Technische Universität Berlin
Fakultät IV - Elektrotechnik und Informatik

Künstliche Intelligenz: Grundlagen und Anwendungen

WS 2014/2015

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

Schriftlicher Test - Teilklausur 1

13.12.2014

Aufgabe 1 – Suchproblem

Eine Schwimmstaffel besteht aus 4 verschiedenen Schwimmern, jeweils einer für jede Disziplin. Die Tabelle zeigt die Bestzeiten von 5 Kandidaten für die einzelnen Lagen. Ihre Aufgabe ist es, aus diesen Kandidaten die schnellste 4x50m Lagen-Schwimmstaffel zusammen zu stellen.

Kandidat / Disziplin	A	B	C	D	E
Rücken	32,5	27	28,5	32	30,5
Brust	38	27,5	37	29,5	36,5
Schmetterling	27,5	23	33,5	24,5	28
Kraul	23,5	21	24	23	25

a) Repräsentieren Sie dieses Problem als Suchproblem.

Auf die 4 freien Plätze der Staffel jeweils einen Schwimmer zuordnen. Die beste Lösung wird über die Aktionskosten gefunden. Zustand enthält die noch unbesetzten Lagen in einer Liste (um den Verzweigungsgrad klein zu halten), die noch nicht berücksichtigten Schwimmer in einer Menge und die Zuordnungen in einer Menge (anfangs leer). Im Zielzustand ist die erste Liste leer und die Lösung kann aus dem 3. Tupelelement abgelesen werden.

S0 = < [R,B,S,K], {A,B,C,D,E}, {} >
SZ = < [], x, y >
Aktion: a(l, s, Si) = Sj mit
Si = < [l|Rest], Schwimmer, Staffel >
l ist das erste Element der Liste l, Rest der Rest (ggfs. auch leere Liste)
s ∈ Schwimmer
Sj = < Rest, s \ {s}, y ∪ {<l,s>}
Aktionskosten: entsprechend Tabelle: Eintrag in Zeile l und Spalte s.

b) Charakterisieren Sie den (zyklenfreien) Suchbaum.

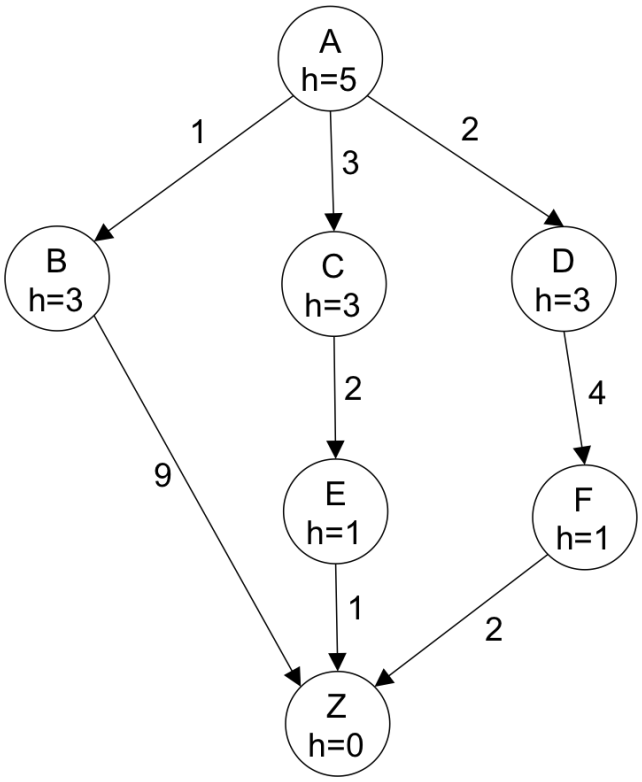
Verzweigungsgrad ist 5 (5 Schwimmer zur Wahl in S0, später jeweils einer weniger)
Tiefe ist 4 (mit jeder Aktion wird eine Lage besetzt, nach 4 Lagen ist ein Blatt erreicht
Bei einer Repräsentation der Lagen in a) als Menge stünden 20 Aktionen zur Wahl

c) Welchen Suchbaumalgorithmus würden Sie einsetzen? Begründen Sie Ihre Entscheidung.

Branch & Bound, denn es handelt sich um nicht-uniforme Aktionskosten, und die kostenminimale Lösung (schnellste Staffel) ist gesucht. A* mit h=0 ginge auch.

Aufgabe 2 – Suchbaumverfahren

Betrachten Sie folgenden gerichteten Graphen. Gesucht ist ein Weg von A nach Z. An den Kanten stehen die Aktionskosten, in den Knoten sind die heuristischen Werte notiert. Tie-Breaks werden aufsteigend in lexikographischer Reihenfolge aufgelöst (A vor B vor C...).



a) Welche Pfade liegen auf dem Stack, nachdem Tiefensuche eine Lösung gefunden hat?

