Inteligência Artificial – Folha de Respostas

1/Fev/2011 – 9h **DI/FCT/UNL, Época Recurso, 2010/11, Duração: 3h** **Exame:** abdcabdc

Nome:	**Número:**

GRUPO I

I.1 Para a heurística ser admissível não poderá sobrestimar o custo real. Assim, para cada estado o valor da heurística deverá ser exatamente igual ao custo real (valor máximo possível para a heurística).

h(x) = h(y) = 0.0 por serem estados objetivo
h(w) = 1
h(v) = 2
h(u) = 5

I.2 Como a heurística é sempre nula, o algoritmo de procura A* corresponde na realidade ao algoritmo de procura de custo uniforme pois o valor da função de avaliação para cada nó f(n) = g(n) + h(n) = g(n) em que g(n) é o custo acumulado, i.e. o custo do nó inicial até n. O nó selecionado encontra-se sublinhado.

A(0)
B(4) D(5)
A(4) D(5) C(6) E(7)
D(5) C(6) E(7) B(8) D(9)

O algoritmo termina devolvendo a solução A->D com custo 5, que é a solução de menor custo. Tal seria esperado pois o algoritmo de custo uniforme garante o óptimo.

I.3

$P(i_1) = 5/50 = 0.1 = 10\%$
$P(i_2) = 5/50 = 0.1 = 10\%$
$P(i_3) = 25/50 = 0.5 = 50\%$
$P(i_4) = 15/50 = 0.3 = 30\%$

I.4

Considere-se a variável Z:

1) O valor 0 em Z é inconsistente pois não existe valor $X < 0$
2) O valor 1 em Z é inconsistente pois não existe valor $X < 1$
3) O valor 2 em Z é inconsistente pois não existe valor Y tal que $2 < Y$
4) O valor 3 em Z é inconsistente pois não existe valor Y tal que $3 < Y$

Como foram removidos todos os valores do domínio de Z, logo não existe solução para o problema.

I.5 P(c | a, d, e, ~ f) = P(c | a, ~b, d, e,~f) pois A,D,E,F é o o Markov Blanket da variável C. Sabe-se ainda que qualquer nó é condicionalmente independente de todos os outros dado o seu Markov Blanket.

Inteligência Artificial – Folha de Respostas

11/Jan/2011 – 9h **DI/FCT/UNL, Época Normal, 2010/11, Duração: 3h** **Exame:** dabadca

Nome:	**Número:**

I.6

contraex :- ponto(X), água(X), observação(X,O,pos),obs(O), not interessante(X).

universal :- not contraex.

:- not universal.

OU

contraex :- ponto(X), água(X), observação(X,O,pos),obs(O), not interessante(X).

:- contraex.

I.7 Os modelos possíveis são M_1={a,d} e M_2={b,c}.

Vamos verificar que $M_1 = \Gamma(M_1) = least\left(\frac{Q}{M_1}\right)$

$$\frac{Q}{M_1} = \left\{ \begin{array}{c} a. \\ d. \\ p :- a, c. \\ p :- b, d. \\ r :- p. \end{array} \right\}$$

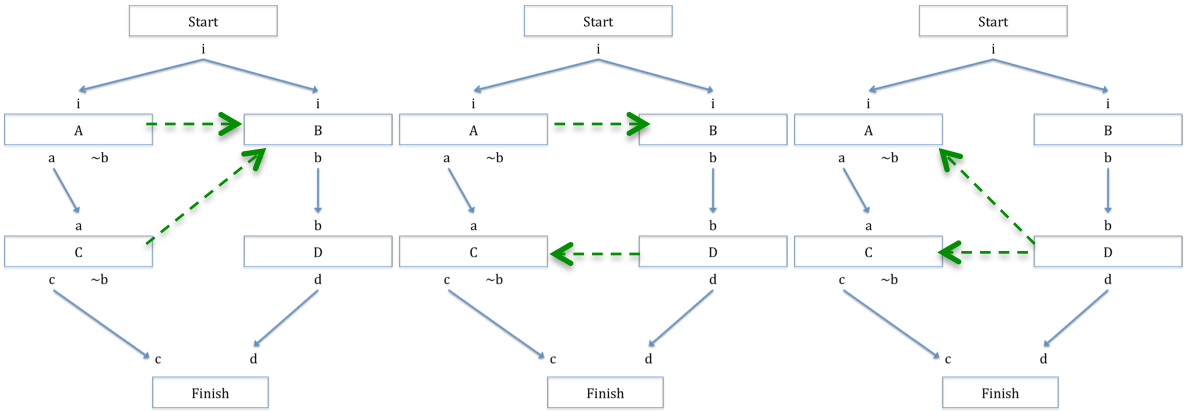
Logo, $least\left(\frac{Q}{M_1}\right) = \{a, d\} = M_1$ como se queria demonstrar. Logo M_1={a,d} é modelo estável.

