Technische Universität Berlin
Fakultät IV - Elektrotechnik und Informatik

Künstliche Intelligenz: Grundlagen und Anwendungen

WS 2013/2014

Albayrak, Fricke (AOT) – Opper, Ruttor (KI)

Schriftlicher Test - Teilklausur 1

Musterlösung
14.12.2013

Aufgabe 1 – Zuordnungsproblem als Suchproblem

Ein Unternehmen eröffnet 4 Baustellen, auf jede Baustelle muss ein Kran. Das Unternehmen besitzt 4 Kräne an unterschiedlichen Standorten. Die folgende Entfernungstabelle zeigt die Entfernungen der Kräne {1,2,3,4} zu den Baustellen {A,B,C,D}.

Baustelle Kran	A	B	C	D
1	60	45	45	50
2	5	55	25	35
3	95	65	60	75
4	15	80	65	85

Gesucht ist eine Zuordnung der 4 Kräne auf die 4 Baustellen, sodass die zu fahrende Gesamtdistanz minimiert wird.

a) Repräsentieren Sie dieses Problem als Suchproblem. Wählen Sie eine möglichst formale Notation und begründen Sie Ihre Designentscheidungen.

Zustand (optional): (A,B,C), wobei A eine Liste noch nicht platzierter Kräne ist, B die Menge der noch nicht berücksichtigten Baustellen und C eine Menge von Paaren (x,y), wobei x∈{1,2,3,4} und y∈{A,B,C,D}. Jedes Paar repräsentiert einen Kran x auf Baustelle y.

Man könnte auch eine Menge zu platzierender Kräne verwenden, aber das hat negativen Einfluss auf den Verzweigungsgrad.

Startzustand S_0: ([1,2,3,4], {A,B,C,D}, { })
Zielzustand: S_Z: ([], { }, M) // M enthält alle Paare

Aktion: Bewege den nächsten Kran x vom Ausgangsstandort zur Baustelle y, sofern y noch nicht mit einem Kran belegt ist. Aktionskosten gemäß Tabelle.

Sei S = ([x|Rest], K, Z);

x das erste Element und Rest die übrigen Elemente der Liste noch nicht platzierter Kräne im 1. Tupelelement; K die Menge der noch nicht platzierten Kräne; Z die Menge der Tupel (a,b) bereits platzierter Kräne a auf Baustellen b.

move(x, y): ([x|Rest], K, Z) --> (Rest, K \ y, Z ∪ (x,y))

anwendbar, wenn erstes Tupelelement keine leere Liste ist.

1. Alternative ohne Menge noch nicht platzierter Kräne: Zustand (optional): Menge von Paaren (x,y), wobei x∈{1,2,3,4} und y∈{A,B,C,D}. Jedes Paar repräsentiert einen Kran x auf Baustelle y.
Startzustand S_0: { }
Zielzustand: S_Z: M, mit |M| = 4 (alle Kräne zugewiesen)
Aktion: Bewege Kran x vom Ausgangsstandort zur Baustelle y, sofern y noch nicht mit einem Kran belegt ist. Aktionskosten gemäß Tabelle.
move(x, y): S --> T

anwendbar wenn ∀z (x,z) ∉ S und ∀z (z,y) ∉ S (weder ist x einer Baustelle zugewiesen, noch steht irgend ein Kran auf der Baustelle y

T = S ∪ (x,y)

2. Alternative: Man könnte auch den Ausgangsstandort, z.B. "G" für Garage, verwenden, dann wäre der Startzustand keine leere Menge:

S_0: { (1,G), (2,G), (3,G), (4,G) }

S_Z: { (1,w), (2,x), (3,y), (4,z) } mit w≠x≠y≠z≠G

und einer Aktionsbeschreibung, die ein (x,G) aus dem aktuellen Zustand in ein (x,y) transformiert, wobei y≠G und nicht bereits im Zustand enthalten ist.

