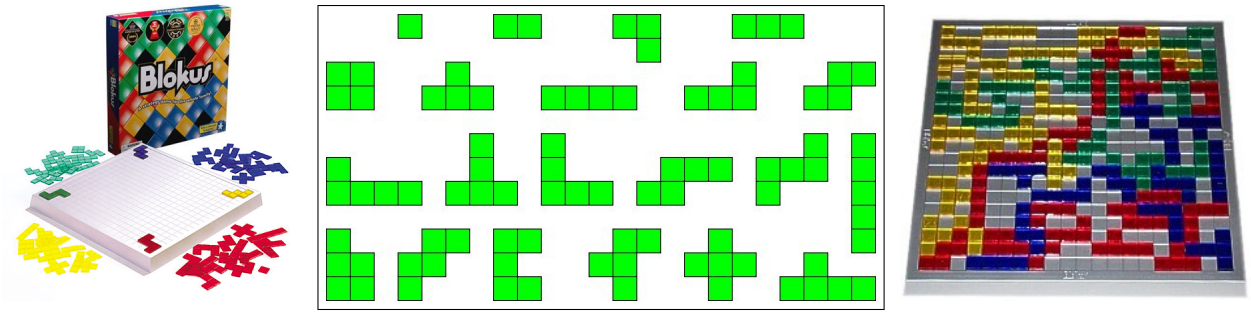
Intelligenter Blokus-Spieler

Bachelorpraktikum Knowledge Engineering

WS 09/10

1 Blokus

Blokus ist ein Brettspiel für 4 Spieler (gelb, rot, grün und blau). Aufgabe jedes Spielers ist es, 21 aus Quadraten bestehende Plättchen (die aus Tetris bekannten 12 Pentaminos, 5 Quadraminos, 2 Triminos, und je ein Domino und ein einzelnes Quadrat) auf einem quadratischen Spielfeld (20×20 Felder) abzulegen, wobei jedes neue Plättchen eines der eigenen an einer Ecke berühren muß, allerdings keins der eigenen an einer Kante berühren darf. Begonnen wird in den 4 Ecken des Spielfelds, die Spieler dürfen reihum je ein Plättchen ablegen. Gezogen wird so lange, bis kein Spieler mehr ablegen kann. Sieger ist, wer am meisten Quadrate abgelegt hat. Gelingt es einem Spieler, alle Plättchen abzulegen, gibt es 15 Bonuspunkte, wird das einzelne Quadrat im letzten Zug abgelegt, erhält man nochmals 5 Bonuspunkte.



(a) Das Spiel (b) 21 Plättchen für einen Spieler (c) Mögliche Endstellung

Figure 1: Blokus

Mehr Informationen finden Sie im Internet, z.B. unter `http://de.wikipedia.org/wiki/Blokus` oder `http://www.blokus.com`

2 Aufgabenstellung

Aufgabe jeder Gruppe ist es, einen intelligenten Spieler für dieses Spiel in Java zu implementieren. Die implementierten Spieler sollen dann am Ende in einem Turnier miteinander verglichen werden, sodaß auch ein entsprechendes Kommunikationsprotokoll gemeinsam erarbeitet und von allen Gruppen realisiert werden muß. Für die zu implementierende KI (Künstliche Intelligenz) gibt es drei mögliche Ansätze, die realisiert werden sollen (jedoch von unterschiedlichen Gruppen):

Multi-Player Minimax Der Minimax-Algorithmus ist der verbreiteste Such-Algorithmus für 2-Personen Spiele wie z.B. Schach. Eine Erweiterung auf 4 Spieler, wie hier erforderlich, ist relativ einfach. Ein Problem ist hier, daß er eine Bewertungsfunktion voraussetzt, die die Qualität einer Stellung für jede Seite einschätzt. Eine derartige Funktion muß von jeder Gruppe, die dieses Verfahren realisiert, erarbeitet werden.

Monte-Carlo Suche Monte-Carlo Suche spielt eine große Anzahl von Spielen zu Ende, und wählt am Ende den Zug aus, der am öftesten gewonnen hat (bzw. die im Durchschnitt größte Punkte-Anzahl erreicht hat). Dazu ist es wichtig, daß eine große Anzahl von Spielen durchgeführt werden kann, d.h. die Zugauswahl für jeden Spieler muß sehr schnell erfolgen können. Am einfachsten ist es, die Spieler zufällig ziehen zu lassen, jedoch sind auch andere Varianten denkbar.

UCT Suche Dieses Verfahren stellt eine Mischung zwischen den obigen beiden Verfahren dar. Es wird einerseits zufällig gesucht, wie in der Monte-Carlo Suche, jedoch wird auf den oberen Ebenen parallel dazu ein Spielbaum aufgebaut, der dafür sorgt, daß nicht alle Varianten gleich wahrscheinlich sind, und vielversprechendere Varianten öfter durchsucht werden.

Ziel des Praktikums ist es letztendlich, festzustellen, welches Suchverfahren sich am besten für dieses Spiel eignet.

3 Anforderungen

Muss

- korrekter Zuggenerator (Generieren aller möglichen Züge in einer Stellung und Möglichkeit zur Überprüfung der Korrektheit anderer Züge)
- eines der drei obigen Suchverfahren (nach Absprache mit den Betreuern, sodaß jedes Verfahren von zumindest einer Gruppe realisiert wird)
- Kommunikationsschnittstelle zur Übertragung der eigenen und Entgegennahme gegnerischer Züge (das Protokoll wird von allen Gruppen gemeinsam erarbeitet)
- gute Skriptbarkeit des Spielers für das Turnier, d.h. idealerweise sollten alle nötigen Parameter, wie Algorithmus, Serveradresse usw. per Kommandozeile übergebbar sein und keine weiteren Aktionen nötig sein
- beschränkte Laufzeit pro Zug

Soll

- effizienter Zuggenerator (z.B. durch Verwendung von Bitboards)
- GUI zur Visualisierung und manueller Zugeingabe (z.B. für das Spiel Mensch gegen KI)
- Variable Zeitkontrolle (Möglichkeit zur Vorgabe einer bestimmten Bedenkzeit)

Kann

- mehrere der angegeben KIs
- völlig eigene KI (nur zusätzlich zu einer angegeben)
- Verschiedene Schwierigkeitsstufen
- Schnittstelle zu einem On-line Spiele-Server
- . . . (seien Sie kreativ)

4 Vorkenntnisse

Die in der Aufgabenstellung zu implementierenden KI-Algorithmen werden in der Vorlesung *Einführung in die Künstliche Intelligenz* besprochen. Es ist daher von Vorteil, wenn Sie diese Vorlesung besucht haben (und wir werden auch Gruppen den Vorzug geben, die die entsprechenden Kenntnisse verfügen). Andererseits sind die Algorithmen auch nicht so komplex, daß die entsprechenden Kenntnisse nicht im Selbststudium erworben werden können. Darüberhinaus sind Vorkenntnisse in Java vom Vorteil.

5 Ansprechpartner

Johannes Fürnkranz (`juffi@ke.tu-darmstadt.de`)
Sang-Hyeun Park (`park@ke.tu-darmstadt.de`)