. D6 | | | | |
| 19. C7 | 20. B6 | 19. F4 | 20. G3 | | | | |
| 21. E1 | 22. D2 | 21. C7 | 22. B6 | | | | |
| | | 23. G1 | 24. F2 | | | | |




| | |
|---|---|
| 13. G7 | 14. F6 |
| 15. F4 | 16. G3 |
| 17. E7 | 18. D6 |
| 19. C7 | 20. B6 |
| 21. G1 | 22. F2 |

[a] Cut wählt eine von zwei gleichwertigen Kanten einer Kette (A5 and B4), siehe Diskussion zum Fall $M = 3$ und Abbildung 5.5 c).

[b] Ab hier ergibt sich dieselbe Zugfolge wie im zweiten Fall.

[c] Dargestellt im Diagramm: offensichtliche Gewinnstellung für Short.

**Abbildung 5.6:** Switching-Game mit Brigt-It-Graph der Ordnung $M = 4$. Bei allen Eröffnungszügen verliert der nach Shannon-Heuristik spielende Startspieler Cut.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. E9 [a] | 2. C9 | 29. H12 | 30. A15 [d] | 57. C5 | 58. A5 | 85. Q8 | 86. R9 |
| 3. C11 [b] | 4. A11 | 31. B14 | 32. J11 | 59. B6 | 60. P5 | 87. Q10 | 88. R11 |
| 5. A13 [c] | 6. B12 | 33. H10 | 34. J9 | 61. N6 | 62. P7 | 89. R1 | 90. T1 |
| 7. C13 | 8. E13 | 35. F16 | 36. M17 | 63. N8 | 64. P9 | 91. S2 | 92. T3 |
| 9. D12 | 10. E11 | 37. K16 | 38. M15 | 65. N10 | 66. P11 | 93. S4 | 94. T5 |
| 11. D10 | 12. G9 | 39. K14 | 40. M13 | 67. N12 | 68. P13 | 95. Q12 | 96. R13 |
| 13. F10 | 14. G11 | 41. K12 | 42. M11 | 69. N14 | 70. P15 | 97. S6 | 98. T7 |
| 15. F12 | 16. G13 | 43. K10 | 44. M9 | 71. P3 | 72. N4 | 99. S8 | 100. T9 |
| 17. G15 | 18. F14 | 45. M7 | 46. K8 | 73. M3 | 74. K4 | 101. S10 | 102. T11 |
| 19. E15 | 20. D14 | 47. H8 | 48. J7 | 75. J3 | 76. H4 | 103. S12 | 104. T13 |
| 21. C15 | 22. J15 | 49. J5 | 50. H6 | 77. G3 | 78. F4 | 105. T15 | 106. S14 |
| 23. H14 | 24. J13 | 51. G5 | 52. F6 | 79. E3 | 80. R3 | 107. R15 | 108. Q14 |
| 25. J17 | 26. H16 | 53. E5 | 54. K6 | 81. Q4 | 82. R5 | 109. P17 | 110. N16 |
| 27. G17 | 28. G7 | 55. M5 | 56. D6 | 83. Q6 | 84. R7 | | |

[a] symmetrische Alternative: 1. P9 2. R9.

[b] symmetrische Alternative: 3. C7 4. A7.

[c] es entsteht eine Kette (A15, B14).

[d] Auflösen der Kette (äquivalente Zugfolge: 30. B14 31. A15).

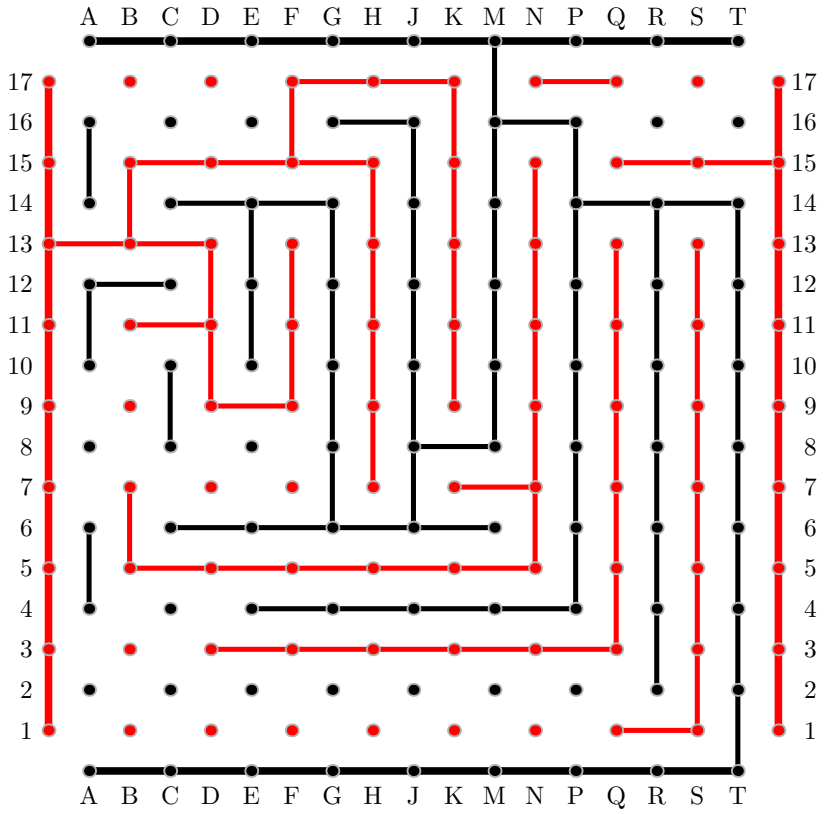



**Abbildung 5.7:** Beispielpartie mit verlierendem Startspieler Cut für $M = 9$ (Shannon vs. Shannon).

Bis $M = 20$ konnten wir diese Analyse durchführen. Abgesehen von Symmetrien, gibt es bei $M = 9$ genau einen Eröffnungszug, nämlich E9, für den der nachziehende Spieler gewinnen kann. Bis auf wenige triviale Situationen ist der sich anschließende Spielverlauf eindeutig bestimmt. Dieser Spielverlauf ist in Abbildung 5.7 dargestellt.

Somit beträgt für $M = 9$ die Siegquote des Startspielers genau $1 - 2/81$, da es 81 bzgl. der Heuristik gleichwertige Eröffnungszüge gibt, von denen E9 und P9 verlieren.

Betrachten wir größere Graphen (bis $M = 20$), so fällt die Siegquote. Für $M \to \infty$ vermuten wir, dass die Siegquote gegen 50 % konvergiert.

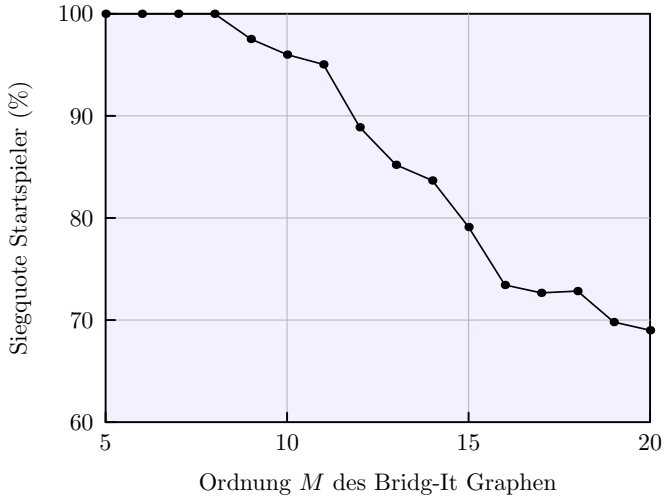



**Abbildung 5.8:** Switching-Game bis $M = 20$: Siegquote des anziehenden Spielers (Shannon vs. Shannon).

## 5.4 Randomisierung

### 5.4.1 Monte-Carlo vs. Shannon-Heuristik

Für das Auffinden von Verlustzügen der Shannon-Heuristik (vgl. Abschnitt 5.3.1) wurden zunächst reine Monte-Carlo-Agenten eingesetzt. D. h. der Startspieler Cut spielt nach der Widerstands-Heuristik und Short spielt mit einem MC($k$)-Agenten. Schon für die Instanzen $M = 3$ und $M = 4$ konnte der MC($k$)-Agent einige Partien für sich entscheiden. Die Siegquoten sind in Abbildung 5.9 dargestellt. Interessanterweise gibt es ein Optimum für mittlere Monte-Carlo-Parameter. Für zu große Monte-Carlo-Parameter sinkt die Siegquote wieder auf den theoretischen Spielwert Null.

