Dynamisches Programmieren und Brettspiele
Ein Überblick

Constantin Gaul

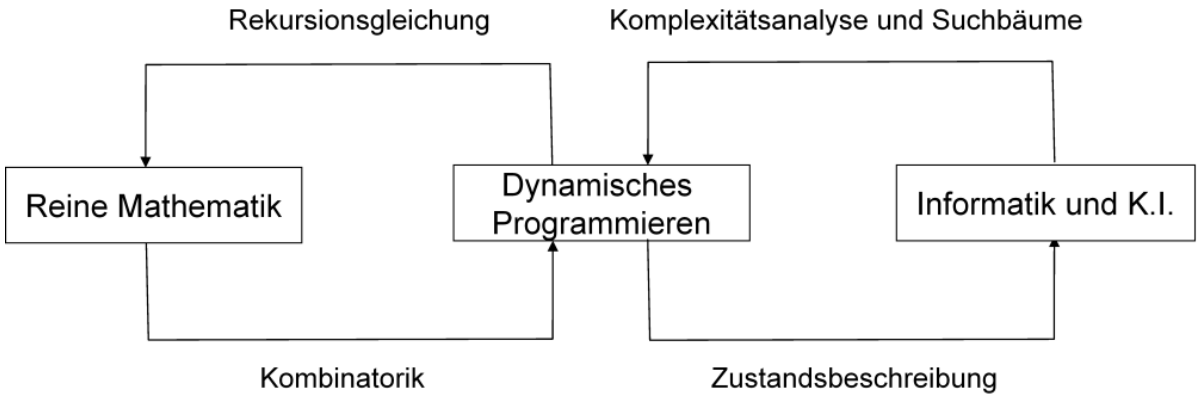
13. Februar 2006

1 Einführung

Dieser Vortrag soll einen kurzen Überblick über den Einsatz der Technik des Dynamischen Programmierens bei der Lösung von Problemstellungen, die sich bei „Brettspielen“ stellen, geben. Es werden Problemstellungen, Ideen, Konzepte und Ergebnisse dazu präsentiert.

1.1 Dynamisches Programmieren

Das Dynamische Programmieren ist ein Teilgebiet des Operation Research, das sich mit den Gebieten der reinen Mathematik und der Informatik überlappt. Beide Disziplinen haben Beiträge zur Dynamischen Programmierung geleistet und ebenfalls Konzepte daraus übernommen. Die Ideen des Dynamischen Programmierens wird hier als allgemein bekannt vorausgesetzt.



1.2 „Brettspiele“

Der Begriff „Brettspiele“ steht hier für eine Klasse von Spielen, darunter auch solche die mit Stift und Papier gespielt werden. Charakterisierend für diese Klasse sind dabei folgende Eigenschaften:

- Ein oder zwei Spieler.
- Spieler befolgt (befolgen) Regeln die den Zustand des Spiels ändern.
- Spieler hat eine Entscheidungen zu treffen, welchen der möglichen Züge er wählt.

Es handelt sich folglich um Spiele, bei denen ein Spieler probiert den anderen zu überlisten oder ein Spieler ein „natürlich“ gegebenes Problem zu lösen hat.

1.3 Begriffe

Als Begriffe werden hierbei verwendet werden:

- ZUSTAND

- STUFE {Zustand} x {Entscheidung}
- POLITIK
- ENTSCHEIDUNG hier zumeist ein ZUG
- REKURSIONSGLEICHUNG oder auch DP-GLEICHUNG
- DIMENSION

2 Deterministische Einspielerspiele

Dies ist die einfachste Art der Brettspiele.

- Ein Spieler
- Vollständige Information des Zustandes (Position)

Zu diesen Spielen gehören zum Beispiel die folgenden Spiele:

- Turm von Hanoi
- Der Zauberwürfel (Rubik's Cube)
- 15-Teile-Puzzle
- Freecell und andere Formen von Patience
- Solitär

Gemeinsamkeiten dieser Spiele ist dabei das Ziel, nämlich das Erreichen eines fest vorgegebenen Zustandes und, dass der Startzustand nicht frei wählbar ist.