Entwicklung des Tiefensuche-Stacks:
<A>; <AB,AC,AD>; <ABZ,AC,AD>.
*Nach dem Finden der Lösung ABZ liegen die beiden Pfade **AC und AD** auf dem Stack.*

b) Welche Pfade hat A* bereits untersucht (expandiert), wenn der Pfad mit E ganz vorn in der sortierten Liste liegt?

Sortierte Liste der Suche bis zur erwähnten Situation:

<A(5)>, <AB(4),AD(5),AC(6)>, <AD(5),AC(6),ABZ(10)>, <AC(6), ADF(7),ABZ(10)>, <ACE(6), ADF(7),ABZ(10)>
Untersucht (d.h. expandiert) wurden die 4 Pfade ***<A(5)>, <AB(4)>, <AD(5)>, <AC(6)>***

c) Welche Pfade liegen in der sortierten Liste, nachdem A* eine Lösung gefunden hat? Geben Sie auch die f-Werte an.

A* weitersuchen: <ACEZ(6), ADF(7),~~ABZ(10)~~>. ABZ aufgrund DP entfernt.
Nach Terminierung liegt noch der Pfad ***ADF(7)*** *in der sortierten Liste*
kein Fehler, wenn ABZ(10) erwähnt wird.

Aufgabe 3 – Constraints

Konstruieren Sie *ein* unlösbares CSP mit nicht leeren Domains und folgenden Eigenschaften:
(1) Das konsistente Belegen einer Variable in Verbindung mit Forward Checking ist möglich,
(2) das Belegen der "Most Constrained Variable" in Verbindung mit Forward Checking führt jedoch zum sofortigen Scheitern.

a) Notieren Sie hier das CSP. Sie dürfen es formal definieren oder den Constraintgraphen zeichnen. In jedem Fall müssen Variablen, Constraints und Domains ersichtlich sein.

Das CSP benötigt mindestens 3 Variablen, beispielsweise
Variablen: a: {1}, c: {2}; b: {1,2}
Constraints: [1] a≠b, [2] b≠c
Das Belegen von a führt zum Entfernen des Werts 1 aus der Domain von b.
MCV ist b. Das Belegen von b führt zum Scheitern von (1) oder von (2), je nachdem, welcher Wert verwendet wird.

b) Ist Ihr CSP 2-konsistent? Begründen Sie kurz.

Das CSP ist nicht 2-konsistent, denn für b=1 existiert kein Wert in der Domain von a, der Constraint [1] erfüllt.

c) Beschreiben Sie die Effekte der Forward Checking Propagierung beim Belegen einer Variable in Fall (1) und in Fall (2).

(1) a=1 und Forward Checking bewirkt durch Auswertung von [1] zu b={2}. Weitere Constraints mit a existieren nicht, also stoppt die Propagierung ohne die Unlösbarkeit zu erkennen.

(2) b ist MCV. b=1 führt zum Scheitern von [1]; b=2 führt zum Scheitern von [2].

Aufgabe 4 – Maschinelles Beweisen

In einer Welt stapelbarer Blöcke und füllbarer Kisten mit
{ A, B, K, T } : Konstanten (für zwei Blöcke, eine Kiste und den Tisch);
{ x, y, z } : Variablen
definiert (4) einen Zustand, in dem *A* und *K* auf dem Tisch liegen und *B* in *K* liegt; und definieren die Regeln (1) bis (3) die transitive "über"-Relation.

(1) ∀x,y auf(x, y) ⇒ über(x, y)
(2) ∀x,y,z in(x, y) ∧ über(y, z) ⇒ über(x, z)
(3) ∀x,y,z über(x, y) ∧ über(y, z) ⇒ über(x, z)
(4) auf(A, T) ∧ auf(K, T) ∧ in(B, K)

a) Beweisen Sie über(B, T) mittels Resolution. (17 Punkte)

KNF (a für auf; ü für über, i für in):
(1) *¬a(x,y) ∨ ü(x,y)*
(2) *¬i(x,y) ∨ ¬ü(y,z) ∨ ü(x,z)*
(3) *¬ü(x,y) ∨ ¬ü(y,z) ∨ ü(x,z)*
(4) *a(A,T)*
(5) *a(K,T)*
(6) *i(B,K)*

(7) *¬ü(B,T)* *// negierte Anfrage in KNF*

nun Resolventen bilden mit Stützmengenstrategie:

(8) *¬i(B,a) ∨ ¬ü(a,T)* *// 7.&2. {x/B, z/T, y/a} (frische Variable a)*
(9) *¬ü(K,T)* *// 8.&6. {a/K}*
(10) *¬a(K,T)* *// 9.&1. {x/K, y/T}*
(11) *{}* *// 10.&5.*

Längere Beweiswege führen über die (nicht unbedingt notwendigen) Klauseln (3) und (4)

Mit Forward Chaining (FC) oder Backward Chaining (BC) ließe sich der Beweis von über(B, T) ebenfalls durchführen. Der Aufwand des Beweises lässt sich messen durch:
- die Zahl der generierten Fakten beim FC,
- die Zahl der generierten Subziele (der erzeugten Knoten im Beweisbaum) beim BC.

b) Finden Sie eine definite Hornregel, die den FC-Beweis von *über(B, T)* aufwändiger macht, aber keine negative Auswirkung auf BC hat. Begründen Sie kurz, warum der FC-Aufwand steigt, der BC-Aufwand hingegen gleich groß bleibt.