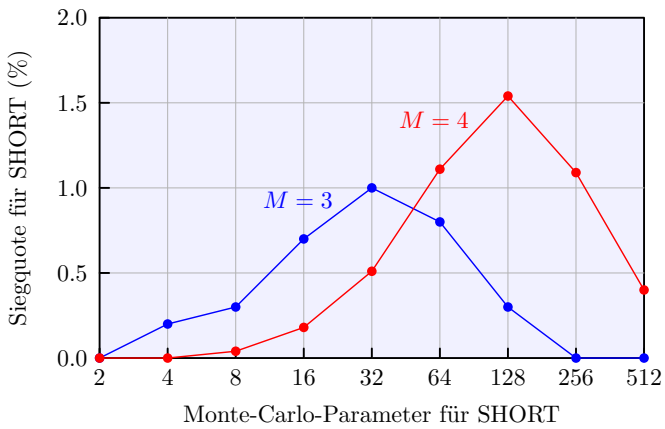



**Abbildung 5.9:** Switching-Game: Siegquote des nachziehenden Spielers Short (Shannon vs. Monte-Carlo).

### 5.4.2 Künstliche Randomisierung

Das Verhaltens der Monte-Carlo-Agenten lässt den Schluss zu, dass ein gewisser Zufallsanteil Agenten verbessern kann. Während dieses Merkmal bei Monte-Carlo-Agenten mit endlichem Monte-Carlo-Parameter schon eingebaut ist, wollen wir ein Konzept entwickeln, künstliche Randomisierung zu beliebigen Heuristiken hinzuzufügen.

Die Randomisierung soll so erfolgen, dass die Wahrscheinlichkeit mit der ein Zug gewählt wird mit der Bewertung durch die Heuristik korreliert ist. Außerdem soll die Stärke des Zufallseinflusses durch einen Parameter $T$ gesteuert werden.

Durch die Widerstands-Heuristik im Switching-Game ist eine Zug-Bewertung durch die Potentialdifferenz $\Delta X_{ij} = |x_i - x_j|$ der zugehörigen Kante gegeben. Wir machen nun folgenden Ansatz:

**Definition 5.7** (Künstliche Randomisierung)**.** *Der mit Parameter $T \geq 0$ randomisierte Shannon-Heuristik-Agent wählt für den nächsten Zug eine Kante $(i, j)$ gemäß der Wahrscheinlichkeitsverteilung*

$$p_{ij} = \frac{1}{Z} \cdot \left( \frac{\Delta X_{ij}}{\Delta X_{\max}} \right)^{1/T} .$$

*Dabei ist $\Delta X_{ij} = |x_i - x_j|$ Potentialdifferenz der Kante $(i, j)$ gemäß Shannon-Heuristik und $\Delta X_{\max} = \max\limits_{(i,j)} \Delta X_{ij}$.*

*Die Konstante $Z$ ist so zu bestimmen, dass $\sum\limits_{(i,j)} p_{ij} = 1$ gilt.*

Die Normierung bzgl. der „besten“ Kante $\Delta X_{\max}$ entspricht dem Anlegen einer solchen äußeren Spannung, so dass diese Kante die Potentialdifferenz eins hat.

Der Parameter $T \geq 0$ steuert die Stärke des Zufallseinflusses. Dabei gilt

- $T \to 0$: Eine Kante mit maximalem $\Delta X_{ij}$ wird gewählt.
- $T \to \infty$: Gleichverteilung unter allen Kanten (mit $\Delta X_{ij} > 0$).

Der Randomisierungsansatz hat die Struktur einer Gibbs-Boltzmann-Verteilung

$$p_{ij} = \frac{1}{Z} \exp\left(-\frac{E_{ij}}{T}\right),$$

welche Ensembles im thermodynamischen Gleichgewicht beschreibt. Damit dies so zusammenpasst ist die (dimensionslose) Energie als

$$E_{ij} = \log \Delta X_{\max} - \log \Delta X_{ij} \tag{5.3}$$

anzusetzen.[11] Den Paramter $T$ bezeichnen wir deshalb als Temperatur der Randomisierung.

### 5.4.3 Ergebnisse mit künstlicher Randomisierung

Zum Testen des Konzeptes wurden Self-Play-Partien von Agenten mit randomisierter Shannon-Heuristik untersucht. Dazu wurden beide Agenten mit einer individuellen Temperatur im Intervall $[0.001, 1]$ versehen und je Datenpunkt 1000 Partien simuliert. Die Ergebnisse für $M = 4$ zeigt Abbildung 5.10 oben.

Man erkennt, dass kleine Temperaturen ($T \lesssim 0.02$) keine relevante Veränderung bewirken. Für große Temperaturen ($T \gtrsim 0.4$) spielen die Agenten so zufällig, dass sie gegen einen nicht (bzw. wenig) randomisierten Agenten verlieren.

Interessant ist der Bereich $T \approx 0.1$. Hier schafft es Short, ca. 5 % der Partien zu gewinnen. Dies ist schon deutlich besser als der Monte-Carlo-Agent MC(128), der ca. 1.5 % der Partien gewann (vgl. Abbildung 5.9).

Abbildung 5.10 mitte (mit 10 000 Partien pro Datenpunkt) zeigt den Fall gleicher Temperaturen, bei dem im Grenzfall $T \to \infty$ nur noch der Anzugsvorteil (bei ansonsten zufälligem Spiel) das Ergebnis bestimmt.

Abbildung 5.10 unten (ebenfalls mit 10 000 Partien pro Datenpunkt) zeigt, dass sich die für Short erreichbare Siegquote für größere Graphen weiter erhöht, während die dazu notwendige Temperatur ausgehend von $T \approx 0.1$ ein wenig abfällt.

[11] Diese Energie ist als Charakterisierung der Kantenbewertung zu verstehen und beschreibt nicht den elektrischen Schaltkreis im physikalischen Sinne.

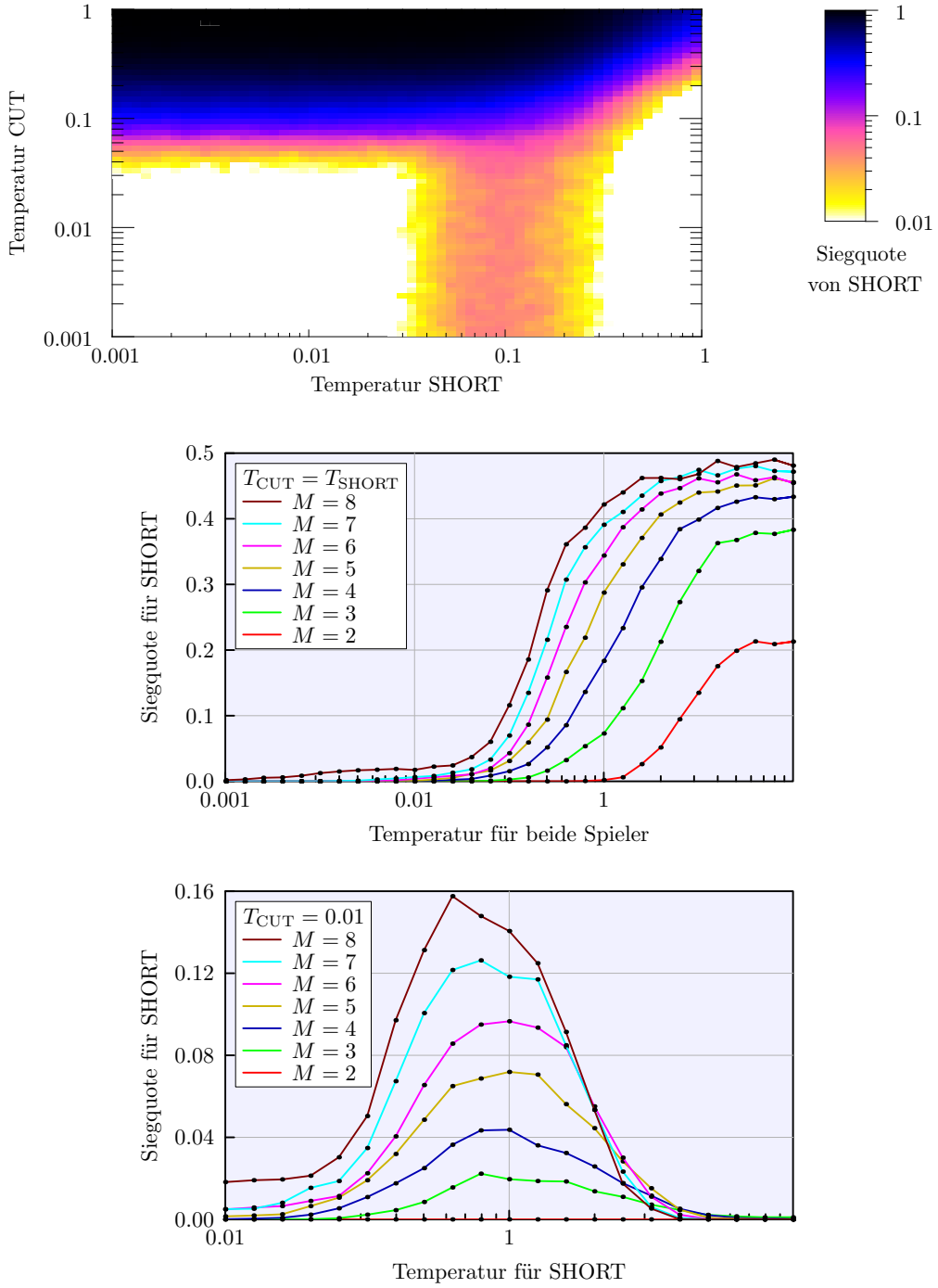



**Abbildung 5.10:** Switching-Game mit Shannon-Heuristik: Siegquote für den nachziehenden Spieler Short, wobei beide Spieler mit Temperatur-Randomisierung spielen.
Oben: individuelle Temperaturen bei Graphgröße $M = 4$.
Mitte und unten: Graphgröße $M = 2, \ldots, 8$ und identischen Temperaturen (mitte) bzw. feste Temperatur für Cut (unten). Für $M = 4$ sind dies also Schnitte durch die obere Abbildung.