2.1 Fragestellung und Modellierung

1. Gibt es eine Lösung?
2. Wie kann die Lösung möglichst schnell erreicht werden?

2.1.1

Um feststellen zu können ob eine gegebene Position p lösbar oder unlösbar ist, wird bei der Dynamischen Programmierung eine Wertefunktion f definiert. Der Wert der Lösung bzw. einer lösbaren Position wird auf +1 gesetzt, der einer unlösbaren auf -1 (oder 0). Mit der Rekursionsgleichung der Dynamischen Programmierung kann so theoretisch jede Position eines Spiels ausgewertet werden. Die Rekursionsgleichung für Frage 1 lautet:

$$f(p) = \max_{T} \{f(T(p))\}$$

p = aktuelle Position
f(p) = Wert von p
T = Menge der möglichen Züge

Wobei der Spieler in Position p aus den Zugmöglichkeiten T auswählen kann und die Position zu T(p) verändert.
Problem: Z.T. sehr hohe Dimension, nicht möglich große Sets T auszuwerten.

2.1.2

Wenn eine Position lösbar ist, interessiert man sich für die beste Möglichkeit sie zu lösen. Zumeist sollte dies die schnellste, gemessen in Zügen, sein. Dies führt zur Gleichung:

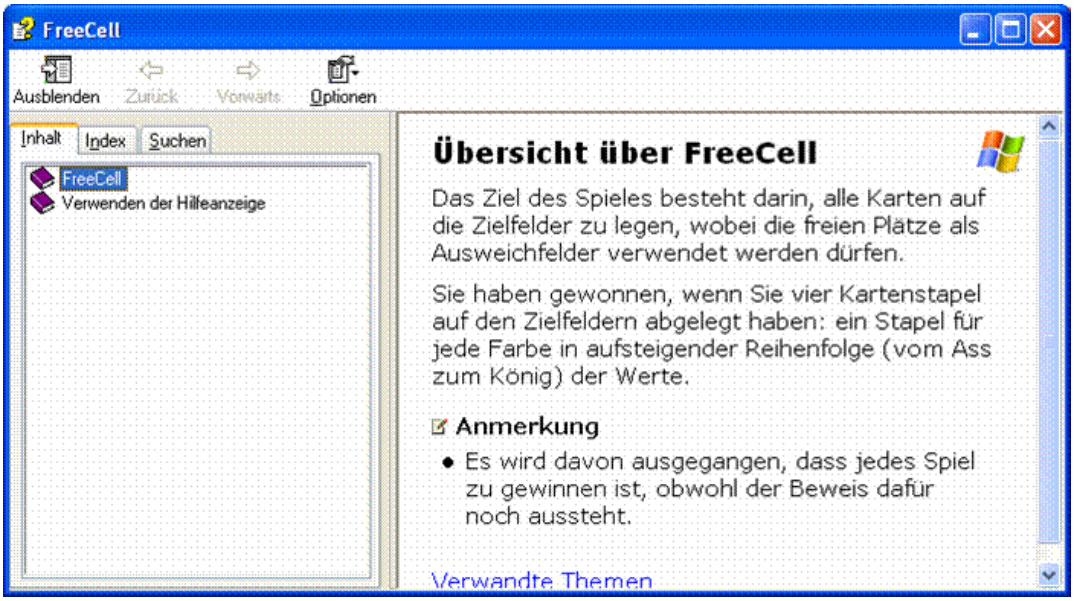
$$f(p) = \min_{T} \{1 + f(T(p))\}$$

Wobei f(p) hier die benötigten Züge bis zur Lösung unter einer optimalen Politik wiedergibt.

Mit diesem Ansatz wurden u. A. schnellstmögliche Lösungen für Rubik's Cube und Solitär gefunden. Auch hier ist die mitunter hohe Dimension ein ernsthaftes Problem. Zumeist wird bei solchen Problemen vorwärts DP eingesetzt (sehr Ähnlich zu der Theorie von Spielbäumen), welches durch den Rückzug auf Subprobleme die mögliche Anzahl von Zuständen einschränkt.

