Gymnasium St. Leonhard

Facharbeit von Adrian Hinrichs

Thema:

Das Prinzip von Spielbäumen

exemplarisch erarbeitet anhand der Implementation eines Computer-Gegners beim Mühle-Spiel

Betreuender Lehrer: Herr Meltzow
Thema wurde gestellt am 5. Februar 2015
Abgabetermin: 18. März 2015

Note: _____ _____

Unterschrift

Korrigierte Fassung vom 25. April 2015

Inhaltsverzeichnis

1 Einleitung

Das Simulieren intelligenten Verhaltens stellt schon seit länger als der Mitte des 20. Jahrhunderts[1] einen für die Menschen faszinierenden Zweig der Informatik dar. Unter das Teilgebiet der künstlichen Intelligenz (KI) fällt auch die computergesteurte Austragung von Spielen, wie z.B. dem Brettspiel Mühle.

Die Implementation einer solchen KI könnte in vielfacher Weise stattfinden, zum Beispiel in der Nachahmung der menschlichen Synapsenstruktur in einem künstlichen Neuronalen Netzes. Eine weitere, simplere Methode, um eine solche KI zu implementieren besteht in der Methodik der Spielbäume, welche ich im Rahmen dieser Facharbeit erarbeiten werde.

2 Theorie

2.1 Spielbäume

Ein Spielbaum ist ein aus der Spieltheorie stammender Baum, mit welchem der Verlauf eines beliebigen Spieles abgebildet werden kann.

Die Kanten sind in diesem Fall die Züge, die Knoten wären einzelne Spielsituationen, wobei die inneren Knoten mit einer Entscheidung verbunden sind und die Blätter die Endzustände des Spieles darstellen. Diese Form der Darstellung wird auch Extensivform genannt, da die Reihenfolge der Aktionen brücksichtigt wird.[2]

Angefangen bei der Wurzel, welche die Ausgangssituation einnimmt, führen die Spieler ihre Züge abwechselnd aus, so das jede Ebene des Baumes einem Spieler zugeordnet werden kann.

Um dies zu verdeutlichen, betrachten wir ein einfaches Marktspiel: Ein Unternehmen ist Marktführer mit einem bestimmten Produkt, ein anderes Unternehmen steht nun vor der Wahl ebenfalls dieses Produkt zu produzieren, also in den Markt einzutreten oder weg zu bleiben. Der Gewinn für den Marktführer bei Wegbleiben des Anderen würden sich auf 36 belaufen, 16 sei der Gesammtgewinn, wenn der Marktführer aggressiv reagiert, 32 wenn er friedlich reagiert. In beiden fällen würden sich die Konkurenten den

[1] Laut Wikipedia [Webb] fand am 13. Juli 1956 am Dartmouth College die erste Konferenz statt, die sich dem Thema künstliche Intelligenz widmete

[2] S. 91f. [BEG05]Berninghaus, Siegfried K. ; Erhardt, Karl-Martin ; Güth, Werner: *Strategische Spiele*. 2. Auflage. Springer, 2005

Gewinn teilen, wobei jedoch das hinzutretende Unternehmen Kosten in der Höhe von 12 tragen müsste, um in den Markt einzutreten. Der Spielbaum zu dieser Simulation ist in Abbildung 1 zu sehen. [3]

Die oberste Ebene „gehört“dem Konkurenten, da er die Entscheidung treffen muss, die

