Spielbäume

Proseminar Theoretische Informatik

Joschka Heinrich*, TU Dresden

28. März 2018

Zusammenfassung

Was sind Spielbäume und wozu können sie verwendet werden? In dieser Ausarbeitung wird das Konzept des Spielbaumes formal eingeführt und anhand des Spiels *Tic-Tac-Toe* beispielhaft erläutert. Mit der Erklärung des Alpha-Beta-Prunings als Optimierung des Minmax-Verfahrens und anderer Heuristiken wird eine Anwendung von Spielbäumen illustriert.

I. Einführung

Motivation. Überblick Aufbau Text.

II. Spielbäume

Um im Folgenden mit Spielbäumen arbeiten zu können, führen wir das Konzept zunächst formal ein und definieren dazu unter anderem den Begriff des *Spiels*, der *Konfiguration* und des *Spielbaums*.

i. Definition

Nullsummenspiel

Alle folgenden Betrachtungen nehmen wir aus der Perspektive eines gewinnorientierten Spielers namens Max vor, der sich einer Anzahl Gegner gegenüber sieht. Es ist also das Ziel, Züge für Max so zu finden, dass dessen Gewinn maximiert, bzw. der Gewinn der Gegner minimiert wird, wobei wir davon ausgehen, dass alle Gegner optimale Entscheidungen treffen. Wir vereinfachen die Betrachtung von Spielen, indem wir uns auf solche ohne Zufallskomponente, d.h. reine Strategiespiele mit vollkommener Information und Spiele mit zwei Kontrahenten—Max und ein zweiter Spieler Min—beschränken. Mit „der Gegner“ ist also im Folgenden stets Min gemeint.

*joschka.heinrich@tu-dresden.de, PGP: B40E 67C7 FF62 C860 7854 A778 6FB9 666F 1147 A401

Ein **Spiel** $S = (R, k_0, F)$ ist nun durch Regeln, in Form einer endlichen Menge von legalen Spielzügen R, eine Anfangskonfiguration $k_0 \in K$ und eine Reihe möglicher Endkonfigurationen $F \subset K$ gegeben, mit K, der Menge aller *zulässigen* Konfigurationen. Eine **Konfiguration** $k \in K$ repräsentiert dabei einen möglichen Zustand des Spieles, bestehend aus einer Beschreibung wiederum der Zustände aller relevanten Spielelemente (bspw. die Position der Zeichen auf dem Tic-Tac-Toe-Feld) inklussive des Spielers, der als nächster an der Reihe ist (bei uns entweder MAX oder MIN).

In Abgrenzung zur Menge der *legalen* Spielzüge können wir uns beliebige andere Spielzüge vorstellen, die zwar möglich, allerdings in dem betrachteten Spiel nicht erlaubt sind. Analog dazu sind über K hinaus weitere Konfigurationen denkbar, die allerdings nicht zulässig sind, d.h. in einem regelkonformen Spiel niemals auftreten können. Durch Anwenden eines legalen Spielzuges auf eine Konfiguration gelangen wir zu einer neuen Konfiguration. Ein legaler Spielzug kann also als eine Funktionen $R : K \to K$ verstanden werden. Seien $u, v \in K$ Konfigurationen und $r \in R$ ein legaler Spielzug, mit $v = r(u)$, dann heißt v **Kindkonfiguration** von u (bezüglich r) und u **Elternkonfiguration** von v (bezüglich r).