2.2 Freecell

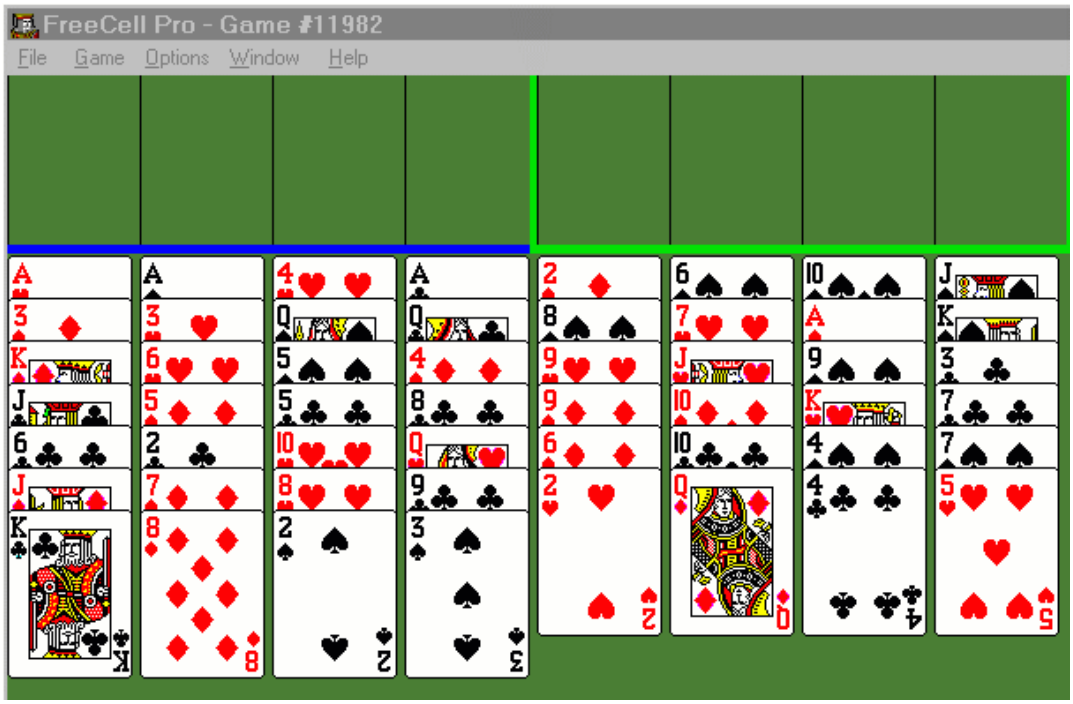
Nach dem Erscheinen von Freecell in Windows 95 startete Dave Ring das „Freecell Projekt", mit dem Ziel alle 32000 unter Windows 95 implementierten Spiele zu lösen oder die Unlösbarkeit zu zeigen.



Im Internet rekrutierte er dafür hunderte Mitstreiter die je 100 konsekutive Spiele lösen sollten. Im Oktober 95 waren alle Spiele mit Ausnahme von Spiel Nummer 11982 gelöst.

2.2.1 Spiel #11982

Der Beweis zur Unlösbarkeit von Spiel 11982 basierte auf der Rekursionsgleichung, konnte aufgrund der hohen Dimension des Problems jedoch nicht direkt mit Dynamischen Programmieren geführt werden und wurde mit einem vorausschauenden Algorithmus in den Game-Tree ausgelagert. Die Unlösbarkeit ist mittlerweile durch mindestens acht unabhängige Computer Programme gezeigt worden.



2.2.2 Daten und Statistik

Insgesamt gibt es jedoch weit aus mehr mögliche Positionen als die unter Windows 95 zunächst implementierten. Für die Berechnung der Anzahl muss berücksichtigt werden: Es gibt:

- 52 Karten
- 8 Stapel, 4 mit 7 und 4 mit 6 Karten
- 4 Spielfarben
- 2 Farben

Dies führt zu 52! möglichen Positionen von denen einige doch äqivalent sind. Dies sind zwei mal 4! mögliche Anordnungen der Stapel und acht Vertauschungen der Spielfarben. Also insgesamt

$$\frac{52!}{(2 \cdot 4!) \cdot 8} = 1,7504 \cdot 10^{64}$$

verschiedenen Startpositionen.

