Welfenlab Competition
Schülerwettbewerb Informatik
http://www.gdv.uni-hannover.de/fuer_schueler/welfenlab_competition/

Nachdem in den letzten Jahren eher theoretische Themen wie Knoten und kürzeste Wege behandelt wurden, beschäftigen wir uns diesmal hauptsächlich mit künstlicher Intelligenz. Ihr sollt nämlich versuchen dem Computer Backgammon beizubringen.

(Bild: de.wikipedia.org)

Backgammon ist eines der ältesten Brettspiele der Welt, die heute gespielte Version basiert auf einer Variante aus dem Mittelalter (siehe Abbildung). Der Erfolg beim Spiel hängt zwar auch vom Glück, vor allem aber von einer guten Strategie ab. Ziel der Competition ist es ein Programm zu schreiben, das möglichst gut Backgammon spielt.
Unabhängig von viel Erfahrung, Ehre und Ruhm gibt es auch etwas Handfestes zu gewinnen:

1. Platz	2. Platz	3. Platz
		
Xbox 360	MP3-Player	Backgammon-Spiel

Wie bewertet man aber nun, ob ein Programm gut oder schlecht Backgammon spielt? Bei allen Spielen, die vom Zufall aber auch von strategischen Faktoren abhängen (wie z.B. Kniffel, alle Pokervarianten oder auch MauMau, Skat oder Doppelkopf), kann man nach einem Spiel nicht entscheiden ob man besser als sein Gegner ist oder nicht, da eben der Zufall eine Rolle spielt (bei Kniffel oder Backgammon der Würfel, bei MauMau, Pokern, Skat und Doppelkopf die Karten). Da jedoch der Zufall auf lange Sicht eine immer kleinere Rolle spielt (keiner kann sagen ob es heute oder morgen in Hannover regnet, im Durchschnitt fallen im Jahr aber 661 mm Niederschlag) gewinnt in z.B. 100000 Spielen der bessere Spieler mit fast an Sicherheit grenzender Wahrscheinlichkeit häufiger. Wir werden daher nach Abgabe der Programme ein Turnier veranstalten, in dem eure Programme mehrere Male gegeneinander antreten werden. In diesem Turnier werden die besten Programme ermittelt. Damit es nicht zu lange dauert, müsst ihr eine bestimmte Zeitspanne (Timeout) einhalten, d.h. euer Algorithmus muss innerhalb einer festen Zeit einen Zug berechnen können.

Es gibt bereits Programme, die sehr gut Backgammon spielen, teilweise sogar besser als jeder Mensch. Ein solches Programm zu entwickeln ist jedoch sehr schwer. Zunächst solltet ihr daher ein Programm schreiben, das die Regeln beherrscht und keine offensichtlichen Fehler macht. Habt ihr das geschafft, könnt ihr an dem Programm weiter feilen und seine Strategie nach und nach verbessern. Um dem Computer Backgammon beizubringen, müsst ihr es natürlich selber möglichst gut spielen können. Macht euch daher mit den Regeln vertraut und spielt ein paar Spiele gegen einen Freund/Freundin, eure Eltern oder sonst jemanden den ihr kennt. Vielleicht kennt ihr ja einen guten Spieler, der euch ein paar Tricks und gute Strategien verrät. Beachtet jedoch, dass es sehr schwierig sein kann eine Strategie, die ein Mensch ohne Probleme umsetzt, einem Computer beizubringen. Arbeitet euch also Schritt für Schritt vor. Die Regeln findet ihr z.B. unter http://de.wikipedia.org/wiki/Backgammon. Der sonst übliche Verdoppelungswürfel soll bei unserer Backgammon-Variante nicht berücksichtigt werden.

Damit eure Programme gegeneinander antreten können und ihr trotzdem freie Wahl in der Programmiersprache habt, brauchen wir ein Protokoll mit dem Spieldaten ausgetauscht werden können. Außerdem muss euer Programm in der Lage sein über das Internet (mit eben jenem Protokoll) mit einem von uns bereitgestellten Backgammon-Server zu kommunizieren.
Als einführendes Beispiel folgt ein Mitschnitt einer Unterhaltung mit unserem Backgammon-Server (Befehle vom Client in Fettdruck):