### 5.4.4 Ergebnisse mit Randomisierung durch finite Iteration

Aufgrund der recht hohen Zeitkomplexität für das exakte Lösen des Gleichungssystems der Widerstand-Heuristik, kann man auch iterative Löser verwenden (siehe Abschnitt 5.2.4). Zum einen spart dies erheblich Rechenzeit ein (Ausnutzung der dünn besetzten Koeffizientenmatrix, wenige Iterationen und ggf. Beschleunigung durch SOR). Zum anderen ist bei vorzeitigem Abbruch schon eine Art Randomisierung eingebaut, zumindest bei zufälliger Wahl der Startlösung.[12]

Auch für diese Art der Randomisierung haben wir experimentell Daten ermittelt. Dazu lassen wir den Startspieler Cut gemäß der nicht-randomisierten Heuristik gegen den Spieler Short mit randomisierter Heuristik spielen. Dabei wurden Temperatur und Iterationszahl unabhängig voneinander variiert, um das Zusammenwirken beider Randomisierungen studieren zu können.

Die Abbildung 5.11 zeigt die Ergebnisse für jeweils 1000 Simulationen pro Datenpunkt mit fester Startlösung $x^{(0)} = \frac{1}{2}$ und ohne Relaxation in der Gauß-Seidel-Iteration. Für hohe Iterationszahlen bzw. niedrige Temperaturen sind die beide Randomisierungsarten auch separat enthalten. Man erkennt wieder die charakteristische Temperatur im Bereich von $T \approx 0.1$ (bzw. bei großen Graphgrößen $M$ auch niedriger) und einen zusätzlichen Effekt durch die endliche Iterationszahl.

Bei Iterationszahlen die etwa der Graphgröße $M$ entsprechen, wird der Agent für Short noch einmal deutlich besser.

## 5.5 Diskussion

Es konnten Minimalbeispiele für das Switching-Game mit Bridg-It-Graphen gefunden werden, bei denen die Shannon-Heuristik versagt. Für $M = 4$ kann ein nachziehender Gegner, der die Shannon-Strategie „durchschaut“, bei jedem Eröffnungszug gewinnen.

Im Self-Play versagt die Shannon-Heuristik ab $M = 9$, wobei die Siegquote des Startspielers allmählich fällt und vermutlich gegen 50 % konvergiert.

Dennoch können einfache Heuristiken helfen, komplexe Situationen zu analysieren. Anshelevich [8] verwendet ein erweitertes Widerstands-Modell (mit iterativem Hinzufügen virtueller Verbindungen) für das verwandte Spiel Hex. Sein Programm Hexy basiert auf der Auswertung des Gesamt-Widerstandes des Netzwerkes in Kombination mit automatischen Deduktionsregeln und ist mit seiner Alpha-Beta-Suche eines der stärksten Hex-Programme [7].

Bei der Analyse des zum Switching-Game sehr ähnlichen Spiels „Connections“[13] wurde gezeigt, dass die Argumentation von Lehman, d. h. der erste Spieler hat eine Ge-

[12] Selbst für eine feste Startlösung ist der Effekt auf einzelnen Kanten kaum vorherzusagen und somit mehr oder weniger pseudo-zufällig.

[13] Connections verfügt über eine zusätzliche lokalen Siegbedingung: Ein Spieler gewinnt auch durch die Erzeugung eines Zykluses in seinem Graphen.

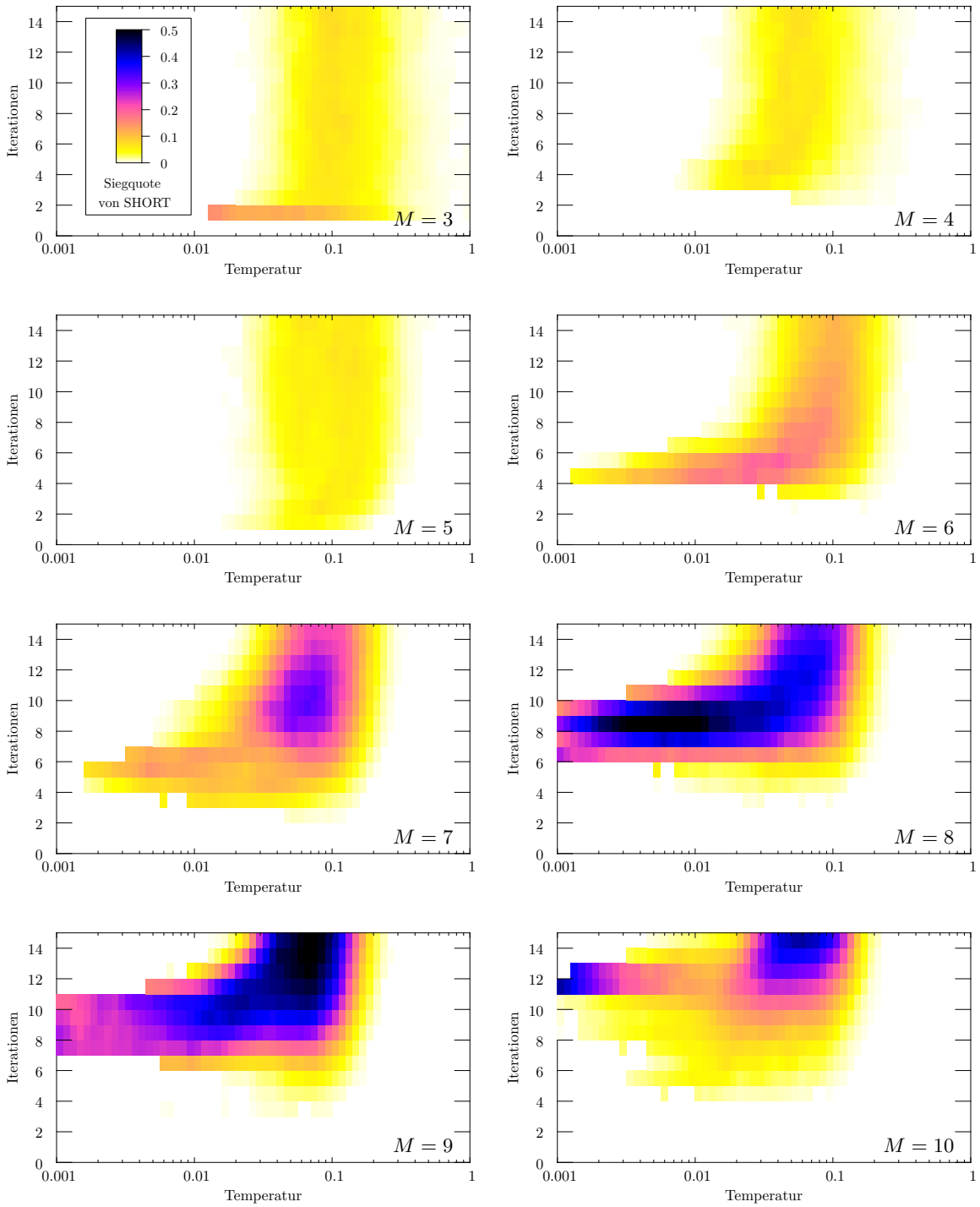



**Abbildung 5.11:** Switching-Game mit Shannon-Heuristik: Siegquote für den nachziehenden Spieler Short bei verschiedenen Graphgrößen $M = 2, \ldots, 10$. Short spielt mit Iterationsverfahren ($\omega = 1$)und Temperatur-Randomisierung.

winnstrategie, erhalten bleibt [42]. Sameith verwendet für Connections eine etwas andere Heuristik, nämlich einfach das Zählen der Anzahl der noch bis zum Sieg fehlenden Kanten. Im Netzwerk-Modell bedeutet dies die Berechnung kürzester Pfade. Mit einer $k$-best-Alpha-Beta-Suche führte diese recht simple Heuristik ebenso zu starken Computer-Agenten [43].

Bei der Untersuchung randomisierter Heuristiken konnten wir feststellen, dass ein bestimmter Anteil an Zufall die Shannon-Heuristiken insofern verbessern kann, dass der Anzugsvorteil deutlich reduziert wird. Dabei ist in jedem Fall das richtige Maß des Zufallsanteils wichtig. Bei zu geringer Randomisierung wird fast ausschließlich der gemäß Heuristik „beste“ Zug gespielt, während bei zu viel Zufall zu oft schlecht Züge gespielt werden.