Als unlösbar wurden identifiziert:		Lösbarkeitsquote
1 Spiel unter den ersten	32 000 (Windows 95)	99,99687 %
8 Spiele unter den ersten	1 000 000 (Windows XP)	99,99992 %
136 Spiele unter den ersten	10 000 000	99,99854 %
320 Spiele unter den ersten	25 000 000	99,99872 %
1282 Spiele unter den ersten	100 000 000	99,99872 %

Hierzu wurde in den meisten Fällen das Programm FCPro verwendet.

3 Stochastische Einspielerspiele

Erste Verkomplizierung der deterministischen Einspielerspiele.

- Ein Spieler.
- Änderungen des Zustandes ist zufallsbehaftet.

Die Rekursionsgleichungen aus 2.1. werden dann mit der Übergangswahrscheinlichkeit π_{pq}^{T} von Zustand p in Zustand q unter T versehen. Zur Maximierung der Gewinnwahrscheinlichkeit f(p)löse:

$$f(p) = \max_{T} \left\{ \sum_{q} \pi_{pq}^{T} \cdot f(q) \right\}$$

Zur Minimierung der erwarteten Züge f(p) bis zum Erreichen der Lösung löse:

$$f(p) = \min_{T} \left\{ 1 + \sum_{q} \pi_{pq}^{T} \cdot f(q) \right\}$$

Beispiele sind verschiedene Varianten von Patience.

4 Deterministische Zweispielerspiele

4.1 Kombinatorische Spiele

Als Kombinatorische Spiele wurden von Richard Guy Spiele definiert, die folgenden Kriterien genügen:

1. Es gibt nur zwei Spieler, diese haben unterschiedliche Bezeichnungen wie Links und Rechts oder Schwarz und Weiß. (Also keine Koalitionen.)
2. Das Spiel hat eine wohldefinierte Menge von Positionen, für gewöhnlich ist diese Menge endlich. Generell gibt es eine Startposition.
3. Es gibt Regeln die die zwei Mengen von möglichen Zügen die Links und Rechts von jeder Position aus machen können definieren .
4. Die Spieler ziehen abwechselnd, es gibt keine Glücks- oder Zufallszüge.
5. Ein Spieler der keinen Zug machen kann ist der Verlierer.
6. Das Spiel endet immer weil ein Spieler keinen Zug mehr machen kann.
7. Beide Spieler kennen den Zustand des Spiels, es herrscht vollständige Information. (Keine Bluffs möglich.)

Unter den Kombinatorischen Spielen gibt es noch eine Einteilung in:

- parteiischen Spiele
- unparteiischen Spiele

Die Definition von Guy ist recht stark, so dass viele bekannte Spiele keine strikt kombinatorischen sind. Beispeiel:
Spiele bei denen der Spieler der den letzten Zug machen muss der Verlierer ist heißen Misère Spiele.In den

Spiel	kein komb. Spiel weil...
Mensch ärgere dich nicht, Backgammon...	Zufallszüge (Würfel)
Schere-Stein-Papier	Simultanes ziehen
Schiffe versenken	Unvollständige Information, Glückszüge
Tic Tac Toe	Mögliches Unentschieden
Schach	Mögliches Unentschieden (bei Patt)
Mastermind:	Unvollständige Information

meisten Fällen ist die Analyse dieser Variante eines Spiels schwieriger.

4.2 Nim

Nim ist ein sehr einfaches deterministisches Zweispielerspiel und wird deswegen in der Literatur häufig als Prototyp für die Analyse behandelt.

4.2.1 Spielregeln

- Starte mit einem oder mehreren Stapeln Bohnen.
- Der Spieler der an der Reihe ist, wählt einen Stapel aus von dem er mehr mindestens eine Bohne wegnimmt.