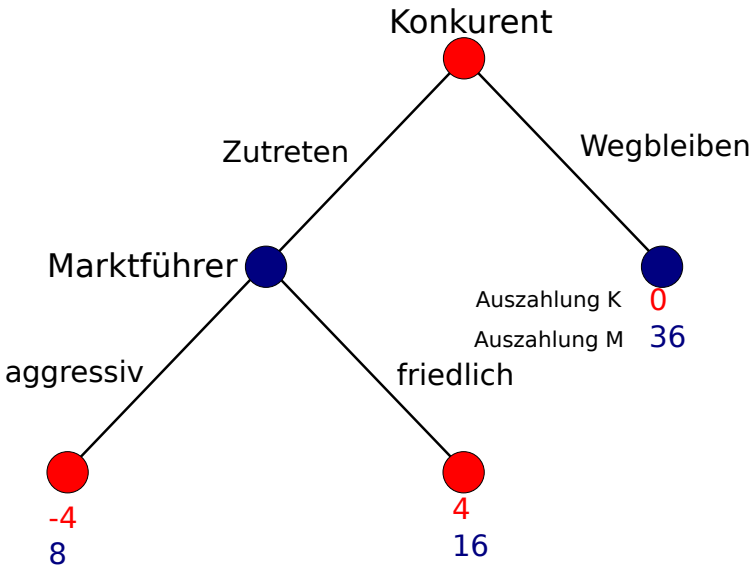


Abbildung 1: Spielbaum des Martkspiels (aus Amann, Erwin; Helbach, Christoph: *Spieltheorie für Dummies*. WILEYVCH Verlag, 2012,Seite 87)

mittlere dem Marktführer und in der untersten Ebene muss keine Entscheidung mehr getroffen werden, da das Spiel beendet ist.

2.2 Nullsummenspiele

Bei Spielen, insbesondere bei Gesellschafts- und Brettspielen, kann eine jede Spielsituation für einen jeden Spieler mit einer sog. Auszahlung bewertet werden, welche den Nutzen der jeweiligen Situation für den jeweiligen Spieler beschreibt. Spiele, bei denen die Auszahlung aller Spieler zusammen null ergibt, nennt man Nullsummenspiele.

Das Marktspiel aus Abschnitt 2.1 ist kein Nullsummenspiel, da der Gewinn für das eine Unternehmen immer höher ist als der Verlust für das Andere und somit Geld „aus dem nichts“alloziert würde.

[3]Beispiel entlehnt aus [AH12]Amann, Erwin; Helbach, Christoph: *Spieltheorie für Dummies*. WILEYVCH Verlag, 2012,Seite 87

2.3 Der Minimax-Algorithmus

Der Minimax-Algorithmus kann verwendet werden, um „eine Strategie zu finden, welche den maximalen Verlust minimiert“, also in Nulsummenspielen eine Strategie zu finden, mit welcher man gewinnt.[4]

Dies wird erreicht, in dem man zunächst die Blätter des Baumes bewertet,eine Möglichkeit dazu besteht darin dem Blatt einen Wert von $+\infty$, wenn man selber gewinnt zuzuweisen, $-\infty$ wenn der Gegner gewinnt, da es keinen größeren Nutzen als den Sieg gibt, und 0 für unentschieden.[5] Danach arbeitet man sich von den Blättern “nach oben“bis zur Wurzel durch und weist jedem Knoten den maximalen Wert seiner Kinder zu, wenn man selber am Zug ist, andernfalls, wenn der Gegner am Zug ist, weist man dem Knoten das Minimum der Werte der Kinder zu.// Der Minimax-Algorithmus führt also nur eine Tiefensuche nach einem maximalen Gewinn durch.[6]

Dieses vorgehen ist in der Praxis jedoch nicht immer möglich, da der Spielbaum meist zu komplex ist, um ihn komplett zu traversieren, daher muss man in diesem Fall auf Heuristiken zurückgreifen (siehe 3.3).

3 Anwendung auf das Brettspiel Mühle

3.1 Spielphasen

Das Brettspiel Mühle ist in drei Spielphasen unterteilbar: die erste Phase ist die „Setzt-Phase“, beide Spieler müssen ihre Steine auf das Feld setzen. Die erste Phase geht maximal bis in die 9. Runde. Die zweite Phase ist die „Zug-Phase“, Spieler können ihre Steine von einem Feld auf das benachbarte Feld ziehen, sie endet sobald ein Spieler nur noch drei Steine hat, dann beginnt für ihn nämlich die „Sprung-Phase“, in welcher er auf jedes freie Feld springen kann.

3.2 Komplexität des Spielbaumes

Die Tatsache, dass der MiniMax-Algorithmus ein Bruteforce-Verfahren ist, birgt aufgrund der Beschaffenheit des Mühle-Spiels einige Probleme.

