Selbst-organisierende, adaptive Systeme (WS 16/17)

Übungsblatt 06 (Bearbeitung bis: 30.11.2016, 23:59 Uhr)

Extensive Spiele & Minimax

Aufgabe 1 *(Tic-Tac-Toe/Minimax)*

In dieser Aufgabe implementieren Sie verschiedene KI-Strategien, die Tic-Tac-Toe spielen. Dazu ist für Sie ein Code-Gerüst vorbereitet, das sowohl eine GUI inkl. "interaktivem" Modus (starten Sie dazu `isse.control.Controller`) als auch eine reine Kommandozeilen-API (siehe z.B. `test.TestGameEngine`) bereitstellt. Testen Sie beide Anwendungen, und analysieren Sie die Funktionsweise der *naiven* Strategie (`isse.model.strategies.NaiveStrategy`).

a) Machen Sie sich zunächst mit der API vertraut, indem Sie einen zufällig spielenden Agenten entwickeln. Zeilen sind von oben nach unten aufsteigend von 0 bis 2 nummeriert, Spalten von links nach rechts. Orientieren Sie sich an der Klasse `NaiveStrategy` und hängen Sie die Strategie für den interaktiven Modus in `StrategyProvider` ein. Entwickeln Sie in Analogie zu `test.TestGameEngine` einen Testfall, der n Spiele gegen den naiven Algorithmus erlaubt. Welche Strategie gewinnt häufiger?

b) Versuchen Sie sich nun an einem etwas schlaueren Agenten. Implementieren Sie einen rein *reaktiven Agenten* (siehe VL 3), welcher einfache Regeln befolgt wie etwa: "Wenn ich in Feld (i,j) gewinnen kann, setze ich (i,j).". Entwickeln Sie weitere sinnvolle Regeln und dokumentieren Sie diese. Verliert diese Strategie noch gegen den Zufall? Können Sie sie im interaktiven Modus persönlich noch schlagen?

c) Implementieren Sie den Minimax-Algorithmus zur Bestimmung der optimalen Strategie. Wie kommen Sie von Spielbrettern zu Nutzenfunktionen? Wie können Sie auf ein Nullsummenspiel abbilden? Überlegen Sie sich, wie Sie den Algorithmus isoliert von der Tic-Tac-Toe-Anwendung testen können, um ihn ggF. für ein anderes Spiel wiederverwenden zu können. Was passiert, wenn 2 Minimax-Strategien gegeneinander antreten? Kann eine der vorherigen Strategien noch gewinnen? Mit wie vielen Knoten im Suchbaum müssen Sie asymptotisch rechnen?

d) Implementieren Sie Alpha-Beta-Pruning, um den Suchbaum zu verringern. Instrumentieren Sie den Code[1], um die Effekte (reduzierte Knoten) zählbar zu machen. Wirkt sich diese Verbesserung auf die Laufzeit aus?

e) In größeren Spielen ist es nicht mehr möglich, den gesamten Suchbaum zu explorieren. In diesem Fall muss die maximale Höhe begrenzt werden und somit erreicht man nicht immer einen Blattknoten, an dem der Wert abgelesen werden kann. Stattdessen bedient man sich Heuristiken, die anzeigen, "wie gut" eine Situation ist. Geben Sie solche Heuristiken für Tic-Tac-Toe, aber auch Vier-Gewinnt an.

Aufgabe 2 *(Extensive Spiele)*

Die folgenden Aufgaben dienen dazu, die in der Vorlesung behandelten theoretischen Konzepte zu extensiven Spielen anzuwenden.

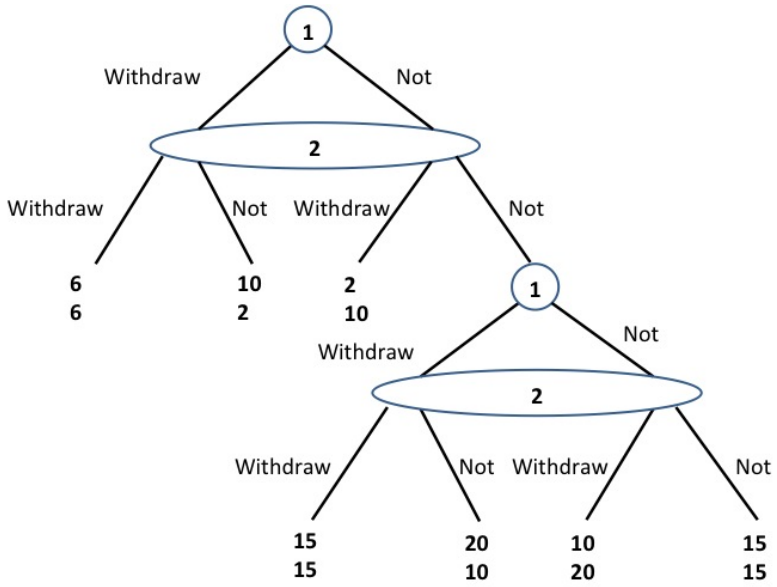
a) Zeigen Sie, dass es kein Normalformspiel geben kann, in dem es keinen pareto-optimalen Ausgang gibt.

[1] Durch das Einführen von Zählvariablen etc.

b) Angenommen, zwei Spieler teilen 50 (gleichwertige) Münzen auf. Der Nutzen von Spielern ist die Menge von erhaltenen Münzen. Zuerst kann Spieler 1 die Münzen aufteilen, danach Spieler 2 den Anteil wählen. Bestimmen Sie ein Nash-Gleichgewicht mittels Rückwärtsinduktion.

c) Betrachten Sie das folgende Spiel

- 2 Spieler haben 10 € bei einer Bank investiert.
- Die Bank kann damit in ein gemeinsames Projekt investieren.
- Spieler können zu zwei Zeitpunkten T_1 und T_2 ihre 10€ (und die Hälfte des verbleibenden Gewinns) abheben (W, *withdraw*) oder nicht (N).
- T_1 liegt *vor* der Fertigstellung, aus den 20 € können lediglich 12 € erwirtschaftet werden.
- T_2 liegt *nach* Abschluss, es können 30 € erarbeitet werden.
- Wählt z.B. Spieler 1 die Abhebeoption zu T_2 und Spieler 2 nicht, erhält er seine 10€ Einsatz + die Hälfte des Überschusses (momentan 20). Spieler 2 erhält die andere Hälfte (und hat einen "Schuldschein" von 10€ – sein Einsatz – bei der Bank).



Nennen Sie ein teilspielperfektes Equilibrium dieses Spiels.

d) Begründen Sie, warum in einem extensiven Spiel mit perfekter Information ein durch Rückwärtsinduktion gefundener Ausgang ein Nash-Gleichgewicht sein muss.

e) Mithilfe von Informationsmengen können Normalformspiele (NF) auch als extensives Spiel mit unvollständiger Information (UVEF) modelliert werden (NF $\rightarrow$ UVEF). Umgekehrt liefert die induzierte Normalform eine Abbildung (UVEF $\rightarrow$ NF). Achten Sie auf Informationsmengen.

- Stellen Sie Fußball/Komödie als UVEF-Spiel dar.
- Stellen Sie zweifaches Gefangenendilemma als UVEF-Spiel dar.