Erweiterungen des Spiels:

- Anzahl der Bohnen die weggenommen werden dürfen ist limitiert.
- Es müssen von mehreren Stapeln Bohnen genommen werden.

4.3 Bellmanns Beitrag

In Bellmans Arbeiten gibt es mehrere Bezugspunkte zur Idee mit Dynamischer Programmierung Brettspiele zu analysieren. Er setzte dazu voraus, dass es möglich ist jeder im Verlauf eines Spiels vorkommenden Position p einen Wert f(p) zuzuordnen. Von Weiß aus gesehen: f(p) = 1 falls p Gewinnerposition f(p) = 0 falls p zu Unentschieden führt f(p) = -1 falls p Verliererposition

- Ein Zug von Weiß ändert die Position p zu Tw(p).
- Weiß hat mehre mögliche Züge aus $T_W = \{W_1, W_2, ...\}$ zu Verfügung.
- Nach dem Zug von Weiß sind die möglichen Positionen also $T_W(p) = \{TW1(p), TW2(p), ...\}$
- Schwarz zieht aus $T_B = \{B_1, B_2...\}$.

Nach dem Zug sind die möglichen Positionen also:

$$T_B(T_W(p)) = \{TB_1(TW_1(p)), TB_1(TW_2(p)), ...TB_2(TW_1(p)), TB_2(TW_2(p)), ...\}$$

Weiß probiert dabei den Wert der Position zu maximieren. Schwarz versucht den Wert der Position zu minimieren. Natürlich weiß Spieler Weiß dies und wählt seinen Zug so dass er nach Minimierung durch Schwarz noch maximal ist.

4.3.1 Die Rekursionsgleichung für Nim

Die oben gemachten Überlegungen und Notationen führen zur folgenden Gleichung:

$$f(p) = \max_{T_W} \left\{ \min_{T_B} \{ f(T_B(T_W(p))) \} \right\}$$

Die Formel gilt so auch allgemein für deterministischen Zwispielerspiele.
Bellmann betrachtete als Beispiel Schach und kommentierte wie folgt:

„As it stands, however, the equation is useless computationally because of the dimensionality barrier, and it is useless analytically because of our lack of knowledge of the intrinsic structure of chess.“

4.4 Zwei Lösungsansätze für Nim

4.4.1 Mit Dynamischem Programmieren

Das Problem ist vollständig beschrieben mit der verbleibenden Anzahl von Bohnen je Stapel. Eine Position nahe dem Ende könnte etwa sein:

$$p = \{1, 2, 2\}$$

Weiß ist am Zug und kann die Position zu einer der folgenden verändern:

$$TW_1(p) = \{2, 2\}, TW_2(p) = \{1, 1, 2\}, TW_3(p) = \{1, 2\}$$

Schwarz's Antworten können daraus ergeben:

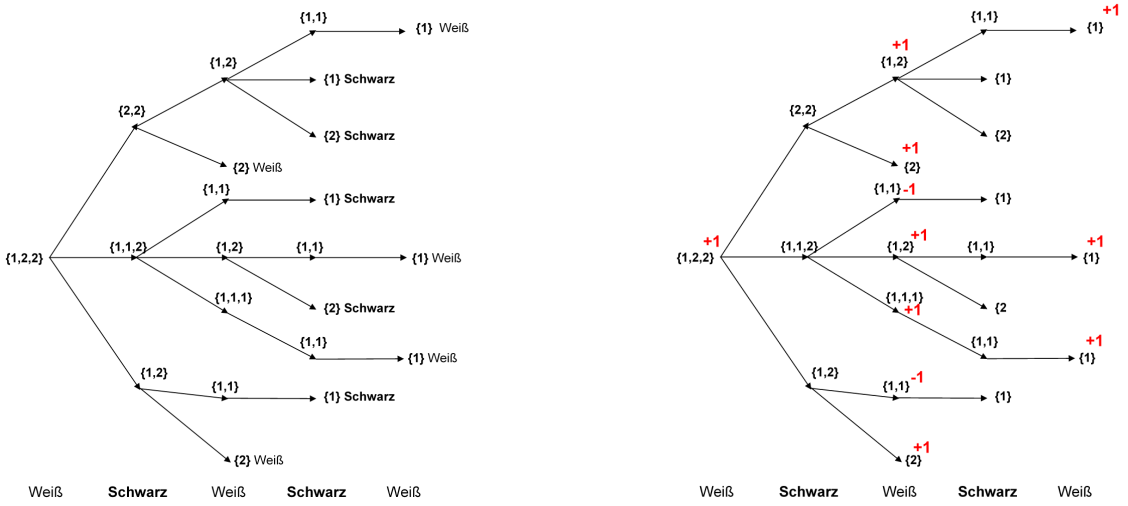
Von: { 2,2 } zu: { 1,2 } , { 2 }

Von: { 1,1,2 } zu: { 1,1 } , { 1,2 } , { 1,1,1 }

Von: { 1,2 } zu: { 1,1 } , { 2 }

Leider ist für keine dieser Positionen der Wert bekannt, jedoch ist offensichtlich, dass Weiß gewinnen kann wenn nur noch ein Stapel übrig geblieben ist. Nutze als Bedingung:

$$f(\{n\}) = +1 \qquad \forall \ n \in \mathbb{N}$$



4.4.2 Mit binärer Arithmetik

Satz: Es gilt dass jede Position äquivalent ist zur selben Position in der in zwei Stapeln die gleiche Anzahl an Bohnen weggenommen wurde.

Diese einfache aber grundlegende Beobachtung vereinfacht die Analyse erheblich und hat zum auffinden folgender Regel beigetragen:

- Schreibe die Anzahl der Bohnen in jedem Haufen binär.
- Addiere Spaltenweise ohne Übertrag die sogenannten Nim-Summen (XOR).
- Spiele so, dass alle Nim-Summen nach dem Zug Null sind.

Wie auch bei Bellman wird hier der Wert der Position so einfach wie möglich beschrieben.

4.4.3 Beispiel und „Beweis“

Wir nehmen wieder die Position p = {1,2,2} aus 4.4.1.
Binär ergibt sich:

$$\begin{aligned} 1 &= 01 \\ 2 &= 10 \\ \underline{2} &= \underline{10} \\ &= 01 \end{aligned}$$

Die Nim-Summen sind 1 und 0. Wir befinden uns in einer Gewinnerposition, da es immer einen Zug gibt der alle Nim-Summen zu 0 werden lässt. Dieser Zug berechnet sich wie folgt:

- Wähle die Spalte j mit der ersten 1 in der Nim-Summe.

- Wähle eine Zeile i mit einer 1 in Spalte j.

Hier also die zweite Spalte und dann die erste Zeile. So eine Zeile lässt sich immer finden da die Anzahl von Einsen in dieser Spalte ungerade ist.

- Nimm Aus Stapel i genau $(Anz.Bohnen)_2 XOR (Nim - Summe)_2$ Bohnen.

Dann muss die Nim-Summe zu Null werden. Hier:

$$(01) XOR (01) = 01$$

Nimm also 1 Bohne aus Stapel 1. Dies führt zu $T_W(p) = \{2,2\}$ mit Nim-Summen 0 und 0.

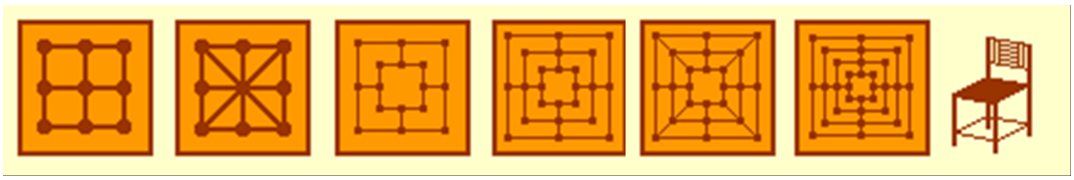
4.5 Positions-Spiele

Unter diesem Begriff kann man verschiedene Spiele wie Tic Tac Toe, Mühle, Fünf in einer Reihe, Blockade Spiele und auch Solitär zusammenfassen.