[4] „Minimax [...] is a decision rule [...] for minimizing the possible loss for a worst case (maximum loss) scenario.“[Webc]
[5] Vergleiche [Webc]
[6] Vergleiche [Weba]

Die Anzahl aller möglichen unterschiedlichen Stellungen beläuft sich schätzungsweise auf etwa $1,8 * 10^{10}$ Stellungen[7], was einer Datenmenge entspricht, welche nicht im Arbeitsspeicher eines gewöhnlichen Computers abgelegt werden kann.

Eine weitere Schwierigkeit in der Komplexität ist die unregelmäßige Struktur des Baumes.
In der ersten Spielphase wächst der Baum zunächst sehr stark, da man zu Beginn auf alle 24 Felder setzen kann. Das Wachstum des Baumes nimmt im Laufe der ersten 9 Züge aber auch rasch ab, da die Felder mit der Zeit alle besetzt werden. Da sich in der zweiten Phase alle Steine um nur ein Feld pro Zug bewegen dürfen, und die Anzahl der Steine im Laufe des Spieles abnimmt, ist auch das Wachstum des Baumes im Vergleich zur ersten Phase nicht sonderlich stark. In der dritten Phase jedoch steigt das Wachstum erneut an, da mindestens ein Spieler seine drei Steine (fast) frei bewegen darf.

3.3 Heuristik

Damit der Baum nicht vollständig berechnet werden muss, wird eine sogenannte Heuristik durchgeführt, auf einer bestimmten Ebene des Baumes werden die Knoten nicht durch die Weiterführung des MiniMax-Algorithmus, sondern durch eine vereinfachte Bewertungsfunktion bewertet.

Die Bewertungsfunktion in meiner Implementation bewertet zunächst jede Gerade, auf der eine Mühle gebildet werden kann einzeln, um anschließend die Bewertungen aller Geraden zu summieren. Es werden für jede Gerade die Werte der Felder summiert und anschließend mit dem Faktor 3 potenziert: Dies bewirkt, dass Mühlen mehr wert sind als fast fertige Mühlen oder drei einzelne, auf dem Brett verteilte Steine.

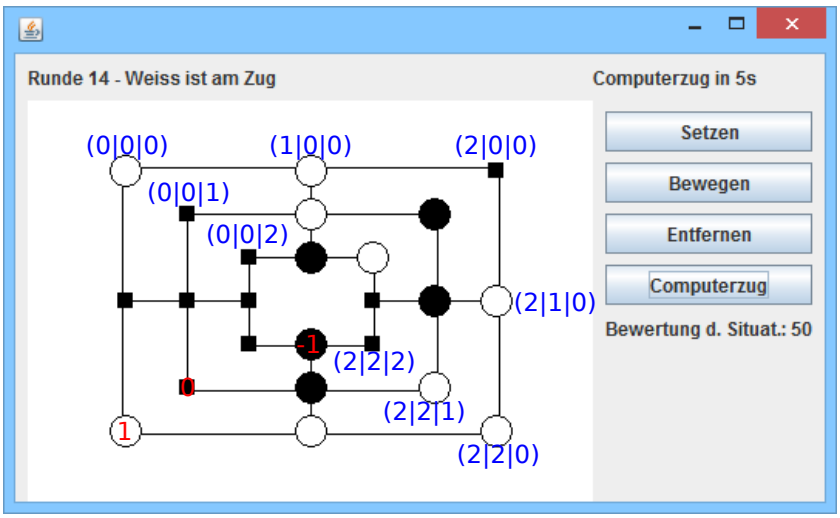
3.3.1 Beispiel

Um die Funktionsweise der Funktion besser zu erklären, sei eine Spielstellung wie in Abbildung 2 gegeben.