I.8 Os modelos da teoria **T** são quatro:

b	**p**	**s**	**c**	**g**
true	false	false	true	true
false	true	false	true	true
false	false	true	false	true
false	false	true	true	true

Como em todos eles se tem g=true, logo g é consequência lógica desta teoria.

I.9 Os passos A e C ameaçam ambos a ligação causal que vai de B para D.



Cada um dos planos POP só tem uma linearização possível, respetivamente, ACBD, ABDC e BDAC

I.10 O primeiro neurónio dispara quando 0.2X+0.8Y-0.4>= 0 ou seja, 0.2X+0.8Y>=0.4. O segundo neurónio dispara quando N1+Z >= 0.5 (com N1 a saída do primeiro neurónio), ou seja efetua a disjunção (OU) das suas entradas. Assim é fácil construir a tabela de verdade:

X	Y	Z	Out
0	0	0	0
0	1	0	1
1	0	0	0
1	1	0	1

X	Y	Z	Out
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

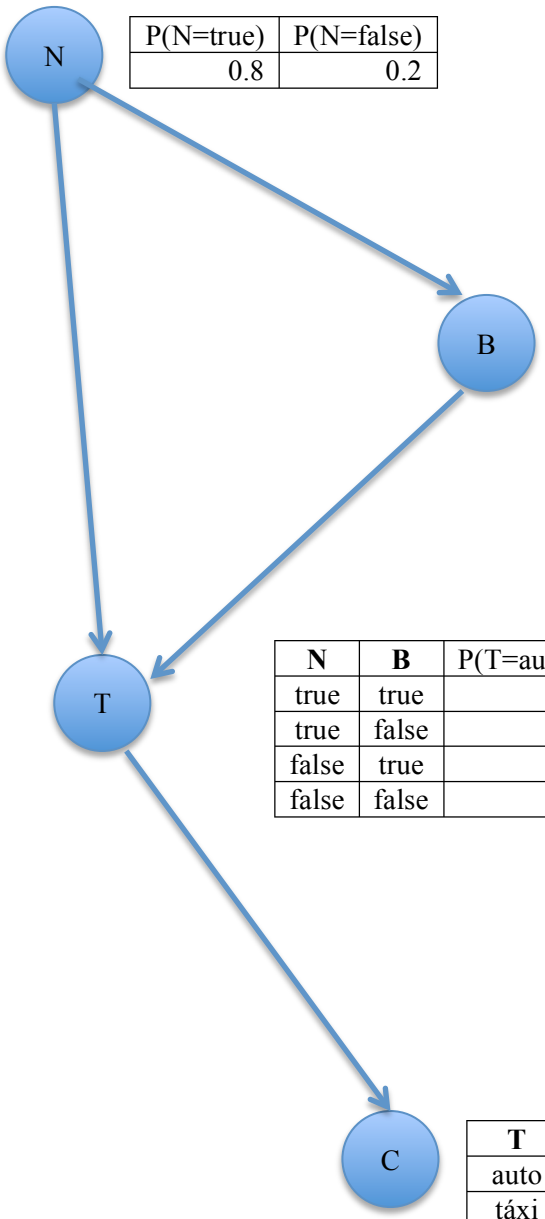
Inteligência Artificial – Folha de Respostas

1/Fev/2011 – 9h **DI/FCT/UNL, Época Recurso, 2010/11, Duração: 3h** **Exame:** abdcabdc

Nome:	**Número:**

GRUPO II

II.1 Variáveis aleatórias Nacional (N), Bagagem (B), Transporte (T), Confortável (C). As variáveis N, B e C são booleanas enquanto T tem domínio {auto,táxi,outro}. A topologia e as TPC da rede são as seguintes:



P(N=true)	P(N=false)
0.8	0.2

N	P(B=true|N)	P(B=false|N)
true	0.6	0.4
false	0.8	0.2

N	**B**	P(T=auto|N,B)	P(T=táxi|N,B)	P(T=outro|N,B)
true	true	0.1	0.6	0.3
true	false	0.1	0.8	0.1
false	true	0	0.9	0.1
false	false	0	0.9	0.1

T	P(C=true|T)	P(C=false |T)
auto	0.3	0.7
táxi	0.8	0.2
outro	0.9	0.1

Inteligência Artificial – Folha de Respostas

1/Fev/2011 – 9h **DI/FCT/UNL, Época Recurso, 2010/11, Duração: 3h** **Exame:** abdcabdc

Nome:	**Número:**

II.2

$P(T = táxi|C = true, B = false) = \alpha P(T = táxi, C = true, B = false)$

$P(T, C = true, B = false)$

$= \sum_{n \in \{true, false\}} P(T, C = true, B = false, N = n)$

$= \sum_{n \in \{true, false\}} P(N = n) \times P(B = false|N = n) \times P(T|B = false, N = n) \times P(C = true|T) =$

$= P(C = true|T) \times \sum_{n \in \{true, false\}} P(N = n) \times P(B = false|N = n) \times P(T|B = false, N = n)$

Para T=táxi

$P(T = táxi, C = true, B = false) = 0.8 \times (0.8 \times 0.4 \times 0.8 + 0.2 \times 0.2 \times 0.9) = 0.2336$

Para T=auto

$P(T = auto, C = true, B = false) = 0.3 \times (0.8 \times 0.4 \times 0.1 + 0.2 \times 0.2 \times 0) = 0.0096$

Para T=outro

$P(T = outro, C = true, B = false) = 0.9 \times (0.8 \times 0.4 \times 0.1 + 0.2 \times 0.2 \times 0.1) = 0.0324$

Logo,

$$P(T = táxi|C = true, B = false) = \frac{0.2336}{0.2336 + 0.0096 + 0.0324} = \frac{0.2336}{0.2756} = 0.847605 \cong 84.76\%$$

II.3

$$P(N = true, B = true, C = false) = \sum_{t \in \{auto, táxi, outro\}} P(N = true, B = true, C = false, T = t) =$$

$$= \sum_{t \in \{auto, táxi, outro\}} P(N = true) \times P(B = true | N = true) \times P(T = t | B = true, N = true) \times P(C = false | T = t)$$

$$= P(N = true) \times P(B = true | N = true) \times \sum_{t \in \{auto, táxi, outro\}} P(T = t | B = true, N = true) \times P(C = false | T = t)$$

$$= 0.8 \times 0.6 \times (0.1 \times 0.7 + 0.6 \times 0.2 + 0.3 \times 0.1) = 0.1056 = 10.56\%$$

Inteligência Artificial – Folha de Respostas

1/Fev/2011 – 9h **DI/FCT/UNL, Época Recurso, 2010/11, Duração: 3h** **Exame:** abdcabdc

Nome:	**Número:**

GRUPO III

III.1 Cada navio pode ser colocado em cada uma das n^2 casas de 2 maneiras possíveis. Logo existindo f navios existem $(2n^2)^f = (2)^f \times (n^2)^f$ estados de acordo com a simplificação. Por curiosidade, caso se assuma que os navios não podem ficar colocados uns por cima dos outros (mas ignorando a sua dimensão) o número de estados é dado por $2^f \times C_f^{n^2}$.

III.2 Um estado pode ser representado por um par *(G,D)* em que *G* é um array *(n+1)*(n+1)* de inteiros, em que a última linha e a última coluna mantém os totais ainda por satisfazer; *D* é um array de dimensão *n* com os navios ainda por colocar. Assuma-se que os arrays têm como menor índice 1.

Estado inicial: dada uma instância (n,TotLin,TotCol,Frota) o estado inicial é formado pelo par (G,Frota) em que G[i,n+1]=TotLin[i] (1 <= i <= n) e G[n+1,j]=TotCol[j] (1 <= i <= n) e, opcionalmente, G[n+1,n+1] = S. Estes valores indicam o que falta ainda preencher em linhas, em colunas e no total. As restantes entradas do array G encontram-se inicializadas com o valor 0.

