Künstliche Intelligenz

Übungsblatt #1
Modellierung & Suche

Prof. Dr. J. Fürnkranz, Dr. G. Grieser

Aufgabe 1.1

Wir betrachten folgende Welt:

Welt: Die Welt der Staubsauger-Akteure besteht aus Räumen, die rasterförmig angelegt sind. Jeder Raum ist entweder sauber oder dreckig und das Saugen von dreckigen Räumen reinigt sie zuverlässig. Versucht der Akteur die Welt zu verlassen (d.h. er läuft gegen eine Wand), so bleibt er stehen und der Berührungssensor spricht an.

Wahrnehmungen: Ein Staubsauger-Akteur bekommt in jedem Schritt einen dreielementigen binären Wahrnehmungsvektor . Das erste Element ist ein Berührungssensor, der bei einer Kollision *wahr* liefert. Ein Fotosensor liefert *wahr*, falls der aktuelle Raum dreckig ist und ein Infrarotsensor liefert *wahr*, falls der Akteur sich an in seinem Ausgangsraum befindet. Die Eingabe der Sensorwerte in das System geschieht, indem die Variable `sensor` mit einem Array der Größe 3 belegt ist.

Aktionen: Ein Staubsauger-Akteur hat für jeden Schritt folgende Möglichkeiten: `vorwärts` (geradeaus in den nächsten Raum gehen bzw. stehenbleiben, falls er vor einer Wand steht), `drehen(`*`Gradzahl`*`)` (*Gradzahl* Grad nach rechts drehen), `saugen` und `abschalten`.

Programm: Wir beschreiben die Programme in einer Pseudonotation.

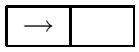
Die aktuellen Werte der Sensoren können

Das Programm des Akteurs besteht aus einer Liste von Anweisungen der Form

```
Fakten/Wahrnehmungen --> Aktion/NeuerFakt.
```

Die Anweisungen werden der Reihe nach bearbeitet. Die Regeln werden von oben nach unten durchgesehen. Wenn alle Fakten der linken Seite einer Anweisung wahr sind bzw. die erwähnten Wahrnehmungen gemacht werden, wird die rechte Seite ausgeführt, ansonsten wird mit der nächsten Regel weitergemacht. Wenn die rechte Seite eine Aktion ist, so wird diese ausgeführt und der neue Wahrnehmungsvektor berechnet (d.h. ein Schritt in der Welt gemacht). Ist die rechte Seite ein Fakt, so wird dieser Fakt als wahr gespeichert und die Regelliste von vorn durchsucht.

a) Das Beispiel aus der Vorlesung
"Wenn der aktuelle Raum schmutzig ist, dann sauge, ansonsten gehe in den anderen Raum"
für die 2-Raum-Welt



(der Staubsauger mit seiner Blickrichtung ist als Pfeil dargestellt) ist beispielsweise wie folgt:

```
sensor[2]==wahr --> sauge      % saugen, wenn der Raum dreckig ist
sensor[1]==wahr --> drehe(180) % umdrehen, falls wir vor einer Wand stehen
--> vorwärts                   % in den anderen Raum gehen
```

(a) Analysieren Sie diesen reflexbasierten Agenten.

(b) Ändern Sie ihn so, daß zunächst aus den Sensordaten eine interne Weltbeschreibung erzeugt wird, die dann (anstelle der direkten Sensordaten) als Bedingungen in den Condition-Action-Rules benutzt werden.

(c) Was passiert, wenn man diesen Agenten in eine 2x2-Welt setzt? Ändern Sie ihn so, daß er auch dort "vernünftig" funktioniert.

(d) Was passiert in einer 3x3-Welt? Wie könnte hier ein reflexbasierter Agent aussehen?

b) Die Welt unseres Staubsauger-Akteurs bestehe nun aus einem 2×2 Schachbrett von Räumen, mit dem Raum links oben als Ausgangsposition.

1	2
3	4

Das Ziel eines Staubsauger-Akteurs nach seiner Aktivierung ist die schnellste Reinigung seiner Welt, d.h. in der kleinsten Anzahl von Schritten. Kollisionen sollen dabei möglichst vermieden werden. Um unnötige Störungen zu vermeiden, soll nur dort gesaugt werden, wo auch Dreck ist. Am Ende soll eine Selbstabschaltung an der Ausgangsposition erfolgen.