Angefangen bei der ersten Ebene werden nun die Steine auf den einzelnen Geraden summiert und potenziert, so dass allein für die „oberste“ Gerade der äußeren Ebene gilt:

$$\left(\begin{pmatrix}0\\0\\0\end{pmatrix} + \begin{pmatrix}1\\0\\0\end{pmatrix} + \begin{pmatrix}2\\0\\0\end{pmatrix}\right)^3 = (1+1+0)^3 = 8$$

[7] Vergleiche [Webd]



Legende:
(x|y|z) Koordinaten der Felder (exemplarisch)
-1; 0; 1 Werte der Felder (exemplarisch)

Abbildung 2: Screenshot des Programmes mit Anmerkungen

Auf diese Weise würde nun auch die andere horizontale und die beiden vertikalen Geraden dieser Ebene berechnet werden, was summiert für diese Ebene den Wert 51 ergibt. Führt man dies nun auch für die anderen Ebenen aus, erhält man den Wert -1 für die mittlere und 0 für die innere Ebene. Zusammen ergibt dies den Wert 50, es müssen jedoch auch noch die Geraden zwischen den Ebenen nach dem gleichem Prinzip mit eingerechnet werden. Da diese Geraden sich gegenseitig aufheben, verändert dies in diesem Fall das Ergebniss jedoch nicht.

4 Architektur

Wie man dem UML-Diagramm (Anhang 6.1) entnehmen kann, besteht die Software aus drei Klassen, die allein für die Logik der KI zuständig sind, und zwei weitere Klassen, welche das Benutzer-Interface bilden, worauf ich allerdings in dieser Arbeit nicht näher eingehen werde.

4.1 Spielfeld

Die Klasse `Spielfeld` ist primär für die Speicherung der Spielsituation während der Ausführung des Programms konzipiert.

Das Spielbrett wird durch ein dreidimensionales Array modelliert, in welchem (wie schon angesprochen) je nach Spieler die Werte 1, -1 oder, falls das Feld nicht besetzt ist 0 abgelegt werden. Das Feld wird dazu mit Koordinaten versehen, wobei die X- und Y-Koordinaten jeweils die Ausdehnung des Feldes zur Seite bzw. nach Unten oder Oben darstellt und die Z-Koordinate den Ring des Mühle-Brettes repräsentiert, wobei der äußere Ring den Wert 0 hat und somit der innerste die Z-Koordinate 2 besitzt.

Die Klasse Spielfeld besitzt außerdem Attribute, um den Spieler, welcher am Zug ist, die Nummer der Runde, die Spielphase (siehe Abschnitt 3.1) und die Tatsache, ob der Spieler, welcher momentan am Zug ist einen Stein des Gegners entfernen darf zu speichern.

Selbstverständlich besitzt das Spielfeld auch Methoden, um Spielsteine zu setzen und zu Bewegen, wobei das Bewegen für die jeweils letzten beiden Spielphasen anwendbar ist.

Das Spielfeld hat außerdem noch eine Reihe von weiteren Methoden, welche Funktionen wie das Überprüfen auf einen Sieger oder das Prüfen auf eine Mühle ausführen. Eine weitere Methode des Spielfeldes ist die Methode `bewerte()`, welche die Heuristik durchführt und das Ergebnis als `Integer`-Wert zurück gibt.

4.2 Spielbaum

Die Klasse `Spielbaum` bildet zugleich die Struktur zur Verwalten des Baumes als auch die Knoten selber.

Jeder Knoten besitzt eine Referenz zu seinem Vater, dem ersten Kind und dem nächsten Bruder, so dass die Kinder jeweils über ihre eigenen Geschwister referenziert sind (siehe Abbildung 3). Der Spielbaum hat ein Attribut `Kosten`, in welchem der „Wert“des Spielfeldes gespeichert wird.

Da der Spielbaum stark anwendungsorientiert ist, fehlen Methoden zum Löschen und Einfügen von bestimmten Knoten, es gibt aber eine Methode, um alle Zugmöglichkeiten, also alle Kinder des im Baum gespeicherten Spielstandes zu generieren. Im Spielbaum befindet sich außerdem die zentrale Methode `Minimax()`, welche rekursiv den Baum aufbaut und bewertet, so wie bei Erreichen der maximalen Rekursionstiefe die Heuristik durchführt.