Die Steuerung über die Temperatur nach Definition 5.7 erlaubt eine stetige Variation des Zufallseinflusses mit einer von den anderen Paramtern recht unabhängigen charakteristischen Temperatur.

Durch eine sinnvolle Wahl der Funktion $E$ aus (5.3) lässt sich das Modell auf nahezu jede Heuristik für Spiele bzw. Optimierungsprobleme übertragen.

## 5.6 Ausblick und offene Fragen

Das Konzept der künstlichen Randomisierung eröffnet ein weites Feld zukünftiger Forschungsthemen. Ein Anknüpfungspunkt ist der Vergleich mit den Ergebnissen aus [5] für Solitärspiele.

- Wie kann man die Randomisierung quantitativ als hilfreiche Vergrößerung des Suchraumes interpretieren?
- Wie übertragen sich die Ergebnisse für andere Heuristiken (z. B. Zählen von Kanten oder Pfaden)?
- Lässt sich die Randomisierung mit dem Konzept der simulierten Abkühlung verbinden?

Ein anderer Bereich ist die Anwendung auf Heuristiken für Optimierungsprobleme. Dabei kann beispielsweise Randomisierung im Vergleich zu $k$-best-Algorithmen studiert werden.

Für die Shannon-Heuristik im Switching-Game gibt es noch einige Fragen, die z. B. im Rahmen von Semesterarbeiten untersucht werden können:

- Wie stark sind Monte-Carlo-Agenten für größere Instanzen?
- Welchen Einfluss hat eine zufällig gewählte Startlösung (anstelle von $x^{(0)} = \frac{1}{2}$) auf die Ergebnisse?
- Wie viele Gauß-Seidel-Iterationen sind optimal?
- Wie ist der Überrelaxations-Parameter $\omega$ optimal zu wählen und wie viele Iterationen lassen sich dadurch einsparen?

# Anhang A

# Deduktionsspiele

## A.1 Beweis eines Lemmas zum Wurzel-Modell

Im Beweis von Satz 4.10 zur Abschätzung der Suchtiefe im Wurzel-Modell wurde folgende Ungleichung verwendet:

**Lemma A.1.** *Sei $N \geq 16$ und $\frac{1}{2}\sqrt{N} \leq x \leq \sqrt{N}$. Dann gilt*

$$1 + \frac{2}{N}\left(x\sqrt{x} + (N-x)\sqrt{N-x}\right) \leq 2\sqrt{N}\,.$$

**Beweis**. Wir teilen die Gleichung durch $2\sqrt{N}$ und führen die Substitution $y := \frac{x}{N}$ durch. Die Behauptung ist dann äquivalent zu

$$\frac{1}{2\sqrt{N}} + \underbrace{y\sqrt{y} + (1-y)\sqrt{1-y}}_{=:\,f(y)} \leq 1\,,$$

wobei nach Voraussetzung $\frac{1}{2\sqrt{N}} \leq y \leq \frac{1}{\sqrt{N}}$ gilt.

Da insbesondere $0 \leq y \leq 1$ gilt, betrachten wir die Funktion $f(y)$ im Intervall $[0,1]$. Sie ist stetig differenzierbar und konvex, wie man an der zweiten Ableitung $f''(y) = \frac{3}{4}y^{-1/2} + \frac{3}{4}(1-y)^{-1/2}$ sieht. Das Minimum liegt bei $y = \frac{1}{2}$ (Symmetriebetrachtung bzw. Nullsetzen der ersten Ableitung). Wegen $0 < \frac{1}{2\sqrt{N}} \leq y \leq \frac{1}{\sqrt{N}} < \frac{1}{2}$ nimmt $f(y)$ am linken Intervallrand $\frac{1}{2\sqrt{N}}$ ihren maximalen Wert an. Damit bleibt also noch

$$\frac{1}{2\sqrt{N}} + f\left(\frac{1}{2\sqrt{N}}\right) \leq 1$$

zu zeigen.

Mit der Substitution $\tilde{y} = \frac{1}{2\sqrt{N}}$ lautet die Behauptung schließlich

$$\tilde{y} + f(\tilde{y}) \leq 1\,, \tag{A.1}$$

mit $0 < \tilde{y} \leq \frac{1}{2\sqrt{16}} = \frac{1}{8}$ wegen der Voraussetzung $N \geq 16$.

Die Ungleichung (A.1) führen wir nun durch äquivalente Umformungen auf eine wahre Aussage zurück (man beachte, dass stets nur nichtnegative Ausdrücke quadriert werden):

$$
\begin{aligned}
& & f(\tilde{y}) &\leq 1-\tilde{y} & \\
\Longleftrightarrow & & \tilde{y}\sqrt{\tilde{y}}+(1-\tilde{y})\sqrt{1-\tilde{y}} &\leq 1-\tilde{y} & \quad |^2 \\
\Longleftrightarrow & & \tilde{y}^3+2\tilde{y}(1-\tilde{y})\sqrt{\tilde{y}}\sqrt{1-\tilde{y}}+(1-\tilde{y})^3 &\leq 1-2\tilde{y}+\tilde{y}^2 & \\
\Longleftrightarrow & & 2\tilde{y}(1-\tilde{y})\sqrt{\tilde{y}}\sqrt{1-\tilde{y}} &\leq 1-2\tilde{y}+\tilde{y}^2-\tilde{y}^3-(1-3\tilde{y}+3\tilde{y}^2-\tilde{y}^3) & \\
\Longleftrightarrow & & 2(1-\tilde{y})\sqrt{\tilde{y}}\sqrt{1-\tilde{y}} &\leq 1-2\tilde{y} & \quad |^2 \\
\Longleftrightarrow & & 4(1-2\tilde{y}+\tilde{y}^2)\tilde{y}(1-\tilde{y}) &\leq 1-4\tilde{y}+4\tilde{y}^2 & \\
\Longleftrightarrow & & 4\tilde{y}-12\tilde{y}^2+12\tilde{y}^3-4\tilde{y}^4 &\leq 1-4\tilde{y}+4\tilde{y}^2 & \\
\Longleftrightarrow & & 0 &\leq 1-8\tilde{y}+16\tilde{y}^2-12\tilde{y}^3+4\tilde{y}^4 & \\
\Longleftrightarrow & & 0 &\leq (1-8\tilde{y})+4\tilde{y}^2(4-3\tilde{y})+4\tilde{y}^4\,. &
\end{aligned}
$$

Die letzte Ungleichung ist wegen $0 < \tilde{y} \leq \frac{1}{8}$ erfüllt. □

## A.2 Beispiel für den Spielbeginn von Beams

Wir betrachten den in Abbildung A.1 dargestellten Spielverlauf. Die erste Anfrage auf Diagonale 17 mit Farbe rot liefert zwei Treffer. Diese beiden müssen mittels der vier verbleibenden roten Steine verbunden sein, jedoch gibt es hierfür noch mehrere Möglichkeiten. Die zweite Anfrage auf Diagonale 3 mit Farbe gelb gibt drei Treffer. Eine Verbindung dieser Steine kann nur über die beiden in Diagonale 2 eingezeichneten Steine erfolgen. Durch die neue Situation ergeben sich weitere Einschränkungen für die Verbindung der roten Steine, woraus folgt, dass das letzte Feld der Diagonale 3 rot sein muss.

Durch weitere Überlegungen kann man schlussfolgern, dass die fünf durch kleine Kreise markierten Felder dieselbe Farben haben müssen. Damit lässt sich per Widerspruchsbeweis weiterhin zeigen, dass auch das zweite Felder der Diagonale 16 rot sein muss.

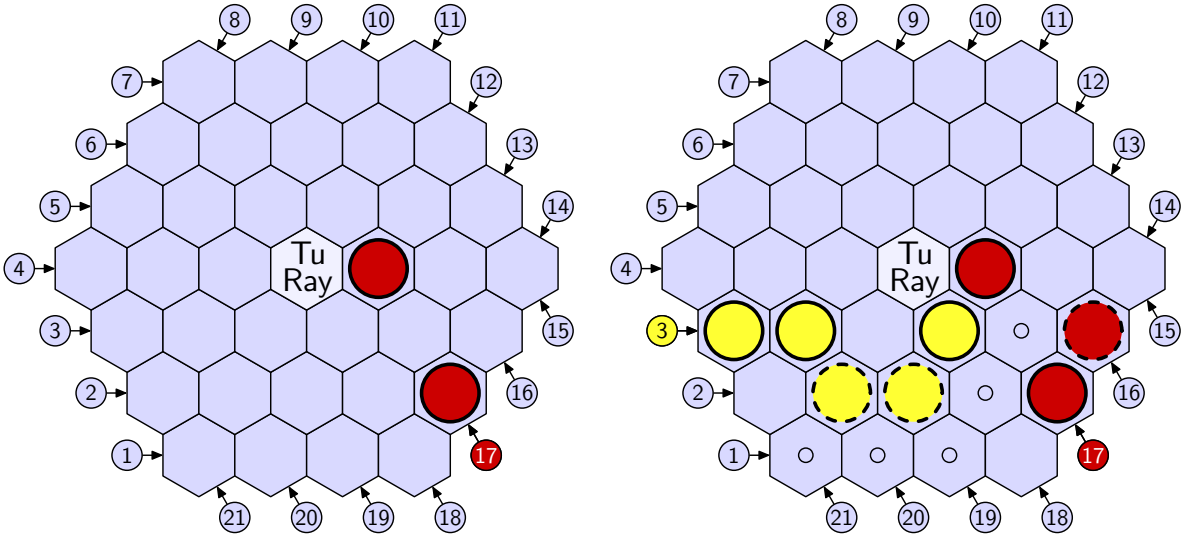



**Abbildung A.1:** Beispiel zu Beams: Informationen nach der ersten und zweiten Anfrage.

## A.3 Beams: Offizielle Spielregeln und Varianten von Michail Antonow

# BEAMS

Ein herausforderndes Denkspiel für 2–3 Personen ab 8 Jahre
(mit zwei Sets bis 5 Personen)
von Michail Antonow
Copyright 1996/2005

BEAMS ist ein spannendes Ratespiel, bei dem Glück und Zufall eine gewisse Rolle spielen, logisches Denken aber entscheidend zum Gewinn führt. Das Ziel: das geheime Farbenmuster des Gegners zu enthüllen – mit so wenigen Fragen wie nur möglich.