(a) Der Roboter schaue zunächst in Richtung Raum 2 und der einzige dreckige Raum sei Raum 3. Geben Sie das Programm an, das das Ziel des Akteurs am besten erfüllt.

(b) Geben Sie ein allgemeines Programm an, das das Ziel des Akteurs am besten erfüllt.

c) Verändern Sie den Roboter so, daß man ihn in eine *beliebige* (aber endliche) Welt setzen kann. Zu jedem Zeitpunkt gibt es einen Pluspunkt pro sauberem Raum, einen Minuspunkt pro schmutzigen Raum und (da diese dann neu gestrichen werden muß) 10 Minuspunkte pro Wand, gegen die gefahren wurde.

(a) Konzipieren Sie einen Roboter, der sich rational in dieser Welt verhält.

(b) Was ändert sich, wenn Sie zusätzlich noch einen Minuspunkt pro Bewegung (*vorwärts, drehen, saugen*) erhalten?

Aufgabe 1.2

Gegeben ist ein Spielbrett mit 7 Feldern, auf dem 3 weiße und drei schwarze Steine verteilt sind (auf jedem Feld darf nur maximal ein Stein liegen):

Ein Stein darf nun wie folgt bewegt werden:

- Ein Stein darf in ein benachbartes leeres Feld gezogen werden. Kosten: 1.
- Ein Stein darf über einen oder zwei andere Steine in ein leeres Feld gezogen werden. Kosten: Anzahl der übersprungenen Steine.

Das Ziel ist nun, die Ausgangssituation

w	s	w	w	s	s	

mit minimalen Kosten in die Zielkonfiguration

w	w	w		s	s	s

zu bringen, d.h. links alle weißen und rechts alle schwarzen Steine und das Loch genau in der Mitte zu haben.

a) Wenden sie die unterschiedlichen uninformierten Suchverfahren auf das Problem an, jeweils einmal mit und ohne *closed list*. Wenn Sie einen Knoten expandieren erzeugen Sie die Nachfolgeknoten jeweils so, daß zunächst der am weitesten links stehende Stein bewegt wird, dann der zweitlinkeste usw.

b) Welche Heuristik würden Sie für eine informierte Suche verwenden? Ist Ihre Heuristik admissible und/oder consistent?

Lösungsvorschlag:

Es bieten sich die beiden Heuristiken h_{MIS} und h_{MAN} aus der Vorlesung an. h_{MAN} ist jedoch nicht konsistent, da die Kosten für das Überspringen eines einzelnen Steines 1 betragen, die Heuristik jedoch hier den Wert 2 berechnet. $\rightarrow h_{MAN}$ ist für diese Aufgabe nicht geeignet.

h_{MIS} dagegen ist admissible und konsistent. (Begründung analog zu nächsten Teilaufgabe.)

c) Wenden Sie den A^*-Algorithmus an.

Lösungsvorschlag:

Als Heuristik verwenden wir h_{MIS}, d.h. die Anzahl falsch platzierter Steine.

Wir beginnen mit | w | s | w | w | s | s | | und expandieren diesen Knoten. Als Nachfolger ergeben sich:

w	s	w		s	s	w	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w	w		s	s	$g(n) = 1, h_{MIS} = 2 \rightarrow f(n) = 3$

w	s	w	w	s		s	$g(n) = 1, h_{MIS} = 2 \rightarrow f(n) = 3$

Da wir nun zwei Knoten mit Bewertung 2 haben, wählen wir einen aus, z.B.

w	s	w	w		s	s	$g(n) = 1, h_{MIS} = 2 \rightarrow f(n) = 3$

Dessen Expansion ergibt

w		w	w	s	s	s	$g(n) = 3, h_{MIS} = 1 \rightarrow f(n) = 4$
w	s		w	w	s	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w		w	s	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w	w	s		s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w	w	s	s		$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$

Da der vorletzte Knoten bereits mit geringeren bisherigen Kosten (d.h. $g(n) = 1$ statt 2 erzeugt wurde, wird er geprunt und unsere Openlist besteht also aus

w	s	w		s	s	w	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w	w	s		s	$g(n) = 1, h_{MIS} = 2 \rightarrow f(n) = 3$
w		w	w	s	s	s	$g(n) = 3, h_{MIS} = 1 \rightarrow f(n) = 4$
w	s		w	w	s	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w		w	s	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w	w	s	s		$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$