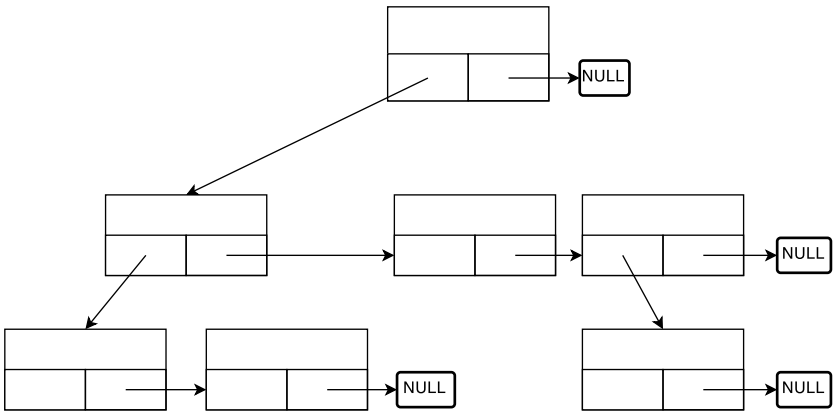


Abbildung 3: Referenzierung des Spielbaumes

4.3 Computergegner

Die Klasse `Computergegner` simuliert den Gegner, indem sie den besten Spielzug berechnen lässt und diesen Spielzug anschließend auf das Spielfeld anwendet, welches ihr übergeben wurde. Diese Aufgabe ist in der Methode `Ziehen` implementiert. Des Weiteren hat die Klasse ein öffentliches Attribut `rekursinstiefe`, in welchem die maximale Tiefe des Spielbaumes gespeichert ist.

5 Résumé

Aufgrund der Limitierung der Ressourcen, welche dem Programm zur Verfügung stehen, kann das Programm keine optimale Strategie finden; teilweise ist es sogar möglich, dass das Programm keine zum Sieg führende Strategie findet, da vor allem in der ersten Spielphase der Computer einige Fehler macht, welche später nicht mehr korrigiert werden können.
Eine Lösung für dieses Problem könnte die Alpha-Beta-Suche darstellen, sie optimiert den MiniMax-Algorithmus, indem einige Äste des Spielbaumes abgeschnitten und nicht weiter verfolgt werden.
Dieses Problem ließe sich eventuell durch den Aufbau einer sogenannten Wissensdatenbank beheben, der Spielbaum würde teilweise in einer Datenbank abgebildet werden, wobei für jede Spielsituation der beste Zug gespeichert wäre.

Dieses Prinzip ist auch in modernen Schachcomputern implementiert. Sie verfügen über spezielle Datenbanken für die Eröffnung, in welchen die Spielsituationen vorgespeichert sind. Diese Optimierung und hohe Rechenleistungen sorgen dafür, dass Schach-

computer so stark sind, dass sie auch für erfahrene Großmeister ein starker Gegner sind.

Literatur

[AH12] Amann, Erwin ; Helbach, Christoph: *Spieltheorie für Dummies*. WILEY-VCH Verlag, 2012

[BEG05] Berninghaus, Siegfried K. ; Erhardt, Karl-Martin ; Güth, Werner: *Strategische Spiele*. 2. Auflage. Springer, 2005

[Weba] *Adversariale Suche für optimales Spiel: Der Minimax-Algorithmus und die Alpha-Beta-Suche*. `http://home.in.tum.de/~adorf/pub/alphabeta-seminar-paper.pdf`. – Abgerufen am 16.01.2015 um 11:00

[Webb] *Künstliche Intelligenz*. `http://de.wikipedia.org/wiki/K%C3%BCnstliche_Intelligenz`. – Abgerufen am 12.03.2015 um 17:55

[Webc] *Minimax*. `http://en.wikipedia.org/wiki/Minimax`. – Abgerufen am 16.02.2015 um 11:00

[Webd] *Mühle (Spiel)*. `http://de.wikipedia.org/wiki/M%C3%BChle_%28Spiel%29`. – ABgerufen am 14.03.2015 um 15:30

Erklärung

Ich versichere, dass ich diese Facharbeit eigenständig verfasst, keine anderen Quellen und Hilfsmittel als die hier angegebenen benutzt und die Stellen der Arbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen sind, in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe. Das gleiche gilt auch für beigegebene Zeichnungen, Kartenskizzen, Darstellungen und Ähnliches.