3. Oder man modelliert die noch zu platzierenden Kräne und verfügbaren Baustellen als separate Mengen, aus denen man jeweils ein Paar verwendet:

S_0: ({1,2,3,4}, {A,B,C,D}, {})

S_Z: { ({}, {}, X }

und einer Aktionsbeschreibung, die im Zustand (A,B,C) ein x∈G und ein y∈B wählt (entfernt) und als (x,y) in C einfügt.

Man kann auch die Aktionskosten in den Zustand mit einbringen. Oder Listen anstelle von Mengen verwenden.

b) Charakterisieren Sie den (zyklenfreien) Suchbaum hinsichtlich Verzweigungsgrad und Tiefe. Geben Sie exakte Zahlen an oder – wenn dies nicht einfach möglich ist – einen begründeten Schätzwert.

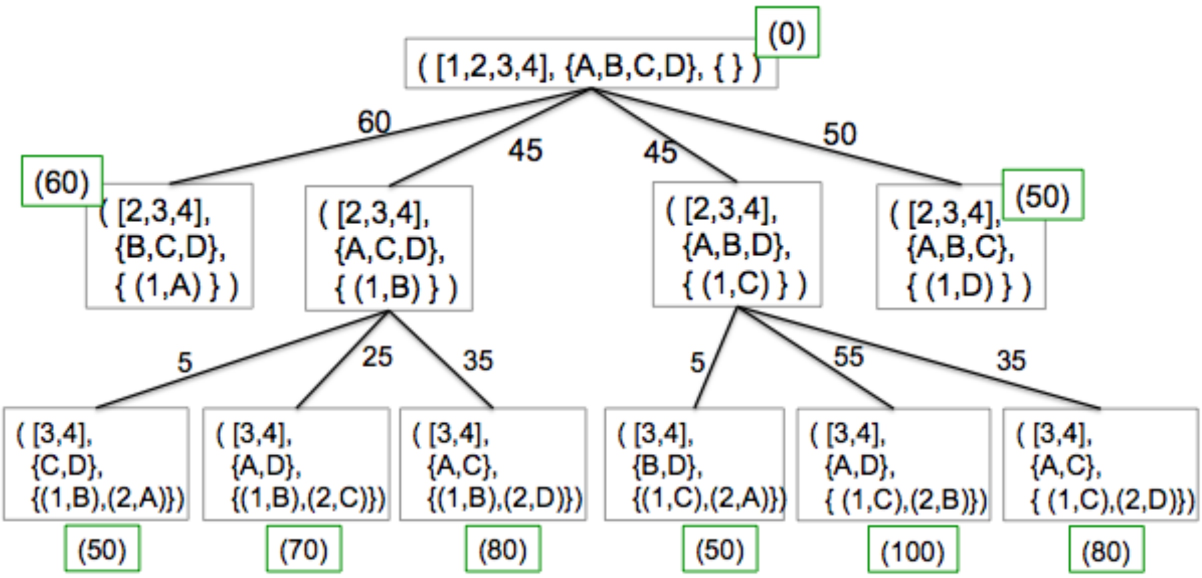
Tiefe = 4, denn nach 4 Aktionen ist zwingend der Zielzustand erreicht. Zyklen können in der gewählten Repräsentation nicht auftreten.
Breite = 4, denn in jeder Aktion ist der zu platzierende Kran eindeutig bestimmt, während (bei der ersten Aktion) 4 Baustellen als Zielorte zur Wahl stehen.

1. Alternative: Tiefe = 4, aber Breite = 4x4, weil sowohl Kran als auch Zielort beliebig gewählt werden können. Der resultierende Suchbaum generiert dann alle Permutationen von Kränen auf Baustellen, was eine deutlich größere Komplexität zur Folge hat.

c) Welches Suchverfahren würden Sie einsetzen?