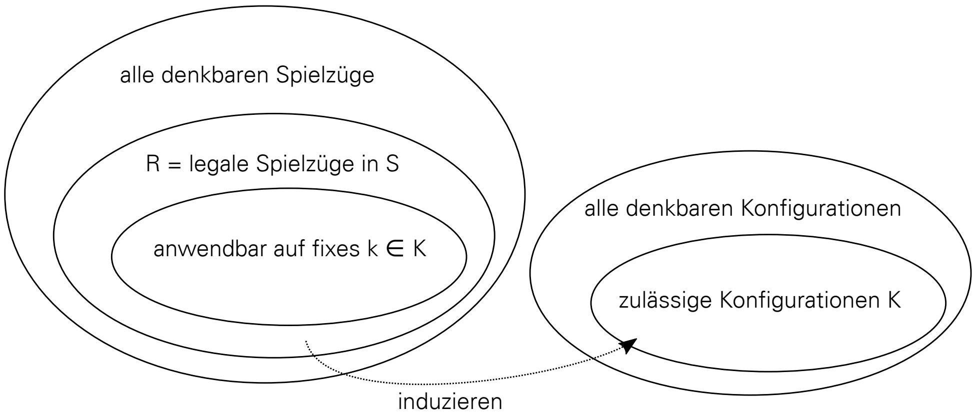

Abbildung 1: VENN–Diagramm Spielzüge und Konfigurationen

Wenn das Anwenden eines Spielzuges zu einer neuen Konfiguration führt, also $u \neq v$ gilt, dann heißt r **anwendbar** auf u.[1] Gibt es mehrere auf eine Konfiguration $k \in K$ anwendbare Spielzüge, erhalten wir eine **Menge von Kindkonfigurationen** $N(k) \subset K$ mit $N(k) = \{r(k) \mid r \in R,\ r \text{ anwendbar auf } k\}$. Auf eine Endkonfiguration $k_f \in F$ sind keine Spielzüge anwendbar, da das Spiel mit Erreichen einer dieser Konfigurationen als beendet gilt: $N(k_f) = \varnothing$.

Die Menge K definieren wir nun induktiv über die Kindkonfiguration:

- k_0 ist Element von K.
- Wenn $k \in K$, dann auch alle $k' \in N(k)$.

Alle zulässigen Konfigurationen lassen sich also aus der Anfangskonfiguration und den legalen Spielzügen ableiten. Zu jeder Konfiguration $k \in K \;\; F$ gehört eindeutig eine Menge von Kindkonfigurationen $N(k)$ und zu jeder Konfiguration $k \in K \;\; k_0$ eine Elternkonfiguration k'. Diese Beziehungen können durch einen Graphen anschaulich dargestellt werden.

Ein **Spielgraph** ist ein gerichteter Graph $G(V, E)$ mit:

- Knoten $V = K$ und

[1] Das bedeutet nicht notwendigerweise, dass sich die Konfiguration der Spielelemente verändert. Zwei Spielkonfigurationen können sich auch darin unterscheiden, welcher Spieler an der Reihe ist.

- Kanten $E = \bigcup_{u \in K} \{(u, v) \mid v \in N(u)\}$

Als zusätzliche Vereinfachung schließen wir aus, eine über R aus k_0 generierte Konfiguration durch erneutes Anwenden legaler Spielzüge wieder erreichen zu können. Wir gelangen also im weiteren Spielverlauf nie zu einer Situation, die bereits aufgetreten ist.[2] Daraus folg unmittelbar, dass G zyklenfrei ist.

Damit ist der Graphen G insbesondere ein Baum[3], mit k_0 als Wurzel und F als Blätter. So kann über den Baum entlang legaler Spielzüge traversiert werden und bspw. ein kompletter Spielverlauf mit n Zügen als Pfad $(k_0, k_1, \dots, k_n \in F)$ subsequenter Kindkonfigurationen dargestellt werden.

ii. Am Beispiel Tic-Tac-Toe

Um obige Definitionen zu veranschaulichen, wenden wir sie nun auf das Spiel Tic-Tac-Toe an. Das Tic-Tac-Toe-Spielfeld besteht aus einem 3×3-Raster mit neun Feldern in die zwei Spieler abwechelnd ihre Zeichen setzen. Dies sind üblicherweise „X“ bzw. „O“. Im Folgenden wird Max mit „X“ und Min mit „O“ spielen.