1	→	`WELCOME:Welcome to the Welfenlab Backgammon Server`
2	←	**`LOGIN:max;witwebolte`**
3	→	`CONFIRM:LOGIN;logged in as max`
4	→	`INFO:LOGIN;max`
5	←	**`JOIN:1;lehrerlaempel;300;1`**
6	→	`CONFIRM:JOIN;1`
7	→	`INFO:JOIN;1 max`
8	→	`INFO:JOIN;1 moritz`
9	←	**`START`**
10	→	`CONFIRM:START;`
11	→	`BOARD:0 -2 0 0 0 0 5 0 3 0 0 0 -5 5 0 0 0 -3 0 -5 0 0 0 0 2 0 0 0`
12	←	**`CONFIRM`**
13	→	`INFO:DICE;6 4`
14	→	`INFO:TURN;4 19 23;6 17 23`
15	→	`BOARD:0 -2 0 0 0 0 5 0 3 0 0 0 -5 5 0 0 0 -2 0 -4 0 0 0 -2 2 0 0 0`
16	←	**`CONFIRM`**
17	→	`DICE:3 4`
18	←	**`TURN:4 24 20;3 24 21`**
19	→	`BOARD:0 -2 0 0 0 0 5 0 3 0 0 0 -5 5 0 0 0 -2 0 -4 1 1 0 -2 0 0 0 0`
20	←	**`CONFIRM`**
21	→	`INFO:DICE;2 3`
		⋮
423	→	`BOARD:13 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2 -2 -1 -3 0 0 7 0`
424	←	**`CONFIRM`**
425	→	`DICE:4 3`
426	←	**`TURN:4 2 0;3 2 0`**
427	→	`ENDGAME:WIN;0;3 0`
428	→	`INFO:ENDGAME;1 max 0 3 0`
429	→	`CONFIRM:JOIN;-1`
430	→	`INFO:JOIN;-1 max`
431	→	`INFO:JOIN;-1 moritz`
432	←	**`LOGOUT`**

Nach der Begrüßung durch den Server (Zeile 1) meldet sich Max zunächst mit seinem Passwort an (Zeile 2) und erhält eine entsprechende Bestätigung vom Server (Zeilen 3 und 4). Der Server informiert alle angemeldeten Benutzer stets über Benutzeranmeldungen (Zeile 4), Beteiligungen an Spielen (Zeilen 7, 8 430 und 431), den Zügen des Gegners (Zeilen 13 und 14) und Spielergebnissen (Zeile 428).
Max erzeugt ein neues Spiel mit einem Timeout von 300 Sekunden und nur einer Runde (Zeile 5). Möchte ein anderer Spieler diesem Spiel beitreten, so muss er das von Max gewählte Passwort (lehrerlaempel) kennen. In Zeile 8 sieht man, dass Moritz dem Spiel erfolgreich beigetreten ist.
Nachdem beide Spieler das Spiel gestartet haben (Zeile 9), wird das Brett übermittelt (Zeile 11). Die erste Zahl in dieser Liste entspricht der Anzahl der ausgewürfelten Spielsteine. Danach folgt die Belegung der 24 Spielfelder. Eine positive Zahl bezeichnet dabei die Anzahl der eigenen Spielsteine, negative Zahlen bezeichnen Spielsteine des Gegners. Gezogen wird immer von Rechts nach Links. (Der Gegner zieht natürlich von Links nach Rechts.) Es folgen die eigenen Spielsteine auf der Bar, die ausgewürfelten Spielsteine des Gegners und die Spielsteine des Gegners auf der Bar. Das Board muss stets vom Client bestätigt werden (Zeile 12).
Zunächst ist Moritz an der Reihe. Er zieht von Feld 19 auf Feld 23 und von Feld 17 auf Feld 23. Max wird darüber nur informiert (Zeilen 13 und 14). Nach jedem Zug wird wieder das Brett vom Server übermittelt. Nun ist Max an der Reihe. Der Server hat eine 3 und eine 4 gewürfelt (Zeile 17). Moritz zieht von Feld 24 auf Feld 20 und von Feld 24 auf Feld 21 (Zeile 18).
Nach vielen weiteren Zügen macht Max den letzten Zug (Zeile 426). Der Server informiert über seinen Gewinn (Zeilen 427 und 428) und beendet das Spiel (Zeilen 429 bis 431).

Dieses einführende Beispiel soll nur einen ersten Eindruck vom Spielablauf geben. Weitere Informationen zu unserem Backgammon-Server und eine vollständige Beschreibung des Protokolls gibt es unter http://www.gdv.uni-hannover.de/fuer_schueler/welfenlab_competition/.