Aachen, den 18.03.2015	
Ort, Datum	Unterschrift

6 Anhang

6.1 UML-Diagramm

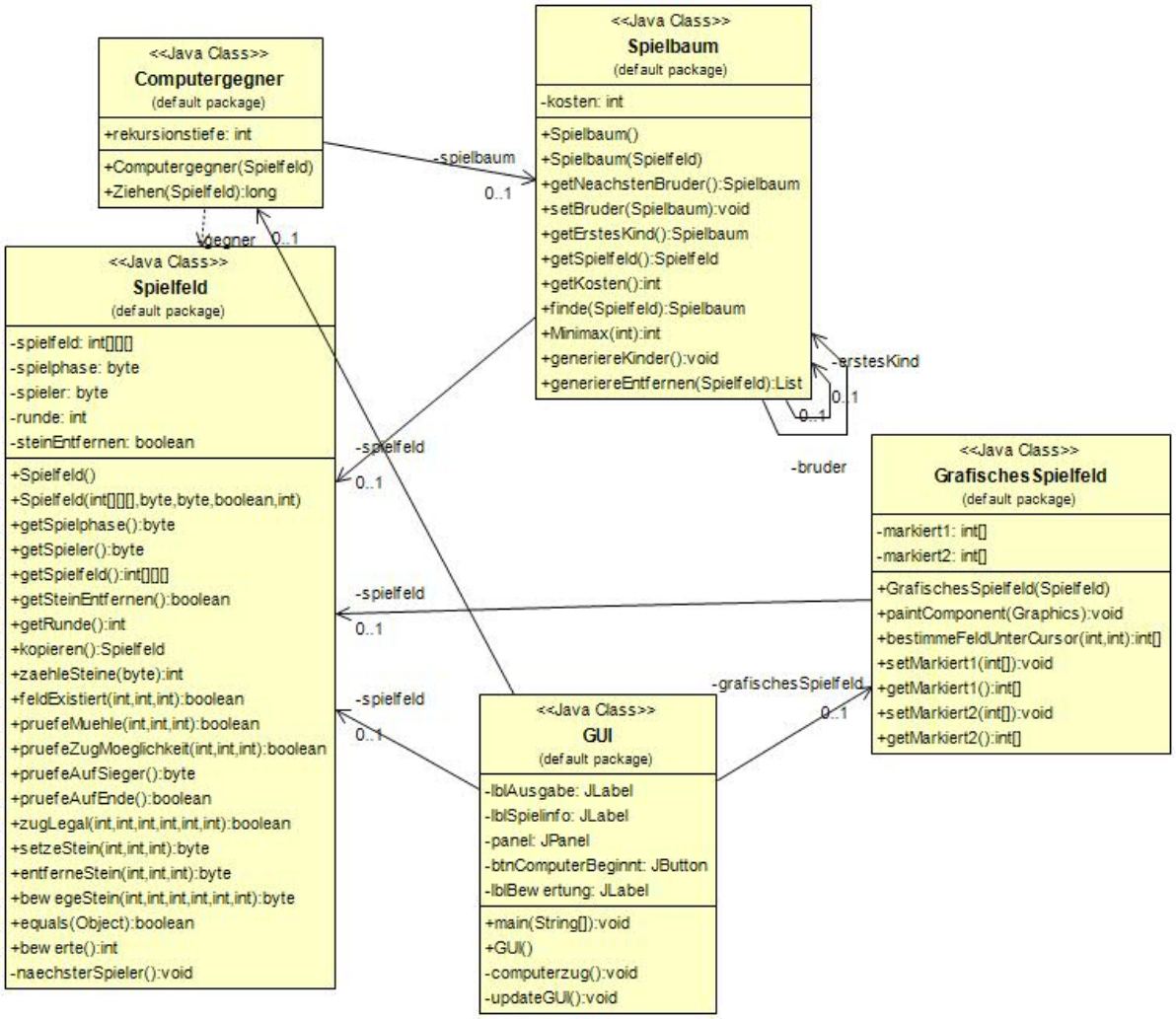


Abbildung 4: UML-Diagramm

6.2 Quelltext

Der Quelltext liegt der Facharbeit in Form einer CD bei.