Ziel jedes Spielers ist es, drei der eigenen Zeichen nebeneinander zu setzen, d.h. in einer Reihe, Spalte oder Diagonale, und gleichzeitig zu verhindern, dass der Gegner seine Zeichen in dieser Weise setzen kann. Das Spiel endet entweder, wenn einer der beiden Spieler gewinnt, sobald er dieses Ziel erreicht, oder wenn kein Zug mehr möglich ist, da alle neun Felder belegt sind. Das Spiel geht in diesem Fall unentschieden aus.

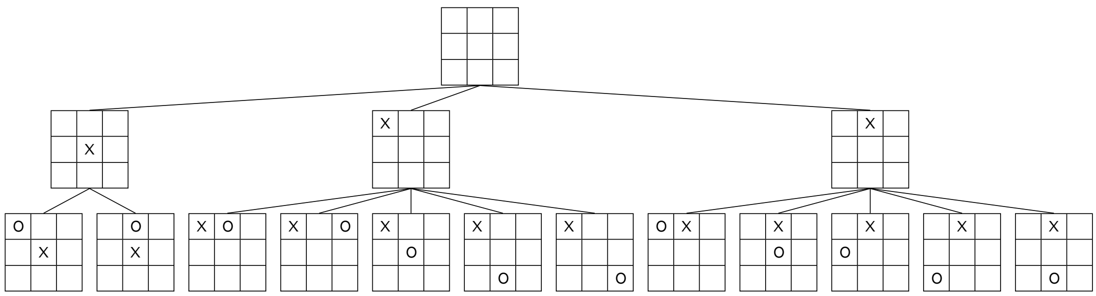

Abbildung 2: Ein Tic-Tac-Toe-Spielbaum der Tiefe 2, bei dem die Anzahl der Konfigurationen bereits durch Ausnutzung von Symetrien optimiert wurde.

nötig: Anwendung der Definition?

III. Zugplanung

Es stellt sich nun die Frage, wie sich aus den bekannten möglichen Zügen, die sich aus einer Spielsituation ergeben, der beste Zug auswählen lässt. Eine Möglchkeit für Max ist es, seine Züge so

[2] Diese Einschränkung schließt viele Spiele wie bspw. Schach von der folgenden Betrachtung aus, da dort durch legale Spielzüge Konfigurationen reproduziert werden können. Viele der Aussagen lassen sich dennoch auch auf diese Art Spiele übertragen.

[3] Wir benutzen also im Folgenden—abweichend der Terminologie der Hauptquelle[1]—, mit den hier getroffenen Annahmen, „Spielgraph“ synonym zu „Spielbaum“. Davon abzugrenzen ist der Begriff „Suchbaum“. Während der *Spielbaum* ein theoretisches Modell von großer (Speicher-)Komplexität ist, wird der *Suchbaum* zur Laufzeit generiert und bildet kein vollständiges Spiel ab.

zu wählen, dass sein Vorteil maximiert wird, wenn er am Zug ist sowie jenen Zug von MIN zu antizipieren, der ihm den größten Nachteil bringen wird, d.h. seinen Vorteil minimiert, da angenommen wird, dass MIN genauso handelt[4]. Dieses Vorgehen führt zum Minmax-Verfahren, das und eine dessen Optimierungen, das Alpha-Beta-Pruning, wir im Folgenden genauer betrachten.

i. Minmax-Verfahren

Zunächst führen wir eine **Gewinnfunktion** $g : F \to \mathbb{N}$ ein, die die Blättern des Spielbaumes bewertet und ihnen einen Wert zuordnet, wie günstig, dieser Ausgang für MAX wäre.

Des Weiteren benötigen wir zwei Typen von Knoten in unserem Spielbaum: **Max-Knoten** (im folgenden mit $\triangle$ gekennzeichnet), an denen MAX am Zug ist und der Nutzen maximiert werden soll und analog dazu **Min-Knoten** ($\bigtriangledown$). In unserem Tic-Tac-Toe-Szenario alternieren die Knoten-Typen mit jedem Halbzug[5], da die Spieler sich stets abwechseln.