Aufgabenstellung
Es soll ein Backgammon-Programm entwickelt werden. In der dazugehörigen Dokumentation sollen Algorithmen und Programmaufbau verständlich dargestellt werden. Folgende Teilaufgaben sollen gelöst werden:

1. Erstelle eine grafische Oberfläche, die es zwei menschlichen Spielern ermöglicht gegeneinander Backgammon zu spielen. Das Programm soll dabei keine irregulären Züge zulassen und merken wann das Spiel zu Ende ist und wer gewonnen hat.

2. Erweitere dein Programm um eine Schnittstelle, die es ermöglicht mit dem obigen Protokoll über ein Netzwerk mit dem Backgammon-Server zu kommunizieren.

3. Entwickle eine künstliche Intelligenz, die sowohl gegen einen menschlichen Gegner als auch über das Internet über den Backgammon-Server gegen andere Computer spielen kann. Folgendes Vorgehen kann dabei zweckmäßig sein:

   - Bring dem Computer die Regeln des Spiels bei. Er darf also keine irregulären Züge machen.
   - Teste deinen Algorithmus indem du gegen ihn spielst und achte auf seine Schwächen. Der Computer dürfte z.B. anfangs sehr leicht auszutricksen sein.
   - Versuche diese Schwächen zu eliminieren indem du gezielt am Algorithmus Verbesserungen vornimmst.

   Versuche auf diese Weise die künstliche Intelligenz so stark wie irgend möglich zu machen. Achte z.B. auch darauf, dass es für nahezu jeden Würfelwurf eine passende Eröffnung gibt, die strategische Vorteile hat. Dein Algorithmus sollte mit unterschiedlichen Timeouts funktionieren. Für das Turnier wird der Timeout relativ niedrig sein (wenige Sekunden), gegen einen menschlichen Spieler könnte sich das Programm natürlich mehr Zeit lassen.

Teilnahmebedingungen

- Anmeldeschluss ist der 15.12.2006.
- Gruppenmeldungen sind nicht möglich.
- Als Programmiersprache sind C, C++, Pascal/Delphi, PHP und Java zugelassen. Die Sprache muss bei der Anmeldung mit angegeben werden. Für die Client-Server-Kommunikation empfehlen wir die SDL_net-Bibliothek (http://www.libsdl.org), ihr könnt aber natürlich auch die der Programmiersprache beiliegenden Pakete benutzen. Es dürfen nur die Standard-Bibliotheken und frei verfügbare GUI-Frameworks wie z.B. wxWindows, Qt oder Gtk verwendet werden.
- Es muss eine etwa 10-seitige Ausarbeitung angefertigt werden, in der die benutzten Algorithmen erklärt werden und in der das Programm ausführlich dokumentiert wird. Außerdem müssen alle Quellen und Hilfsmittel angegeben werden.
- Das erstellte Programm und die Ausarbeitung sind spätestens bis zum 12.03.2007 bei uns einzureichen.
- Es dürfen nur Schülerinnen und Schüler der Sekundarstufe I oder II allgemeinbildender Schulen aus Niedersachsen an dem Wettbewerb teilnehmen. Familienangehörige von Mitarbeitern des Fachgebietes Graphische Datenverarbeitung an der Universität Hannover sind leider ausgeschlossen.
- Der Rechtsweg ist ausgeschlossen.

Bewertungskriterien

- Lauffähigkeit und Korrektheit des Programms
- Gut verständliche Dokumentation der Algorithmen
- Besondere eigenständige Ideen
- Turnierergebnis
- In Zweifelsfällen: Strukturierter Programmierstil
- ...

Anmelden kannst du dich unter http://www.gdv.uni-hannover.de/fuer_schueler/welfenlab_competition/. Solltest du es dann nicht schaffen, dein Programm rechtzeitig abzugeben, verfällt deine Anmeldung.

So, das war's erstmal. Hoffentlich hast du ein wenig Lust bekommen, bei diesem Wettbewerb mitzumachen. Es geht bei dieser Competition nicht darum, alles perfekt zu machen. Wir freuen uns auch über Teillösungen. Wenn wir merken, dass du dich mit der Problematik beschäftigt hast, hier und da ein paar interessante eigene Ideen hattest und deine Gedanken und Algorithmen dokumentierst, hast du gute Chancen unter den drei Besten zu sein. Bei Rückfragen kannst du dich gerne bei uns melden.
E-Mail: competition@gdv.uni-hannover.de

Viel Erfolg wünschen
Prof. Dr. F.-E. Wolter und das Team der Welfenlab Competition!