4.5.1 Tic Tac Toe und Mühle-Spiele

Tic Tac Toe ist eines der einfachsten Positions-Spiele und kann mit der Bellmanschen Methode problemlos analysiert werden. Die optimale Strategie führt hier zu einem Unentschieden.

Mühle-Spiele gibt es in verschiedenen Größen. Die einfachste Variante ist schon von Ovid erwähnt worden. Bei der 3x3 Variante wird mit drei Spielsteinen gespielt. Mit der Rekursionsgleichung kann man eine optimale Strategie bestimmen, bei der der erste Spieler die Mitte besetzten muss und die stets zu einem Unentscheiden führt.



4.5.2 Durch totale Enummeration gelöste Spiele

Mit den Möglichkeiten moderner Computer ist es mittlerweile machbar, Spiele mit totaler Enummeration zu lösen. Beispiele hierfür sind Vier-Gewinnt und das „klassische“ Mühle-Spiel.

4.5.3 Endspiele

Positionen in denen die Anzahl von Spielsteinen oder Entscheidungsmöglichkeiten bereits erheblich reduziert worden ist, sind sogenannte Endspiele. Durch die Reduktion der Dimension lassen sich für diese Subprobleme einfacher Lösungsstrategien finden. Jedoch gibt es auch hier mitunter noch viele mögliche Zustände. Als Endspielprobleme bezeichnet man etwa:

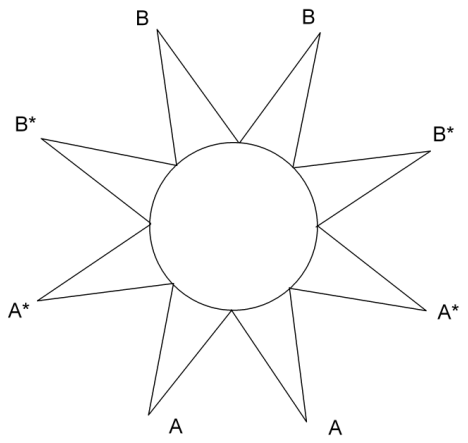
- Dame mit weniger als 9 Steinen
  - über 150 Mrd. Zustände
- Schach mit weniger als 6 Figuren.

4.6 Blockade Spiele

Bei Blockade Spielen ist es Ziel, den Gegner in eine Position zu bringen in der er keinen Zug mehr machen kann. Das einfachste Spiel dieser Art ist das Maori Spiel Mu Torere.

- Spieler A und B haben je 4 Spielsteine.
- Abwechselnd muss jeder Spieler einen Zug machen, dabei darf er einen seiner Steine in ein angrenzendes freies Feld bewegen.

- In den ersten beiden Zügen dürfen nur die äußeren Steine bewegt werden.



Mu Torere wurde im Rahmen einer Arbeit über die Verwendung von Mathematik in anderen Kulturen von Marcia Ascher untersucht. Ihre Ergebnisse zeigen, dass das Spiel unter einer optimale Strategie für beide Spieler unendlich geht. Weicht jedoch ein Spieler von dieser Strategie ab kann der Gegenspieler gewinnen.

5 Zweispielerspiel bei unvollständiger Information

Zu dieser Art von Spielen gehört Mastermind, das bereits mit vielen hier bereits vorgestellten Methoden analysiert worden ist. Pascal van Hentenryck nutz die numerische Variante „Bulls and Cows“ (Hier ist eine Folge vier verschiedener Ziffern aus {0...9} zu erraten.) um eine suboptimale Strategie zu finden. Hierfür formuliert er die Regeln wie folgt:

- Finde eine Vermutung die mit den vorhergehenden konsistent ist.
- Beachte das Feedback.
- Stopp wenn der Code korrekt ist, andernfalls füge das Feedback zum Wissen über die Position hinzu.