Jedem Knoten u im Spielbaum wird nun ein **Minmax-Wert** $minmax(u) \in \mathbb{N}$, der dem Nutzen aus Sicht von MAX entspricht, zugeordnet. Dabei ergibt sich $minmax(u)$ rekursiv aus den Minmamax-Werten der Kindknoten $N(u)$ und kann wird mit einer Tiefensuche[6] wie folgt berechnet:

$$minmax(u) = \begin{cases} g(u) & \text{wenn } u \in F \\ \max\limits_{v \in N(u)} minmax(v) & \text{wenn } u \text{ Max-Knoten} \\ \min\limits_{v \in N(u)} minmax(v) & \text{wenn } u \text{ Min-Knoten} \end{cases}$$

Der rekursive Abstieg im Baum endet mit dem ersten Fall des Erreichens einer Endkonfiguration, damit, dass die Gewinnfunktion angewendet wird. Je nach dem in welcher Ebene (Min oder Max) des Baumes wir uns befinden wird beim folgenden Aufstieg entsprechend das Minimum bzw. Maximum der Kinder nach oben weitergegeben.

Schließlich wählt MAX den Zug mit dem größten Minmax-Wert, bzw. den Zug der zu genau dem Kindknoten führt, der vom Minmax-Algorithmus mit dem gleichen Minmax-Wert, wie der des aktuellen Knotens, annotiert wurde.

ii. Alpha-Beta-Pruning

IV. ZUSAMMENFASSUNG

Schlussfolgerung. Ausblick. Anwendbarkeit, Zufallsspiele

LITERATUR

[1] Sascha Klüppelholz. Entwurfs- und Analysemethoden für Algorithmen – Skript zur Vorlesung, SS 2016

[4] In Anlehnung an [2] tragen MAX und MIN auch genau aus diesem Grund ihre Namen: ist MAX an der Reihe, wird der Nutzen *maximiert*; ist es MIN, wird er *minimiert*; stets aus der Sicht von MAX.

[5] Ein Zug entspricht einer Runde, in der sowohl MAX und MIN einmal an der Reihe waren und besteht aus zwei Halbzügen.

[6] Als zusätzliche Optimierung, die sich auch aus weiteren Gründen anbietet, wie wir sehen werden, bietet sich dafür in der Praxis die Verwendung der *iterativen Tiefensuche* an.

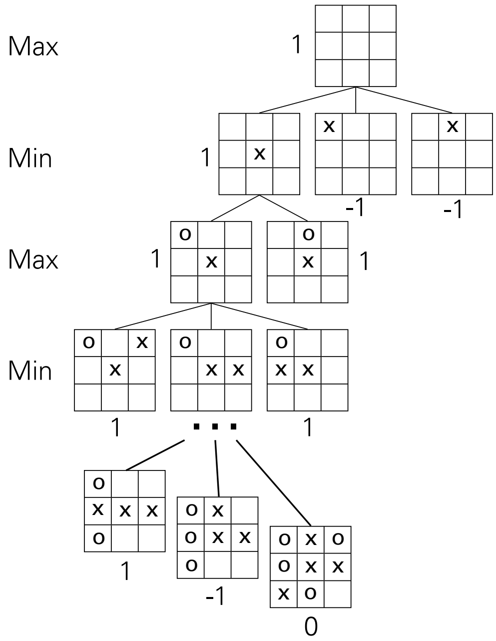

Abbildung 3: Ein Ausschnitts eines hypothetischen Spielbaumes an dem das Prinzip des Minmax-Verfahrens nachfollzogen werden kann: eine Gewinnfunktion bewertet Endkonfigurationen mit 1 bei Gewinn, -1 bei Niederlage und 0 bei Untenschieden; die Minmax-Werte werden entsprechend des Algorithmus' bis zur Wurzel propagiert; Max wählt schließlich den ersten Zug mit dem Kreuz in der Mitte, da der Minmax-Wert hier maximal ist.

[2] Russel, Norvig. Künstliche Intelligenz – Ein moderner Ansatz, 3., aktualisierte Auflage, 2012

Diese Ausarbeitung und zugehörige Präsentation sind auch auf github zu finden:
`https://github.com/foobar0112/tic`.