### Das Spielmaterial

(Standardbox ca. 16 x 16 cm für die **6 x 6** und die **5 x 6** Version):

- Zwei faltbare Sichtblenden,
- Vier Spielplatten **6 x 6** mit Quadraten auf der einen und Sechsecken auf der anderen Seite,
- Vier Spielplatten **5 x 6** mit Quadraten auf der einen und Sechsecken auf der anderen Seite,
- Zwei Zipp-Tüten mit je 6 Scheiben (18 mm) in 6 Farben mit Punkten auf der einen Seite zum Lösen der Aufgabe,
- Eine kleine Schachtel mit je 16 Marker (10 mm) in 6 Farben.
- Zwei Zipp-Tüten mit je 6 Chips (15 mm) in 6 Farben zum kreativen Gestalten des Farbenmusters,

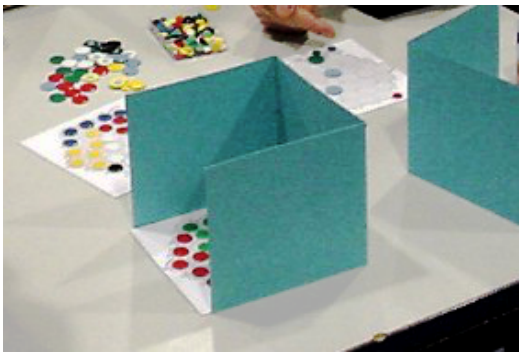


**Abbildung A.2:** Beams. (Foto: I. Althöfer, Lange Nacht der Wissenschaften, Jena 2009)

(Falls der Verlag sich nur für die **6 x 6** Version entscheidet, entfallen die vier **5 x 6** Spielplatten.

Wenn nur die **5 x 6** Version gewählt wird, entfallen die vier **6 x 6** Spielplatten, sowie alle 12 Chips, 12 Scheiben und 16 Marker einer Farbe.)

## Vorbereitung

Die Spieler einigen sich ob sie die quadratische oder die sechseckige Variante und dann ob sie die **5 x 6 (BEAMS 30)** oder die **6 x 6 (BEAMS 36)** Version spielen wollen.

Bei zwei Spielern: die Spieler sitzen sich gegenüber, die kleine flache Schachtel mit den Markern offen in Reichweite in der Mitte des Tisches. Jeder legt zwei Spielplatten der gewählten Variante flach vor sich. Eine der Platten wird mit Hilfe einer Sichtblende vor den Blicken des Gegenspielers abgeschirmt. Die Spieler erhalten je eine Zipp-Tüte mit Chips und mit Scheiben. Für die **5 x 6** Variante werden die Chips und Scheiben einer Farbe aussortiert.

## Und so wird gespielt

Zuerst bildet jeder Spieler auf seiner abgeschirmten Spielplatte ein Farbenmuster, indem er alle 30, bzw. 36 Chips auf die Felder verteilt. Eine **einzige Bedingung** ist dabei zu beachten: alle sechs Felder, die mit Chips einer Farbe belegt sind, müssen zusammenhängen.

Beachten Sie: Auf dem quadratischen Brett gelten auch die übereck angrenzenden Felder als zusammenhängend. Das bedeutet, dass Ketten unterschiedlicher Farbe sich überkreuzen können. Auf dem Sechseckbrett können sich verschiedenfarbige Ketten niemals überkreuzen.

Beachten Sie: Auf dem **6 x 6** Sechseckbrett bleibt das Feld im Zentrum unbesetzt. Eine Kette einer Farbe muss um dieses Feld herumgeführt werden, um als zusammenhängend zu gelten.

Die Spieler versuchen nun, das geheime Farbenmuster des Gegners mit so wenigen Fragen wir nur möglich zu erraten.

Stellen Sie sich vor, das Spielbrett sei so etwas wie eine geschlossene Black Box – mit winzigen Fenstern rundherum, durch die Sie einen Lichtstrahl hineinschicken können (daher der Name des Spiels). Jedes Fenster hat eine eigene Nummer. Stellen Sie sich weiter vor, so ein Lichtstrahl ist nur für Gegenstände einer bestimmten Farbe empfindlich. Wenn Sie also einen „gelben Strahl“ durch ein Fenster hineinschicken, wird er nur alle gelben Gegenstände auf seinem Weg beleuchten und alles andere ignorieren.

Spieler A beginnt indem er durch ein beliebiges Fenster einen Strahl beliebiger Farbe hineinschickt. Er kann z. B. sagen: „14 grün?“ Das heißt, er hat durch das Fenster Nr. 14 einen Strahl senkrecht heruntergeschickt, der nach grünen Gegenständen sucht. Ein Strahl durchquert das ganze Brett, immer ausgehend vom genannten Fenster. Die Felder, die er passiert, werden immer vom Einfallfenster aus gezählt. (Das leere Feld in der Mitte des **6 x 6** Sechseckbretts wird dabei nicht gezählt, die Strahlen überspringen es einfach.)

Auf die Frage von Spieler A – in unserem Beispiel: „14 grün“ – muss Spieler B nun wahrheitsgemäß antworten. Er sagt: „Kein Treffer“, wenn sich keine grünen Chips auf dem Wege des Strahls befinden. Oder er sagt. „Treffer auf Felder zwei und vier“, wenn

auf dem zweiten und dem vierten Feld (gezählt vom Fenster Nr. 14 aus) grüne Chips liegen.

Jedes mal, wenn ein Strahl durch ein Fenster geschickt wird, setzen beide Spieler je einen Marker dieser Farbe auf das genannte Fenster: der fragende Spieler auf sein offenes Spielbrett, und der antwortende Spieler auf sein verdecktes Brett. So wissen beide Spieler welche Fenster benutzt worden sind.

Wichtig: Jedes Fenster kann nur einmal benutzt werden.

Zurück zum Beispiel: Auf die Antwort „14 grün: Treffer auf Felder zwei und vier“ nimmt Spieler A zwei grüne Scheiben und legt sie auf das zweite und auf das vierte Feld, ausgehend vom Fenster Nr. 14, mit der vollfarbigen Seite nach oben. Das ist gesicherte Information: für die gesamte Dauer des Spieles kann er sicher sein, dass diese zwei Felder grün sind, und ebenso sicher sein, dass alle anderen Felder auf dieser Strecke nicht grün sind.

So wechseln sich die Spieler mit Fragen und Antworten ab. Sie können frei entscheiden welches Fenster sie benutzen und nach welcher Farbe Sie fragen wollen. Dabei ist darauf zu achten, dass jede Information festgehalten wird indem man Marker auf die benutzen Fenster setzt und Scheiben mit der vollflächigen Seite nach oben auf die gemeldeten Felder legt.

> **Wichtig:** Alle Antworten müssen absolut korrekt sein – denn BEAMS ist ein Spiel der reinen Logik. Eine einzige falsche Auskunft, eine undeutlich ausgesprochene Zahl, oder auch nur eine falsch gelegte oder verrutschte Scheibe kann das ganze Spiel verderben, wenn sie nicht umgehend korrigiert wird. Achten Sie darauf, dass alle Angaben stimmen und auch richtig verstanden werden. Wenn eine falsche Information nicht mehr behoben werden kann, erhält der dafür verantwortliche Spieler zehn Strafpunkte.

Nach einigen Zügen werden die Spieler anfangen, mehr oder weniger begründete Überlegungen anzustellen, wie die verbleibenden Scheiben auf die noch offenen Feldern verteilt sein könnten. Glaubt ein Spieler, seine Vermutung könnte stimmen, legt er Scheiben der vermuteten Farbe auf die entsprechenden Felder – mit dem Farbpunkt nach oben.

> **Nicht vergessen:** vollflächige Seite nach oben = sichere Information
> Seite mit dem Punkt nach oben = Vermutung, Spekulation.