Ist das Ziel die Lösung in möglichst wenig Versuchen herauszubekommen bedingt dies das Setzen von ein paar unkonsistenten Vermutungen.

6 Stochastische Zweispielerspiele

Hier dargestellt am Memory Spiel. Wir gehen davon aus, dass die Spieler ein perfektes Gedächtnis haben, denn dann ist die Strategie entscheidend.

Zielfunktion: Maximiere die richtig aufgedeckten Kartenpaare.

Die verbliebenen Karten bestehen aus n Paaren von Karten von denen k alte bekannt sind. In jedem Zug muss sich der Spieler für einen der drei möglichen Spielweisen entscheiden:

- Spiele „Alt“-„Alt“
- Spiele „Neu“ und wenn kein Paar lösbar eine weitere „Neu“ Karte.
- Spiele „Neu“ und wenn kein Paar lösbar eine „Alt“ Karte.

Von Interesse ist hier wann ist „Neu“-„Neu“ und wann „Neu“-„Alt“ zu spielen.

Durchgeführt wurden die Rechnungen von Zwick und Patterson, die dazu folgende Rekusionsgleichung aufstellten:

6.1 Die Rekusionsgleichung für Memory

$$f(n,k) = max\Bigg\{\Bigg\{\frac{k}{2n-k}\left(1+f(n-1,k-1)\right) - \frac{2(n-k)}{2n-k}f(n,k+1)\Bigg\},$$
$$\Bigg\{\frac{k}{2n-k}(1+f(n-1,k-1)) - \frac{(2n-k)}{2n-k}\left(\frac{k-1}{2n-k-1}(1+f(n-1,k)) + \frac{2(n-k-1)}{2n-k-1}f(n,k+2)\right)\Bigg\}\Bigg\}$$

Die Rekursionsgleichung umfasst nur die Varianten „Neu“-“Alt“ und „Neu“-“Neu“, denn wäre in einer Stufe „Alt“-“Alt“ optimal, so auch für den Gegner und es wäre ein „totes Spiel“.

Der erste Teil der Gleichung bezieht sich auf den Zug „Neu“-“Alt“ und setzt zusammen aus :
Wahrscheinlichkeit $\frac{k}{2n-k}$ eine zu einem der k bekannten Paare passende Karte umzudecken multipiziert mit dem Wert des gewonnenen Paares plus dem Wert des neuen Zustandes, bei dem noch n-1 Paare im Spiel sind von denen k-1 bekannt sind, also (1+ f (n-1,k-1)).
Abzüglich der Wahrscheinlichkeit $\frac{2(n-k)}{2n-k}$ kein Paar aufdecken zu können multipliziert mit dem Wert den das Spiel für den Gegner bekommt, der jetzt von f(n,k+1) aus spielt

Daraus folgen die Regeln:

- Spiele „Alt“-„Alt“ falls 3k $\geq$ 2(n+1) und n+k ungerade.
- Spiele „Neu“-„Alt“ falls k $\geq$ 1 und n+k gerade oder (n,k)=(6,1).
- Spiele „Neu“-„Neu“ sonst.

Die scheinbar einfache Struktur des Problems täuscht hier. Der Beweis der Resultate von Zwick und Patterson ist lang und komplex.

7 Schluss

Dynamisches Programmieren ist eine, zumindest von der Theorie her, sehr leistungsstarke Möglichkeit Brettspiele zu analysieren. In der Praxis ist die Leistungsfähigkeit jedoch oft von den Kapazitäten der zur Verfügung stehenden Rechner begrenzt. Man kann also erwarten, dass mit zunehmender Rechenleistung auch das Dynamische Programmieren mehr eingesetzt werden kann. Zudem eröffnet sich auf dem Feld der künstlichen Intelligenz ein weites Einsatzspektrum welches den Entscheidungen auf Brettspielen sehr nahe steht.

8 Literatur

David Smith *Dynamic Programming and Board Games*
Jürgen Köller *www.mathematische-basteleien.de*
Michael Keller *www.solitairelaboratory.com*
Leif Bennett *www.alumnus.caltech.edu*