Branch & Bound (alternativ: A* mit Trivialheuristik h=0 in allen Zuständen) findet die optimale Lösung.

d) Notieren Sie den ersten Pfad der Länge 2, den der in c) benannte Suchbaumalgorithmus expandiert.
Pfade der Länge 2 sind auf Baumebene 2 zu finden. Der Suchbaum für die ersten 3 expandierten Knoten sieht folgendermaßen für die gewählte Problemrepräsentation aus (in kleinen Kästchen an den Knoten stehen die f-Werte, der nächste zu expandierende Knoten ist fett (rot) umrandet):

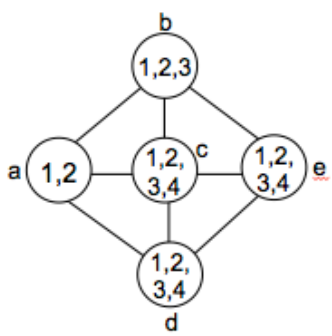


(Nach dem Suchbaum war nicht gefragt, es reicht aus, eine der beiden nachfolgenden Lösungen zu notieren. Natürlich sind andere Ergebnisse bei einer differierenden Suchproblemrepräsentation denkbar):
Auf Baumebene 2 existieren 2 Knoten mit minimalen Pfadkosten (50). Einer dieser Pfade wird als erstes expandiert, also

a) < ([1,2,3,4], {A,B,C,D}, { }), ([2,3,4], {A,C,D}, { (1,B) }) , ([3,4], {C,D}, {(1,B),(2,A)}) > (50) oder
b) < ([1,2,3,4], {A,B,C,D}, { }), ([2,3,4], {A,B,D}, { (1,C) }) , ([3,4], {B,D}, {(1,C),(2,A)}) > (50)

Aufgabe 2 – Constraints

CSP mit 5 Variablen {a,b,c,d,e}, die über 8 Ungleichheitsconstraints (≠) miteinander verbunden sind. Die Wertebereiche stehen in den Knoten.



a) MRV ist a (nur 2 Werte in der Domain). Nach a=1 wird der Wert 1 aus den Variablen {b,c,d} entfernt.

b) MCV ist c mit einem Vernetzungsgrad von 4. Nach c=1 ergeben sich folgende Wertebereichseinschränkungen durch den AC-3-Algorithmus:
   1. a={2}, b={2,3}, d={2,3,4}, e={2,3,4}
   2. b={3}, d={3,4}

3. e={2,4}
   Das 2-konsistente CSP ist also definiert durch: a={2}, b={3}, d={3,4}, e={2,4}.

Aufgabe 3 – Maschinelles Beweisen

Die folgende Wissensbasis beschreibt mit 4 Formeln eine einfache Blockswelt. O/2 und A/2 sind Prädikate (sie repräsentieren die on/2 bzw. above/2 Relation); {1,2,T} sind Konstanten (für zwei Blöcke bzw. den Tisch); {u,v,w,x,y,z} sind allquantifizierte Variablen. Sie sollen nun 3 Beweise für A(2, T) führen.

(1) $\neg O(u, v) \vee A(u, v)$
(2) $\neg A(x, y) \vee \neg A(y, z) \vee A(x, z)$
(3) $O(2, 1)$
(4) $O(1, T)$

a) Führen Sie einen Vorwärtsverkettungsbeweis.
b) Führen Sie einen Rückwärtsverkettungsbeweis.
c) Führen Sie einen Resolutionsbeweis.

Hornregeln:
(1) $O(u, v) \Rightarrow A(u, v)$
(2) $A(x, y) \wedge A(y, z) \Rightarrow A(x, z)$
(3) $O(2, 1)$
(4) $O(1, T)$

Vorwärtsverkettung:

(5)	A(2,1)	(1), (3) mit { u/2, v/1 }
(6)	A(1, T)	(1), (4) mit { u/1, v/T }
(7)	A(2, T)	(2), (5), (6) mit { x/2, y/1, z/T }

Rückwärtsverkettung:
Jede Rekursionstiefe ist jeweils um 1 Tabulator eingerückt. Zu Beginn einer Zeile steht die Nummer der angewendeten Regel, mit der das linke Subziel der letzten Iteration entfernt wird. Dahinter die Substitution. Als letztes die Menge der noch zu beweisenden Subziele - verwendet wird immer das links stehende für die nächste Rekursion.
Das Ziel in 1. wird zunächst durch Regel (1) durch ein Subziel ersetzt, und anschließend nach 2. wird Backtracking ausgelöst, weil für das Subziel keine Hornregel existiert. In 3. sorgt Regel (2) zunächst dafür, dass 2 Subziele entstehen; 5. und 7. sind verkürzende Regeln (Fakten).

