

The Answers

The Answers

1

a)

[bookwork]

depth first: choose node on search frontier furthest from start

depth-limited depth first: stop depth first search if path depth $d = \text{limit}$ id depth first: depth limited depth limited first with $\text{limit}=0$, $\text{limit}=1$, $\text{limit}=2$ df: optimal no, complete no, time complexity b^m ($m = \text{max depth of tree}$), space complexity b^m dl-df: optimal no, complete no, time complexity b^l ($l = \text{limit}$), space complexity b^l id df: optimal yes, complete yes, time complexity b^l ($l = \text{limit}$), space complexity b^d

b)

[application]

```
select2( [H|T], (H,Other) ) :-
    select1( T, Other ).
```

```
select2( [_|T], Pair ) :-
    select2( T, Pair ).
```

```
select1( [H|T], H ).
```

```
select1( [_|T], H ) :-
    select1( T, H ).
```

c)

[application]

([(a,A), (b,B), (c,C), (d,D)], T) where

A, B, C, D are l or r to indicate which side of the river, T is the time elapsed

([(a,l), (b,l), (c,l), (d,l)], 0)

([(a,r), (b,r), (c,r), (d,r)], T), $T < 16$

```
state_change( 2lr, (List, Time), (Newlist, NewTime) ) :-
```

```
    select2( [a,b,c,d], Pair ),
    both_left( Pair, List ),
    change_banks( Pair, r, List, NewList ),
    slowest( Pair, Elapsed ),
    NewTime is Time + Elapsed.
```

```
state_change( 1lr, (List, Time), (Newlist, NewTime) ) :-
```

```
    select1( [a,b,c,d], One ),
    member( (One,l), List ),
    replace( One, r, List, NewList ),
    crossing_time( One, Elapsed ),
    NewTime is Time + Elapsed.
```

Similar rules for one or two people crossing right to left

```
both_left( (X,Y), List ) :-  
    member( (X,l), List ),  
    member( (Y,l), List ).
```

```
change_banks( (X,Y), Bank, List, NewList ) :-  
    replace( X, Bank, List, Temp ),  
    replace( Y, Bank, Temp, NewList ).
```

```
replace( X, B, [(X,_)|T], [(X,B)|T] ).  
replace( X, B, [H|T1], [H|T2] ) :-  
    replace( X, B, T1, T2 ).
```

```
slowest( (X,Y), Slower ) :-  
    crossing_time( X, Faster ),  
    crossing_time( Y, Slower ),  
    Faster < Slower, !.
```

```
slowest( (X,Y), Slower ) :-  
    crossing_time( X, Slower ).
```

[Solution

1,2, cross
1 comes back
5,8 cross
2 comes back
1,2 cross (again)

time elapsed = 2+1+8+2+2 = 15]

2

a)

[bookwork]

uniform cost: choose node on search frontier with least actual cost from start node (cost function g)

best first: choose node on search frontier with least estimated cost to goal node (heuristic function h)

A^* : choose node n on search frontier with least estimated path cost from start node to goal node through n ($f = g + h$)

Uni cost: optimal yes (assuming ...), complete yes, time complexity b^d ($m = \max$ depth of tree), space complexity b^d

Best f: optimal no, complete no, time complexity b^d , space complexity b^d (but this is worst case, do much better than this with a good heuristic)

A^* : optimal yes, complete yes, complexity depends on heuristic

b)

[understanding]

i)

$P_0 = \{ \langle \text{start}, 0 \rangle \}$

$P_{x+1} = \{ \langle p_i + 1, n_j \rangle \mid \exists (p_i, g) \in P_x . \langle \text{frontier}(p_i), e, n_j \rangle \in R \}$

ii)

$P'_0 = \{ \langle \text{start}, 0 \rangle \}$

$P'_{x+1} = \{ \langle p_i + 1, n_j \rangle \mid \exists \text{op} \in \text{Op} . \exists (p_i, g) \in P'_x . \langle \text{frontier}(p_i), e \rangle = \text{op}(n_j) \}$

iii)

$\forall n_1, e, n_2, \langle n_1, e, n_2 \rangle \in R \iff \exists \text{op} \in \text{Op} . \text{op}(n_1) = \langle n_2, e \rangle$

c)

[understanding]

$P'_G = \bigcup_{x=0}^{\infty} P'_x$

$P'_0 = \{ \langle s, 0 \rangle \}$

$P'_1 = \{ \langle s, n_1 \rangle, e_1, \dots \}$ for all n_i such that $\text{op}(s) = \langle n_i, e \rangle$