Vermutungen können nachgeprüft werden, indem man weitere Fenster benutzt um zusätzliche Fragen zu stellen. Beachten Sie, dass jedes Feld auf dem sechseckigen Spielbrett von drei Seiten angepeilt werden kann. Auf dem quadratischen Spielbrett kann jedes Feld sogar von vier Seiten abgefragt werden.

In vielen Fällen kann eine Vermutung allein durch logisches Denken bestätigt oder widerlegt werden. So vermeiden Sie überflüssige Fragen, die Ihr Spielergebnis belasten.

Irgendwann wird einer der Spieler überzeugt sein, das komplette Farbenmuster des Gegners entschlüsselt zu haben. Er erklärt: „Ich bin fertig.“ Für ihn ist das Spiel damit beendet. Jetzt muss er nur noch die Fragen seines Spielpartners beantworten, bis auch dieser glaubt, die komplette Lösung gefunden zu haben. Nun beseitigt man die Sichtblenden und überprüft die Ergebnisse.

### Wie wird abgerechnet

Für jede gestellte Frage ( = benutztes Fenster) erhält der Spieler einen Minuspunkt. Dazu kommt ein Minuspunkt für jedes Feld, das er mit einer falschen Farbe belegt hat. Gewonnen hat der Spieler mit der niedrigsten Zahl. Werden mehrere Partien gespielt, addiert man die Punkte. (Je mehr Partien gespielt werden, umso geringer wirkt sich der Glücksfaktor aus.)

### BEAMS für mehrere Spieler

Auch drei Personen können BEAMS mit einer Standard-Schachtel spielen. Spieler A ist in der ersten Runde der Aufgabensteller, der das geheime Farbenmuster mit einem Satz Chips erstellt. Spieler B und C stellen abwechselnd laut Fragen, die A ebenso laut (also für beide) beantwortet. B und C verwerten diese Informationen aber jeder für sich hinter seiner Sichtblende. Wenn einer von beiden meint, fertig zu sein, zeigt er seinen Lösungsvorschlag zur Kontrolle verdeckt an A, der das Ergebnis festhält. Wenn auch der zweite Ratende fertig ist, wird auch sein Ergebnis notiert. In der zweiten und dritten Runde übernehmen dann B und C die Rolle des Aufgabenstellers.

Mit zwei Standard-Schachteln kann BEAMS von bis zu fünf Personen gespielt werden: einer ist Aufgabensteller und die anderen stellen reihum Fragen, die für alle gelten, die aber jeder für sich getrennt verarbeitet. Das Spiel ist zu Ende wenn jeder einmal Aufgabensteller gewesen ist.

BEAMS kann übrigens auch von einer Person allein gespielt werden. Sollte Ihr Partner nicht in der Stimmung zum Aufgabenlösen sein, würde es genügen, wenn er ein geheimes Farbenmuster für Sie erstellt und Ihre Fragen danach beantwortet.

### BEAMS Puzzles

Eine besondere Herausforderung für Spieler mit etwas Erfahrung sind die BEAMS Puzzles, die für jede Version komponiert werden können (eine kleine Musterliste liegt bei.)

# Anhang B

# Codes

## B.1 Vorbemerkungen

Die Codes sind in Fortran-95 geschrieben. Dieser Fortran-Dialekt stellt (trotz nahezu vollständiger Abwärtskompatibilität) gegenüber Fortran-77 eine moderne modularisierte Programmiersprache dar. Die Syntax ist sehr einfach und leicht verständlich. Auf einige – für das Verständnis der folgenden Quelltexte wichtige – Besonderheiten sei hier kurz eingegangen.

**Besondere Zeichen:**

- `!` leitet Kommentare ein
- `;` trennt mehrere Befehle in einer Zeile
- `**` Potenz: `x**3`$=x^3$
- `%` Typkomponente: `x%Start`$=x_{\text{Start}}$
- `:` Indexbereich (s. u.)

**Variablendeklaration:** Variablen haben einen Typ und optional Attribute. Die Attribute geben z. B. an, ob es sich um Felder (**`dimension(...)`**), dynamische Variablen (**`allocatable`**) oder Zeiger (**`pointer`**) handelt. Man muss sich nicht selbst um Dereferenzierung kümmern, was viele Fehlerquellen vermeidet. Den Werte-Bereich der Datentypen **`integer`** und **`real`** kann man ebenso spezifizieren (**`kind`**-Attribut) bzw. global durch Compiler-Optionen vorgeben.

Bsp: **`real,dimension`**`(4,3)::x,y` deklariert zwei 4×3-Felder

**Indexkonzept und der Doppelpunkt:** Feldindizes beginnen standardmäßig bei 1. D. h. **`real,dimension`**`(`$n$`)::x` deklariert `x(1)` bis `x(`$n$`)`. Dagegen startet **`dimension`**`(0:`$n$`)` bei `x(0)`.

Der Doppelpunkt beschreibt einen Indexbereich. Man kann sogar die Grenzen weglassen, wenn man sie (noch) nicht kennt oder das ganze Feld ansprechen möchte. **`dimension`**`(:,1:3)` ist also ein ($n$×3)-Feld, $n$ beliebig, zur Verwendung bei Parametern von **`subroutine`** bzw. **`function`** oder bei dynamischen Feldern.

Beim Zugriff auf Feldvariablen kann man auch „:“ verwenden oder die Indizes ganz weglassen, denn viele Operationen lassen sich auch mit ganzen Feldern ausführen. Bsp.: `x(1:10)=0` ⇔ `x(:)=0` ⇔ `x=0` ⇏ `x(10)=0`.

**Zähl-Schleifen:** `do i=a,b,c; ... ; end do` durchläuft die ganzen Zahlen von $a$ bis $b$, wobei optional ein Inkrement $c$ angegeben werden kann (auch negativ).

**Logische Ausdrucke:** `==` (gleich), `/=` (ungleich) `<`, `<=`, `>`, `>=` (Vergleich), `.not.` (nicht), `.and.` (und), `.or.` (oder), man beachte die Punkte vor und nach den Schlüsselwörtern.

**Parameterübergabe:** Variablen werden an Funktionen und Unterprogramm grundsätzlich per Referenz übergeben (call by reference).

Der Rückgabewert einer Funktion wird wie eine normale Variable deklariert und behandelt.

Für Fortran stehen mehrere leistungsfähige Compiler zur Verfügung. Typischerweise lautet der Aufruf des freien GNU-Compilers

`gfortran -O3 -fimplicit-none -fdefault-real-8` *Quelle(n)*`.f95`

Im Schnitt sind Fortran-Programme ca. 30% langsamer als C-Programme, dafür aber deutlich weniger fehlerträchtig. Im Vergleich zu Interpretersprachen (z. B. Matlab), ergibt sich (für elementare Befehle) trotzdem ein Geschwindigkeitsvorteil von ca. Faktor 50–100.

## B.2 Siegwahrscheinlichkeiten für MC-Analyse

Die folgende Routine ist die Implementierung des in Abschnitt 2.4.6 skizzierten Algorithmus' zur schnellen Berechnung der Zugauswahl-Wahrscheinlichkeiten bei reinen Monte-Carlo-Agenten. Ab Zeile 13 werden die Binomialkoeffizienten berechnet. Dies ist zumindest für große Versuchszahlen $N$ nicht unproblematisch. Deshalb wird zunächst der Logarithmus der Koeffizienten berechnet:

$$\begin{aligned}\log b_N(j,p) &= \log \binom{N}{j} p^j (1-p)^{N-j} \\ &= \underbrace{j \log p + (N-j)\log(1-p)}_{=\texttt{lbn1}} + \underbrace{\sum_{k=1}^{j} (\log(N-k+1) - \log k)}_{=\texttt{lbn2}}\end{aligned}$$

Für sehr große $N$ ($N \gg 10^6$) treten bei der Summation (besonders durch die Differenzbildung in Zeile 28) Rundungsfehler auf. In Zeile 32 wird deshalb nachskaliert um zu erreichen das Summe der Binomialkoeffizienten möglichst genau den Wert 1 hat.