Wir wählen den Knoten mit dem geringsten Wert, also

w	s	w	w	s		s	$g(n) = 1, h_{MIS} = 2 \rightarrow f(n) = 3$

Dessen Expansion ergibt

w	s		w	s	w	s	$g(n) = 3, h_{MIS} = 3 \rightarrow f(n) = 6$
w	s	w		s	w	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w	w		s	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w	w	s	s		$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$

Auch hier werden wieder der vorletzte und der letzte Knoten ignoriert, unsere Openlist ist nun:

w	s	w		s	s	w	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w		w	w	s	s	s	$g(n) = 3, h_{MIS} = 1 \rightarrow f(n) = 4$
w	s		w	w	s	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w		w	s	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s	w	w	s	s		$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$
w	s		w	s	w	s	$g(n) = 3, h_{MIS} = 3 \rightarrow f(n) = 6$
w	s	w		s	w	s	$g(n) = 2, h_{MIS} = 2 \rightarrow f(n) = 4$

Wählen wir nun wieder zufällig einen unter den Knoten mit der Bewertung 4 aus, sagen wir

w		w	w	s	s	s	$g(n) = 3, h_{MIS} = 1 \rightarrow f(n) = 4$

Dessen Expansion ergibt

	w	w	w	s	s	s	$g(n) = 4, h_{MIS} = 1 \rightarrow f(n) = 5$
w	w		w	s	s	s	$g(n) = 4, h_{MIS} = 1 \rightarrow f(n) = 5$
w	w	w		s	s	s	$g(n) = 4, h_{MIS} = 0 \rightarrow f(n) = 4$
w	s	w	w		s	s	$g(n) = 5, h_{MIS} = 2 \rightarrow f(n) = 7$

Der Knoten | w | w | w | | s | s | s | ist ein Endknoten, wir sind jedoch im allgemeinen noch nicht fertig!

Wir wissen jetzt, daß das Ziel höchstens mit Kosten 4 erreicht werden kann. Es gibt zwar noch Knoten in unserer Openlist, diese sind jedoch alle ebenfalls mit 4 oder höher bewertet. Da unsere Heuristik admissible ist, d.h. die realen Kosten unterschätzt, werden jedoch alle noch offenen Knoten mindestens die Kosten 4 verursachen und wir sind tatsächlich fertig.

Im allgemeinen jedoch kann der Fall eintreten, daß in der Openlist noch Knoten mit Kosten $<$ unseren gefunden Zielkosten enthalten sind. In diesem Fall müssen die auch noch untersucht werden, da es ja noch einen kürzeren Weg geben kann.

Aufgabe 1.3

a) Beweisen Sie, daß die Heuristiken h_{MIS} und h_{MAN} aus der Vorlesung consistent sind.

Lösungsvorschlag:

Die Heuristik h_{MIS} zählt die Anzahl der falsch platzierten Steine. Mit jedem Zug kann maximal ein Stein mehr in die richtige Position gebracht werden, d.h. es gilt $h_{MIS}(n) \leq h_{MIS}(n') + 1$, falls n' eine Nachfolgeposition von n ist. Auf der anderen Seite sind die Kosten für einen Zug (d.h. die Bewegung eines Steines) jeweils 1, d.h. es gilt $h_{MIS}(n) \leq c + h_{MIS}(n')$ für alle Nachfolgepositionen n'.

Die Manhattan-Distanz h_{MAN} zählt die Anzahl der Felder , die jeder Stein minimal überqueren muß, um zur Zielposition zu gelangen. Auch hier gilt wieder, daß mit jedem Zug sich die Manhattan-Distanz maximal um 1 verringern kann, d.h. $h_{MAN}(n) \leq h_{MAN}(n') + 1$, ergo $h_{MAN}(n) \leq c + h_{MAN}(n')$.

b) Beweisen Sie, daß die Kombination zweier admissible Heuristiken mittels $\max$ wieder admissible ist.

Lösungsvorschlag:

Seien h_1 und h_2 zwei admissible Heuristiken, d.h. es gilt $h_1(n) \leq h^*(n)$ und $h_2(n) \leq h^*(n)$, für alle Zustände n. Daraus folgt direkt $\max\{h_1(n), h_2(n)\} \leq h^*(n)$ für alle Zustände n. Somit ist die Heuristik $h_3(n) = \max\{h_1(n), h_2(n)\}$ ebenfalls admissible.