So uniform cost picks the only path in P'_0 , the cheapest e_i in P'_1 , the cheapest in the union of P'_1 and the subset of P'_2 generated, and so on

d)

[understanding]

admissible, monotonic

actual cost of goal node is f^*

node expanded if $g(n) + h(n) < f^*$

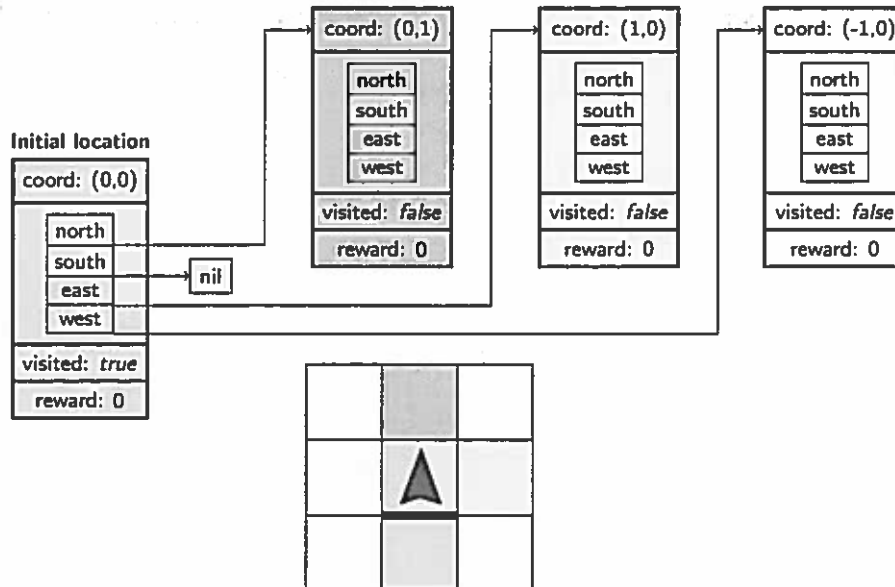
maximize $h(n)$ without over-estimating

3

a)

[Bookwork]

Search & Reward: Initialise



E3 16 Artificial Intelligence | L7 Path Learning & Planning

Path Finding

adopt a basic exploration strategy (random, follow left wall,...) search until exit (goal state) found – receive reward propagate reward backwards from exit state through states which lead to such exit state (credit assignment)

Path Following

from each state, move to next state with the highest reward – optimal move

Path Learning (by “reinforcement”)

repeat

path following, using a (sub)-strategy to visit unexplored part of the grid credit assignment on-line (exploration and change detection)

propagate rewards backwards

until: done enough training runs, good enough solution, whole maze explored, run out of time/money...

b)

[Understanding]

exploit until find change then re-search, updating map.

Rate of change of environment faster than replanning

Do not replan – might make plan and be out of date

Do replan – might be constantly interrupting to replan and make no progress

c)

[Understanding]

search to fixed ply rather than exhaustive

use depth first search

apply heuristic function rather than assign 1 or 0

use propagation and cut-off rules to prune search space

4

a)

[bookwork]

resolution: a rule of inference used for automated theorem proving

rule:

$p \quad \neg p$ / contradiction

$p + q \quad \neg p$ / q

$p_1 + p_2 + \dots + p_i + \dots + p_m \quad \neg p_i + q_1 + q_2 + \dots + q_n$ / $p_1 + p_2 + \dots + q_1 + q_2 + \dots + q_n$
 $+ \dots + p_m$

in logic programming: query and horn clauses

b)

[bookwork]

unification: solving the problem of equating symbolic expressions

algorithm

uninstantiated variable unifies with atom, term or other uninstantiated variable

atom unifies with atom

term unifies with term if functors same, arity same, pairwise arguments unify

in logic programming: way of binding values to variables in resolution

c)

$\neg \text{longnose}(X1) \vee \text{seebetter}(X1, Y1)$

$\neg \text{bigeyes}(X2) \vee \text{smellbetter}(X2, Y2)$

$\neg \text{seebetter}(X3, Y3) \vee \neg \text{smellbetter}(X3, Y3) \vee \text{chase}(X3, Y3)$

$\neg \text{chase}(X4, Y4) \vee \neg \text{faster}(X4, Y4) \vee \text{catch}(X4, Y4)$

$\neg \text{wolf}(X5) \vee \neg \text{littlegirl}(Y5) \vee \neg \text{catch}(X5, Y5) \vee \text{eat}(X5, Y5)$

d)