Teste de estado objetivo: $\forall_{1 \le k \le n} G[k, n+1] = G[n+1, k] = 0$. Repare-se que não é suficiente ter-se G[n+1,n+1] = S = 0 (mas é necessário). Logo, pode-se testar primeiro se S=0 se não for, devolve-se falso. Só quando S=0 é que é necessário ir confirmar no array G que todos os totais estão a 0.

Operadores: colocarHoriz(C,Y,X) e colocarVert(C,Y,L), em que C é o comprimento do navio e (Y,X) a coordenada (Linha,Coluna) onde foi colocado.

Função sucessor:

(colocarHoriz(C,Y,X),(G',D')) ∈ S((G,D))

sse D[C] > 0 (existe um barco disponível)

e o retângulo de casas da grelha G de extremos (Y-1,X-1) a (Y+1,X+C) está preenchido com 0s (é preciso cuidado extra para quando os extremos caem fora dos limites)

e G[Y,n+1] >= C (há casas por pintar suficiente nessa linha)

e G[n+1,X] > 0 e ... e G[n+1,X+C-1] > 0 (há casas suficiente por pintar nas colunas)

e D'[C] = D[C] – 1 (atualização dos navios disponíveis)

e G'[Y,n+1] = G[Y,n+1] – C (atualização totais em linhas)

e G'[n+1,X] = G[n+1,X]-1, ..., G'[n+1,X+C-1] = G[n+1,X+C-1]-1 (atualização totais em colunas)

e G'[Y,X] = 1, ..., G'[Y,X+C-1] = 1 (preenchimento do navio na grelha)

e G'[n+1,n+1] = G[n+1,n+1]-C (atualização do total de casas por preencher).

e as restantes entradas de G' = G.

As condições para (colocarVert(C,Y,X),(G',D')) ∈ S((G,D)) são idênticas às anteriores, mas trocando linhas por colunas.

Custo do operador: 1

III.3 Uma heurística admissível nunca sobrestima o custo real.

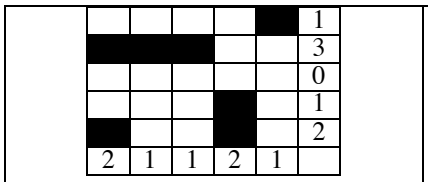
a) Não é admissível. Poderá não ser necessário colocar todos os navios para encontrar solução. Veja-se a solução para a instância dada em que foram colocados apenas 4 dos 5 navios. Logo neste caso a heurística devolveria erradamente o valor 1 quando o custo é 0 (estado objetivo).
b) É admissível. O valor S – C indica o número total de casas por preencher (a quantidade que se mantém em G[n+1,n+1] na representação do problema da alínea **III.2**). Ao dividirmos S – C pelo comprimento do maior navio por colocar estamos a assumir que preenchemos todas essas casas com o menor número de navios possível. Logo nunca irá sobrestimar o custo real.
c) Também é admissível. M indica o maior comprimento de barcos que se pode colocar (note-se que M pode ser maior, igual ou menor do que o maior barco disponível). Logo pelos motivos indicados anteriormente, esta heurística não sobrestima o custo real. Repare-se que quando M é maior do que o maior barco disponível esta heurística vai ser inferior à anterior.
d) Como b) e c) são admissíveis o máximo das duas também o é. Mas repare-se que nenhuma das heurísticas domina a outra. Porquê?

III.4 Sim, o algoritmo é completo. Corresponde basicamente a um problema de satisfação de restrições. O algoritmo descrito corresponde a uma procura com retrocesso limitada em profundidade atribuindo-se em cada passo um navio a uma posição (no máximo a profundidade da árvore será *f* correspondendo ao número de navios a colocar). Há vantagem em colocar-se primeiros os navios maiores pois permite falhar mais cedo reduzindo o tamanho da árvore de procura (ou alternativamente, estes navios têm menos possibilidades de ser colocados logo devem ser tentados primeiro pela heurística das variáveis mais constrangidas).