Gesucht ist eine Hornregel, die die Menge der Fakten bei FC erhöht, die aber beim BC nicht zur Anwendung kommt:
∀x,y auf(x, y) ⇒ groß(x). Diese Regel generiert für jedes auf(x,_) ein groß(x)-Fakt. BC hingegen würde diese Regel nicht anwenden, weil groß(x) kein Subziel im Beweisbaum darstellt.
Falsch wäre eine Hornregel der Art { ¬auf(x, y), ¬in(u, v), in(x, v) }. Diese sorgt zwar für zusätzliches Wissen bei FC, würde aber auch im BC zur Anwendung kommen. Hierfür gibt es 2 Punkte Abzug.

c) Finden Sie eine definite Hornregel, die den BC-Beweis von *über(B, T)* aufwändiger macht, aber keine negative Auswirkung auf FC hat. Begründen Sie kurz, warum der BC-Aufwand steigt, der FC-Aufwand hingegen gleich groß bleibt.

Gesucht ist eine Hornregel, die bei BC zur Anwendung kommt, bei FC hingegen nicht:

∀x,y höher(x, y) ⇒ über(x, y). Diese Regel kommt - bei entsprechender hoher Priorisierung - in BC zur Anwendung und löst anschließend Backtracking aus, weil das Ziel höher/2 nicht erfüllbar ist. FC hingegen generiert kein Fakt höher/2, sodass diese Regel auch nie zur Anwendung kommen kann.

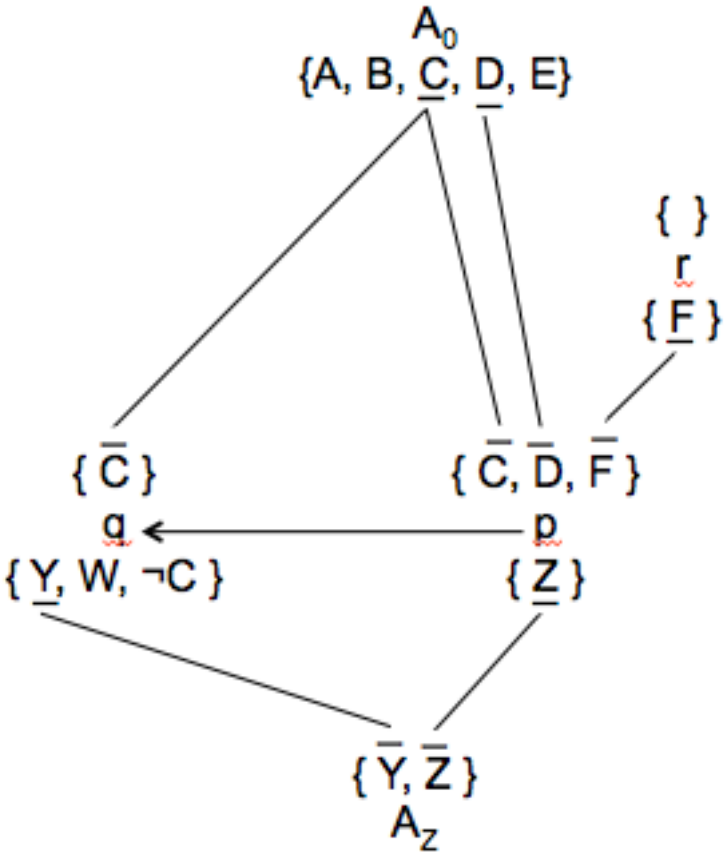
Aufgabe 5 – Partial-Order Planning

Betrachten Sie folgendes aussagenlogisches Planungsproblem ohne Variablen mit 4 Operatoren *o, p, q, r*, Startzustand S_0 und Ziel S_Z. Die PRE von *r* ist leer, ebenso die DEL von *p* und *r*:

ACT *o*	ACT *p*	ACT *q*	ACT *r*
PRE: A, B	PRE: C, D, F	PRE: C	PRE:
ADD: X	ADD: Z	ADD: Y, W	ADD: F
DEL: A	DEL:	DEL: C	DEL:

S_0 = {A, B, C, D, E} S_Z = {Y, Z}

a) Zeichnen Sie den Partial-Order Plan.



b) (6 Punkte)
Was geschieht, wenn der Operator *r* durch folgenden Operator *r'* ersetzt wird?

ACT r` PRE: C, D ADD: F DEL: C

Dieser „F"-Generator kann keine Aktion im Lösungsplan sein, denn C ist im DEL, C wird aber von der kausal abhängigen Aktion p benötigt. Eine Auflösung der Bedrohung ist nicht möglich. Mit r`anstelle von r existiert daher kein Lösungsplan.