Ab Zeile 34 werden die Wahrscheinlichkeiten gemäß dem skizzierten Algorithmus berechnet. Die Variable `w` entspricht dabei der Wahrscheinlichkeit $p^+$, dass der aktuelle Zugkandidat besser als alle zuvor betrachteten Zugkandidaten ist.

```
1  ! Schnelle Berechnung der Maxima mehrerer Binomialverteilungen
2  subroutine binomdist_fast(n,m,p,x)
3      integer::n                      ! Zahl der Versuche
4      integer::m                      ! Zahl der Binomialverteilungen
5      real,dimension(:)::p            ! WSK-Parameter der Binomialverteilungen
6      real,dimension(:)::x            ! Rueckgabe: WSK fuer Auswahl eines Zuges
7    integer::i,j,k,t
8    real,dimension(0:n,m)::bn         ! Binomialkoeffizienten bn(j,k)=b_n(j,p_k)
9    real,dimension(m)::bs             ! bs(j,k)=Summe(i=0..j,bn(i,k))
10   real,dimension(m,m)::xm           ! xm(i,k)=WSK, dass Zug i bei insgesamt k
11                                     ! Siegern ausgewaehlt wird (Zahl der Siege fest)
12   real::lp,lq,lbn1,lbn2,w,v,vs,pt
13   ! Initialisiere Binomialverteilungen bn
14   do k=1,m
15     if (p(k)<eps) then              ! p_m=0 (eps = 10^-14)
16       bn(0,k)=1.0
17       bn(1:n,k)=0.0
18     else if (p(k)>1-eps) then       ! p_m=1
19       bn(0:n-1,k)=0.0
20       bn(n,k)=1.0
21     else                            ! 0<p_m<1 (echte Binomialverteilung)
22       lp=log(p(k)); lq=log(1-p(k))
23       lbn1=n*lq                     ! lbn1+lbn2=log(bn(j,k)) (wg. Stabilitaet)
24       lbn2=0.0
25       bn(0,k)=exp(lbn1+lbn2)
26       do j=1,n
27         lbn1=j*lp+(n-j)*lq
28         lbn2=lbn2+log(n-j+1.0)-log(1.0*j)
29         bn(j,k)=exp(lbn1+lbn2)
30       end do
31     end if
32     bn(:,k)=bn(:,k)/sum(bn(:,k))    ! Nachskalierung (zusaetzliche Stabilitaet)
33   end do
34   ! Berechne sukzessive die Wahrscheinlichkeiten aller Zugmoeglichkeiten
35   x=0
36   bs=0
37   do j=0,n                          ! alle moeglichen Werte den Maximums
38     xm=0.0
39     w=1.0
40     do i=1,m                        ! sukzessive die Zugkandidaten hinzufuegen
41       xm(i,1)=w*bn(j,i)             ! neuer Zug i besser als alle andere Zuege
42       w=w*bs(i)
43       do t=i-1,1,-1                 ! alle Anzahlen von Mehrfachsiegern
44         pt=bn(j,i)/(t+1)            ! normierte Tie-Break-WSK
45         vs=0.0
46         do k=1,i-1                  ! alle bisher betrachteten Zuege
47           v=xm(k,t)
48           xm(k,t)=bs(i)*v           ! neuer Zug i schlechter
```

```
49          xm(k,t+1)=xm(k,t+1)+pt*v*t ! k-Anteil von Tie-Break mit Zug i
50          vs=vs+v
51        end do
52        xm(i,t+1)=xm(i,t+1)+pt*vs  ! i-Anteil von Tie-Break mit Zug i
53      end do
54    end do
55    do i=1,m                       ! WSKen der Tie-Breaks summieren
56      do t=1,m
57        x(i)=x(i)+xm(i,t)
58      end do
59      bs(i)=bs(i)+bn(j,i)
60    end do
61  end do
62 end subroutine
```

## B.3 Exakte Analyse von Subtraktionsspielen

Das folgende Programm führt die in Abschnitt 2.4 beschriebene exakte Analyse von Self-Play-Experimenten mit Monte-Carlo-Agenten durch.

Aus Übersichtlichkeitsgründen ist hier nur der Code für Subtraktionsspiele (vgl. Abschnitt 2.5.2) abgedruckt. Bei diesen ist der Konfigurationsraum besonders einfach, denn es ist nur die Haufengröße und der Spieler am Zug anzugeben. Das Programm kann beliebige Mehrzug-Subtraktionsspiele analysieren, die hier abgedruckte interaktive Eingabe ist jedoch nur für Zwei-Zug-Subtraktionsspiele ausgelegt.

Das Modul `binomdist_mod` (Zeile 3) enthält das Unterprogramm `binomdist_fast` aus Abschnitt B.2.

```
1  ! Exakte Analyse von MC-Self-Play bei Subtraktionsspielen
2  program subtraktionsspiel
3    use binomdist_mod                  ! Enthaelt die Routine binomdist_fast
4    integer::n,mc1,mc2
5    real::r1,r2
6    real,dimension(:),allocatable::rnd ! Siegquoten f. Spieler am Zug
7                                       ! bei Zufallsspiel
8    real,dimension(:,:),allocatable::x ! Siegquoten f. Spieler 1 bei MC-Spiel
9    integer,dimension(2)::s            ! Zugmenge (die Analyse funktioniert
10                                      ! auch fuer mehr als 2 Zugmoeglichkeiten)
11   print *,"Zwei-Zug-Subtraktionsspiel␣S[s1,s2](n)␣fuer␣MC(m1)␣vs.␣MC(m2)."
12   print *,"Starthaufengroesse␣n=";       read *,n;
13   print *,"Zugmoeglichkeit␣1:␣s1=";      read *,s(1);
14   print *,"Zugmoeglichkeit␣2:␣s2=";      read *,s(2);
15   print *,"MC-Paramert␣Spieler␣1:␣mc1="; read *,mc1;
16   print *,"MC-Paramert␣Spieler␣2:␣mc2="; read *,mc2;
17
18   call rnd_init(n,s,rnd)            ! Siegquoten fuer Zufallspartien berechnen
19   call mc_calc(n,s,rnd,mc1,mc2,x)   ! Monte-Carlo analysieren
20   r1=x(1,n); r2=x(2,n)              ! Ergebnisse extrahieren
```

```
21   print *,"Sieqquote␣im␣Hinspiel␣␣:",r1
22   print *,"Sieqquote␣im␣Rueckspiel:",r2
23   print *,"Sieqquote␣gemittelt:␣␣␣",(r1+r2)/2
24
25 contains
26
27 subroutine rnd_init(n,s,rnd)        ! rnd initialisieren
28     integer::n                      ! max. Haufengroesse
29     integer,dimension(:)::s         ! Zugmenge
30     real,dimension(:),allocatable::rnd ! Siegquote fuer Spieler am Zug
31                                        !              bei Zufallsspiel
32                                     ! rnd(h): h=akt. Haufengroeoesse
33   integer::i,z,zn,sk
34   real::r
35   sk=size(s)                        ! Anzahl moeglicher Zuege
36   if (allocated(rnd)) deallocate(rnd)
37   allocate(rnd(n))
38   do z=1,n                          ! Bewerte alle Haufengroessen z
39     r=0
40     do i=1,sk                       ! Mittelwert ueber alle Zugmoeglichkeiten
41       zn=z-s(i)
42       if (zn>0) r=r+rnd(zn)         ! fuer zn<=0 waere rnd(zn)=0
43     end do
44     rnd(z)=1-r/sk
45   end do
46 end subroutine
47
48 subroutine mc_calc(n,s,rnd,mc1,mc2,x)  ! Monte-Carlo analysieren
49     integer::n                        ! max. Haufengroesse
50     integer,dimension(:)::s           ! Zugmenge
51     real,dimension(:),allocatable::rnd ! Siegquote fuer Spieler 1 bei
52                                       ! Zufallsspiel
53     integer::mc1,mc2                  ! Monte-Carlo-Parameter der Spieler
54     real,dimension(:,:),allocatable::x ! Siegquoten f. Spieler 1 bei MC-Spiel
55                                       ! x(p,h): p=Spieler am Zug,
56                                       !         h=akt. Haufengroesse
57   integer::i,z,zn,sk
58   real,dimension(:),allocatable::pr,px ! WSK-Vektor fuer Zugmoeglichkeiten
59   real::r
60   sk=size(s)                          ! Anzahl moeglicher Zuege
61   if (allocated(x)) deallocate(x)
62   allocate(x(2,n))
63   allocate(pr(sk))
64   allocate(px(sk))
65   do z=1,n                            ! Bewerte alle Haufengroessen z
66     do i=1,sk                         ! Siegquoten bei Zufallsspiel
67       zn=z-s(i)
68       if (zn>0) then
69         pr(i)=1.0-rnd(zn)
70       else
71         pr(i)=1.0
72       end if
```

```
    end do
    ! Spieler 1 am Zug
    call binomdist_fast(mc1,sk,pr,px)  ! MC analysieren
    r=0.0
    do i=1,sk
      zn=z-s(i)
      if (zn>0) then
        r=r+px(i)*x(2,zn)
      else
        r=r+px(i)*1.0
      end if
    end do
    x(1,z)=r
    ! Spieler 2 am Zug
    call binomdist_fast(mc2,sk,pr,px)  ! MC analysieren
    r=0.0
    do i=1,sk
      zn=z-s(i)
      if (zn>0) then
        r=r+px(i)*x(1,zn)
      else
        r=r+px(i)*0.0
      end if
    end do
    x(2,z)=r
  end do
  deallocate(pr)
  deallocate(px)
end subroutine

end program
```

# Quellen- und Literaturverzeichnis


[1] Abramowitz, M., Stegun, I.: *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover Publications, 1964.
http://apps.nrbook.com/abramowitz_and_stegun/index.html
http://dlmf.nist.gov

[2] Ahlswede, R., Wegener, I.: *Suchprobleme.* B. G. Teubner, 1979.