1. (-) A(2, T)
2. (1) { u/2, v/T } O(2, T) fail
3. (2) { x/2, z/T } A(2,y), A(y,T)
4. (1) { u/2, v/y } O(2,y), A(y,T)
5. (3) { y/1 } A(1,T)
6. (1) { u/1, v/T } O(1,T)
7. (4) {}

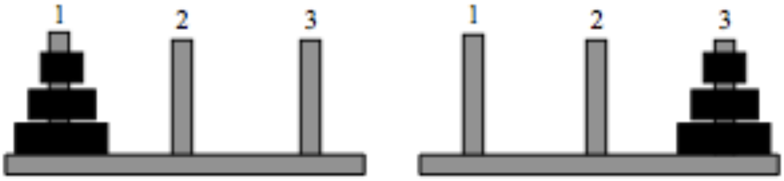
Resolution:

(5) $\neg A(2,T)$	
(6) $\neg O(2,T)$	(1),(5) { u/2, v/T) }
(7) $\neg A(2,y), \neg A(y,T)$	(2),(5) { x/2, z/T) }
(8) $\neg O(2,y), \neg A(y,T)$	(7),(1) { u/2 }

(9) ¬A(1,T)	(8),(3) { y/1 }
(10) ¬O(1,T)	(9),(1)
(11) {}	(10),(4)

Aufgabe 4 – Türme von Hanoi - Planung

Drei Scheiben unterschiedlicher Größe liegen sortiert auf einem Stift. Ziel ist es, den Stapel von Stift 1 nach Stift 3 zu bewegen. Es stehen insgesamt 3 Stifte zur Verfügung. Es kann jeweils die oberste Scheibe eines Stiftes auf einen anderen Stift gelegt werden, vorausgesetzt sie wird auf einen leeren Stift oder eine größere Scheibe gelegt.



Startzustand Zielzustand

a) Definieren Sie geeignete Konstanten und Prädikate.

pit/1, disk/2, smaller/2, free/1, empty/1, on/2, at/2

T = Table (bottom)

ohne Typprädikate und ohne smaller ist die Modellierung nicht für andere Problemgrößen anpassbar.

b) Beschreiben Sie den Start- und Zielzustand.

S_0 = {on(A,B), on(B,C), on(C,T), at(A,1), at(B,2), at(C,2), free(A), pit(1), pit(2), disk(A), disk(B), disk(C), pit(3), empty(2), empty(3), smaller(1,2), smaller(2,3), smaller(1,3)}

S_Z = {at(A,3),at(B,3),at(C,3)}

c) Definieren Sie ein Aktionsschema move(s, x, y) für das Bewegen einer Scheibe s, die auf dem Stab x auf einer anderen Scheibe liegt, auf einen nicht leeren Stab y.

ACT	move(x, y, z) // disk x von pit y nach z; x liegt auf z
PRE	disk(x), free(x), at(x,y), on(x,w), y≠z, at(a,y), free(a), smaller(x,a)
ADD	free(y), at(x,z), on(x,a)
DEL	at(x,y), free(a)

d) relevant und konsistente Aktionen bzgl. move/3, die in einen "inkonsistenten" Vorgängerzustand führt:

move(C,1,3) --> ({x/C,y/1,z/3}) erfüllt Teilziel at(C,3). Generiert SZ-1={at(A,3),at(B,3),disk(C),free(C),at(C,1),on(C,w),at(C,1),free(C),smaller(C,a)}

Begründung: smaller(C,a) ist nicht beweisbar. Irgendwann wird Rückwärtsplanung dieses Teilziel untersuchen und Backtracking auslösen.