$\text{wolf}(\text{bbw})$

$\text{littlegirl}(\text{rrh})$

$\text{bigeyes}(\text{bbw})$

$\text{longnose}(\text{bbw})$

$\text{faster}(\text{bbw}, \text{rrh}).$

$\neg \text{eat}(\text{bbw}, \text{rrh})$

$\neg \text{wolf}(X5, Y5) \vee \neg \text{littlegirl}(X5, Y5) \vee \neg \text{catch}(X5, Y5) \quad \{X5 \rightarrow \text{bbw}, Y5 \rightarrow \text{rrh}\}$

$\neg \text{littlegirl}(X5, Y5) \vee \neg \text{catch}(X5, Y5)$

$\neg \text{catch}(X5, Y5)$

$\neg \text{chase}(X4, Y4) \vee \neg \text{faster}(X4, Y4) \quad \{X4 \rightarrow X5, Y4 \rightarrow Y5\}$

$\neg \text{seebetter}(X3, Y3) \vee \neg \text{smellbetter}(X3, Y3) \vee \neg \text{faster}(X4, Y4) \quad \{X3 \rightarrow X4, Y3 \rightarrow Y4\}$

$\neg \text{longnose}(X1) \vee \neg \text{smellbetter}(X3, Y3) \vee \neg \text{faster}(X4, Y4) \quad \{X1 \rightarrow X3, Y1 \rightarrow Y3\}$

$\neg \text{smellbetter}(X3, Y3) \vee \neg \text{faster}(X4, Y4)$

-bigeyes(X2) \vee -faster(X4,Y4)
>Y3}

{X2 \rightarrow X3, Y2-

-faster(X4,Y4)

contradiction

5)

a)

[bookwork]

$p \ \& \ q \ / \ p, \ q$

$\neg(p \ \& \ q), \ p \ / \ \neg q$

$\neg(p \ + \ q) \ / \ \neg p, \ \neg q$

$p \ + \ q, \ \neg p \ / \ q$

$\neg(p \ \rightarrow \ q) \ / \ p, \ \neg q$

$p \ \rightarrow \ q, \ p \ / \ q$

$p \ \rightarrow \ q, \ \neg q \ / \ \neg p$

$p \leftrightarrow q, \ p \ / \ q$ etc.

PB bivalence branch $p \mid \neg p$ every formula is either true or false

closure $p, \neg p$ / close trying to make every formula true on a branch, can't make p and $\neg p$ true on the same branch

if you have one component of a beta formula on a branch, it can safely be analysed as there is no use in applying pb as it will not add any information, ie

suppose

$p \ + \ q$

p

try branching on subformula p :

branch 1 $\neg p$... close immediately

branch 2 p ... we already knew that

try branching on sub-formula q :

branch 1 $\neg q$

add p ... we already knew that but now add

branch 2 q

we have to prove $\neg p$ on two branches

we can close on literals if we can close on sub-formulas, but any complex formula, trying to F and $\neg F$ true then we must have at least one literal different in the row of the truth table

if it is sound to close on contradictory sub-formulas it is sound to close on complementary literals

As we are guaranteed to end up with literals the algorithm is complete

b)

[Application]

1	premise	$(\neg A \ \& \ \neg B) \rightarrow (\text{out} \leftrightarrow \neg C)$
2	premise	$(\neg A \ \& \ B) \rightarrow (\text{out} \leftrightarrow \neg C)$
3	premise	$(A \ \& \ \neg B) \rightarrow \neg \text{out}$
4	premise	$(A \ \& \ B) \rightarrow \text{out}$
5	neg conc	$\neg (\text{out} \leftrightarrow ((A \ \& \ B) + (\neg A \ \& \ \neg C)))$

Branch 1

6	PB1	out
7	\leftrightarrow , 5, 6	$\neg ((A \ \& \ B) + (\neg A \ \& \ \neg C))$
8	a, 7	$\neg (A \ \& \ B)$
9	a, 7	$\neg (\neg A \ \& \ \neg C)$
10	b, 3, 6	$\neg (A \ \& \ \neg B)$

Branch 1.1

11	PB1	$(\neg A \ \& \ \neg B)$
12	a, 11	$\neg A$
13	a, 11	$\neg B$
14	b, 1, 11	$\text{out} \leftrightarrow \neg C$
15	b, 6, 14	$\neg C$
16	b, 9, 12	$\neg \neg C$
17	-- 16	C

Close (15,17)

Branch 1.2

18	PB2	$\neg (\neg A \ \& \ \neg B)$
----	-----	-------------------------------

Branch 1.2.1

19	PB1	$\neg A \ \& \ B$
20	a, 19	$\neg A$
21	a, 19	B
22	b, 2, 19	$\text{out} \leftrightarrow \neg C$
23	b, 6, 14	$\neg C$
24	b, 9, 20	$\neg \neg C$
25	--, 24	C

Close (23,25)

Branch 1.2.2

26	PB2	$\neg (\neg A \ \& \ B)$
----	-----	--------------------------

[we now have $\neg (A \ \& \ B)$, $\neg (\neg A \ \& \ B)$, $\neg (\neg A \ \& \ \neg B)$ and $\neg (A \ \& \ \neg B)$ on the branch...]