[3] Althöfer, I.: *Computer-aided Game Inventing.* Technical Report, Friedrich-Schiller-Universität Jena, 2003.
http://www.minet.uni-jena.de/preprints/althoefer_03/CAGI.pdf

[4] Althöfer, I.: *On the Laziness of Monte-Carlo Game Tree Search in Non-tight Situations.* Technical Report, Friedrich-Schiller-Universität Jena, 2008.
http://www.althofer.de/mc-laziness.pdf

[5] Althöfer, I., Heuser, S. *Randomised Evaluations in Single-Agent Search.* ICGA Journal, Vol. 28 (2005) No. 1, S. 21–31.
http://www.minet.uni-jena.de/preprints/althoefer_05/altheuser.pdf

[6] Althöfer, I.: *Game Self-Play with Pure Monte-Carlo: The Basin Structure.* Eingereicht bei IEEE Transactions on Computational Intelligence and AI in Games, 2010.
http://www.althofer.de/monte-carlo-basins-althoefer.pdf

[7] Anshelevich, V. V.: *Hexy plays Hex.* 2000.
http://home.earthlink.net/~vanshel

[8] Anshelevich, V. V.: *A Hierarchical Approach to Computer Hex.* Artificial Intelligence **134**, 2002.

[9] Antonow, M.: *Deduktionsspiel Beams (1995).* Persönliche Gespräche 2009 und 2010.

[10] Auer, P., Cesa-Bianchi, N., Fischer, P.: *Finite-time Analysis of the Multiarmed Bandit Problem.* Journal Machine Learning **47** (2–3), 2002.

[11] Berghmana, L., Goossensa, D., Leus, R.: *Efficient solutions for Mastermind using genetic algorithms.* Computers & Operations Research, Vol. 36 (2009), S. 1880–1885.

[12] Berlekamp, E.,R., Conway, J. H., Guy, R. K.: *Winning Ways for Your Mathematical Plays.* Vol. 1–2, 2nd edition, AK Peters, Ltd., 2001/03.

[13] Bewersdorff, J.: *Glück, Logik und Bluff.* 3. Auflage, Vieweg, 2003.

[14] Browne, C.: *Connection Games: Variations on a Theme.* A K Peters, 2005.

[15] Browne, C.: *Automated Generation and Evaluation of Recombination Games.* Dissertation, Queensland University of Technology Brisbane, 2008.
http://www.cameronius.com/cv/publications/thesis-2.47.zip

[16] Brualdi, R. A.: *Introductory Combinatorics.* 5th ed, Prentice Hall, 2009.

[17] Brügmann, B.: *Monte Carlo Go.* Max-Planck-Institut für Physik, München, 1993.
http://www.althofer.de/Bruegmann-MonteCarloGo.pdf

[18] Chase, S. M.: *An Implemented Graph Algorithm for Winning Shannon Switching Games.* Comm. of the ACM, 15 No. 4, 1972.

[19] Chen, S. T., Lin, S. S.: *Optimal Algorithms for $2 \times n$ AB Games – A Graph-Partition Approach.* Journal of Information Science and Engineering, Vol. 20 (2004), S. 105–126.

[20] Cover, T. M., Thomas, J. A.: *Elements of Information Theory.* 2. Auflage, Wiley, 2006.

[21] Erdmann, J.: *The Characterization of Chance and Skill in Games.* Dissertation, Friedrich-Schiller-Universität Jena, 2010.
http://www.althofer.de/erdmann-doctoral-thesis.pdf
*Towards a Characterization of Chance in Games.* ICGA Journal, Vol. 32 (2009), S. 187–205.

[22] Fischer, T.: *Optimale Strategien für Mastermind.* Die Wurzel – Zeitschrift für Mathematik, Heft 3+4/2002, S. 52–57.

[23] Fischer, T.: *Bridg-It – Beating Shannon's Analog Heuristic.* Technical Reports on Optimization, 2009.
http://www.minet.uni-jena.de/Math-Net/reports/sources/2009/09-07report.pdf

[24] Fischer, T.: *Deduction Games and Rényi Entropy Strategies.* Eingereicht bei ICGA Journal, 2011.

[25] Flammenkamp, A.: *Lange Perioden in Subtraktions-Spielen.* Dissertation, Universität Bielefeld, 1996.

[26] Gardner, M.: *The Second Scientific American Book of Mathematical Puzzles and Diversions.* Simon and Schuster, 1961.

[27] Heinz, E. A.: *Scalable Search in Computer Chess.* Vieweg, 2000.

[28] Huang, L. T., Lin, S. S.: *Optimal Analyses for $3 \times n$ AB Games in the Worst Case.* Advances in Computer Games, Lecture Notes in Computer Science. Springer, 2010.

[29] Irving, R.: *Towards an Optimum Mastermind Strategy.* Journal of Recreational Mathematics, Vol. 11 (1978–79), S. 81–87.

[30] Kearfott, R. B.: *Interval Computations: Introduction, Uses, and Resources.* Euromath Bulletin **2** (1), 1996.
http://interval.louisiana.edu/preprints/survey.pdf

[31] Knuth, D.: *The Computer as Master Mind.* Journal of Recreational Mathematics, Vol. 9 (1976–1977), S. 1–6.

[32] Kooi, B.: *Yet Another Mastermind Strategy.* ICGA Journal, Vol. 28 (2005), S. 13–20.

[33] Koyoma, K. and Lai, T.: *An Optimal Mastermind Strategy.* Journal of Recreational Mathematics, Vol. 25 (1993), S. 251–256.

[34] Kuriakose, C. P.: *Circuit Theory: Continuous and Discrete Time Systems, Elements of Network Synthesis.* Prentice-Hall of India Pvt. Ltd., 2005.

[35] Lehman, A.: *A Solution to the Shannon Switching Game.* Jounal of the SIAM **12** No. 4, 1964.

[36] Maire, F.: *The solution of Shannon Game.* 2004. Erschienen im Anhang von [14].

[37] Mansfield, R.: *Strategies for the Shannon switching game.* 2000.
www.math.psu.edu/melvin/logic/shannon.pdf

[38] Pronzato, L, Wynn, H. P., Zhigljavsky, A. A.: *Using Rényi entropies to measure uncertainty in search problems.* In: Lectures in Applied Mathematics Vol. 33 (1996), Mathematics of Stochastic Manufacturing Systems (G. George Yin, Qing Zhang, Hrsg.).

[39] Reams, C.: *Playing perfect Hangman.* ICGA Journal, Vol. 32 (2009), S. 149–153.

[40] Reinhardt, L.: *Analyse von Monte-Carlo-Methoden bei Subtraktionsspielen.* Diplomarbeit, Friedrich-Schiller-Universität Jena, 2008.

[41] Rényi, A.: *On measures of information and entropy.* Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability, 1960, S. 547–561.
http://digitalassets.lib.berkeley.edu/math/ucb/text/math_s4_v1_article-27.pdf

[42] Sameith, J.: *Entscheidungs-Unterstützung im Multiple-Choice-Modus – Realisierung eines k-Best-Algorithmus für das Spiel Connections.* Diplomarbeit, Friedrich-Schiller-Universität Jena, Fakultät für Mathematik und Informatik, 2002.
http://www.joerg.sameith.net/sam02_dipl.pdf

[43] Sameith, J.: *Connect-It.* 2002.
http://www.joerg.sameith.net/denken_connect.html

[44] Schreiber, L.: *Mathematische Untersuchung des Deduktionsspiels Beams.* Diplomarbeit, Friedrich-Schiller-Universität Jena, Fakultät für Mathematik und Informatik, 2009.
http://www.althofer.de/diplomarbeit-schreiber.pdf

[45] Shannon, C. E.: *Collected Papers.* Herausgegeben von Sloane, J. A. und Wyner, N. A. D., Wiley-IEEE Press, 1993.

[46] Heinz-Nixdorf-Museums-Forum Paderborn: *Codes and Clowns: Claude Shannon – The Juggling Scientist.* Sonderausstellung vom 6. November 2009 bis 28. Februar 2010.
http://www.hnf.de/Shannon

[47] Heinz-Nixdorf-Museums-Forum Paderborn: *Ausstellung zum einhundertsten Geburtstag von Alan Turing.* Das Spiel „Beams“ wird unter dem Namen „Tu Ray“ die Ausstellung im Jahre 2012 begleiten.
http://en.hnf.de

# Ehrenwörtliche Erklärung

Hiermit erkläre ich:

- dass mir die geltende Promotionsordnung der Fakultät für Mathematik und Informatik der Friedrich-Schiller-Universität Jena bekannt ist,
- dass ich die Dissertation selbst angefertigt habe, keine Textabschnitte oder Ergebnisse eines Dritten oder eigener Prüfungsarbeiten ohne Kennzeichnung übernommen habe und alle von mir benutzten Hilfsmittel, persönliche Mitteilungen und Quellen in meiner Arbeit angegeben sind,
- dass alle Personen die mich bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts unterstützt in der Danksagung (Seite 4) genannt sind,
- dass die Hilfe eines Promotionsberaters nicht in Anspruch genommen wurde und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe und
- dass ich weder die gleiche, eine in wesentlichen Teilen ähnliche oder eine andere Abhandlung bei einer anderen Hochschule als Dissertation eingereicht habe.

Jena, den 9. Februar 2011