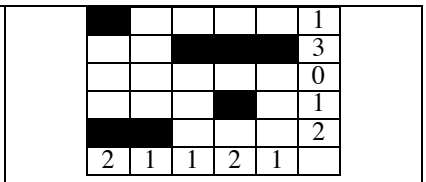
Este problema também pode ser resolvido com programação por conjunto de respostas cuja solução se detalha:

```
#const n=5.

linha(1..n).
coluna(1..n).
direcao(h;v).
pintadas(0..n).

navio(a;b;c;d;e).

comprimento(a,1).
comprimento(b,1).
comprimento(c,2).
comprimento(d,3).
comprimento(e,4).

totlinha(1,1).
totlinha(2,3).
totlinha(3,0).
totlinha(4,1).
totlinha(5,2).

totcoluna(1,2).
totcoluna(2,1).
totcoluna(3,1).
totcoluna(4,2).
totcoluna(5,1).

% Coloca um navio numa determinada casa segundo uma das duas direções
0 { posicao(N,D,L,C):direcao(D):linha(L):coluna(C)} 1 :- navio(N).

% Elimina modelos com o navio fora da grelha:
:- posicao(N,h,L,C), linha(L), coluna(C), comprimento(N,Comp), C + Comp > n+1.
:- posicao(N,v,L,C), linha(L), coluna(C), comprimento(N,Comp), L + Comp > n+1.

% “Pinta” as casas ocupadas pelos navios
ocupado(L,X,N) :- navio(N), linha(L), coluna(C), posicao(N,h,L,C),
                  comprimento(N,Comp), coluna(X), X >= C, X < C+Comp.

ocupado(Y,C,N) :- navio(N), linha(L), coluna(C), posicao(N,v,L,C),
                  comprimento(N,Comp), linha(Y), Y >= L, Y < L+Comp.

% Impede colisões ou adjacências entre navios distintos
:- linha(L;L1), coluna(C;C1), adjacente(L,C,L1,C1), navio(N1;N2), N1 != N2,
   ocupado(L,C,N1), ocupado(L1,C1,N2).

d(-1;0;1).
adjacente(L,C,L1,C1) :- linha(L;L1), coluna(C;C1), d(DX;DY), L1 = L + DY, C1 = C + DX.

% Conta o número de casas pintadas em linhas e colunas
somalinha(L,S) :- S {ocupado(L,C,N):coluna(C):navio(N)} S, linha(L), pintadas(S).
somacoluna(C,S) :- S {ocupado(L,C,N):linha(L):navio(N)} S, coluna(C), pintadas(S).

% Garante que os modelos têm o número certo de casas pintadas em linhas e colunas
:- linha(L), not somalinhacorreta(L).
:- coluna(C), not somacolunacorreta(C).

somalinhacorreta(L) :- somalinha(L,T), totlinha(L,T).
somacolunacorreta(C) :- somacoluna(C,T), totcoluna(C,T).

% Controla geração de output:
#hide.
#show posicao(N,D,L,C).
%#show ocupado(N,L,C).
%#show somalinha(L,S).
%#show somacoluna(C,S).
```

```
% São gerados 16 modelos diferentes correspondendo basicamente a 2 soluções distintas.
% Isto é motivado por simetria das soluções, uma de fácil resolução e outra mais complexa.

% Primeiro, colocar um submarino na vertical ou na horizontal é a mesma coisa.
% Podem-se eliminar os modelos em que o submarino é colocado na vertical com a restrição

:- posicao(N,v,L,C), linha(L), coluna(C), comprimento(N,1).

% Obtendo-se os modelos:
Answer: 1
posicao(a,h,1,1) posicao(b,h,4,4) posicao(c,h,5,1) posicao(d,h,2,3)
Answer: 2
posicao(a,h,1,5) posicao(b,h,5,1) posicao(c,v,4,4) posicao(d,h,2,1)
Answer: 3
posicao(a,h,5,1) posicao(b,h,1,5) posicao(c,v,4,4) posicao(d,h,2,1)
Answer: 4
posicao(a,h,4,4) posicao(b,h,1,1) posicao(c,h,5,1) posicao(d,h,2,3)

% Como os navios têm nome vão-se obter modelos geometricamente iguais mas
% conceptualmente diferentes. É possível eliminar estes modelos mas exige uma modelação muito
% mais complexa para evitar a utilização de nomes. Podem tentar...
```



Conjuntos de resposta 2 e 3



Conjuntos de resposta 1 e 4