Branch 2

27	PB2	$\neg \text{out}$
28	\leftrightarrow , 5, 27	$(A \ \& \ B) + (\neg A \ \& \ \neg C)$
29	b, 4, 27	$\neg (A \ \& \ B)$
30	b, 28, 29	$\neg A \ \& \ \neg C$
31	a, 30	$\neg A$
32	a, 31	$\neg C$

		Branch 2.1
33	PB1	(-A & B)
34	a, 33	-A
35	a, 33	B
36	b, 2, 33	out <-> -C
37	b, 27, 36	--C
38	--, 37	C
		Close (32,38)

		Branch 2.2
39	PB2	-(-A & B)
40	b, 39, 31	-B

		Branch 2.2.1
41	PB1	-A & -B
42	b, 1, 42	out <-> -C
43	b, 27, 42	--C
44	--, 43	C
		Close (32,44)

		Branch 2.2.2
46	PB2	-(-A & -B)
47	b, 31, 46	--B
48	--, 47	B
		Close (40,48)

6

a)

[bookwork]

$$wff = \Box wff \mid \Diamond wff$$

$$M = \langle W, R, \Vdash \rangle$$

Where W is a non-empty set of worlds

R is binary accessibility relation on W

\Vdash denotation function $p \rightarrow 2^W$ maps each proposition to subset of W where it is true

$$\Vdash M, a \Box p \leftrightarrow \text{forall } b . aRb \rightarrow \Vdash M, b p$$

$$\Vdash M, a \Diamond p \leftrightarrow \text{exists } b . aRb \wedge \Vdash M, b p$$

b)

[bookwork, understanding]

$\Box p$ is true because there is no b such that aRb thus the antecedent aRb is false so does not matter that $\Vdash M, b p$ is false, false implies anything is true, so by \leftrightarrow this means that $\Vdash M, a \Box p$ is true.

$\Diamond p$ is false because there is no b such that aRb thus the left-hand conjunct is false so false and anything is false, so by \leftrightarrow this means $\Vdash M, a \Diamond p$ is false

c)

[understanding – but for the first part, just take the model given in part b]

Model $M = \langle \{a\}, \{\}, \Vdash p \Vdash = \{a\} \rangle$

Then

$$\Vdash M, a p \quad \text{is true}$$

$$\Vdash M, a \Box p \quad \text{is true} \quad \text{by semantics of box}$$

$$\Vdash M, a \Diamond p \quad \text{is true} \quad \text{by implication of axiom}$$

but

$$\Vdash M, a \Diamond p \quad \text{is false} \quad \text{by semantics of dia}$$

contradiction

assume $\Box p$

we want to show $\Diamond p$

$$\rightarrow \text{forall } b . aRb \rightarrow \Vdash M, b p \quad \text{by semantics of box}$$

$$\text{but } \text{forall } a . \text{exists } b . aRb \quad \text{by seriality}$$

$$\rightarrow \Vdash M, b p \quad \text{by forall}$$

$$\rightarrow aRb \text{ and } \Vdash M, b p$$

$$\rightarrow \text{exists } b . aRb \wedge \Vdash M, b p$$

$$\leftrightarrow \Diamond p$$

as required

d)
 [application]
 in K
 1: $\neg(\text{box } p \rightarrow \text{dia } p)$
 1: $\text{box } p$
 1: $\neg \text{dia } p$
 ???

in S5
 1: $\neg(\text{box } p \rightarrow \text{dia } p)$
 1: $\text{box } p$
 1: $\neg \text{dia } p$
 1: p
 1: $\neg p$
 close

In K, we cannot be sure there is a world to go to make either $\text{box } p$ or $\neg \text{dia } p$ true.

e)
 [application]
 1: $\neg(p \leftrightarrow q) \rightarrow (\Box p \leftrightarrow \Box q)$
 1: $p \leftrightarrow q$
 1: $\neg(\Box p \leftrightarrow \Box q)$

Branch 1
 1: $\Box p$
 1: $\neg \Box q$
 2: $\neg q$
 2: p

Model $M = \langle \{1,2\}, \{ (1,2) \}, \Vdash p \Vdash = \{1,2\} \Vdash q \Vdash = \{1\} \rangle$