MAC0425 Inteligência Artificial — Prova Substitutiva (PSUB)

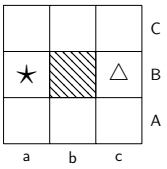
NOME: **GABARITO** NUSP: **04252019**

ASSINATURA: .....

INSTRUÇÕES:

- Leia atentamente os enunciados.
- Este exame é composto por questões dissertativas; todas as respostas devem ser justificadas.
- Escreva suas respostas nos locais indicados, **utilizando caneta esferográfica azul ou preta**.
- Use os versos das folhas como rascunho; escreva sua reespota apenas quando tiver certeza.
- Você pode consultar apenas uma folha A4 (frente e verso) individual.
- O uso de equipamentos eletrônicos (calculadoras, celulares, computadores) não é permitido.
- **Esse exame contém 7 questões valendo 2 pontos cada. Você deve resolver pelo menos 5 delas para conseguir nota máxima.**
- **Duração:** 120 minutos.

As Questões 1 a 3 referem-se ao mundo de células (*gridworld*) representado na figura ao lado, no qual um agente inicialmente em △ deseja chegar à célula ⋆ com o menor número de ações possível. As células hachuradas indicam locais inacessíveis. A cada "turno" o agente se move para alguma célula adjacente válida (acima, abaixo, esquerda ou direita). Não há possibilidade de permancer na mesma célula em um turno.



Questão 1 Formalize o problema como uma busca em espaço de estados determinística. Descreva o conjunto de estados S, o estado inicial s_0, os estados meta M, as ações aplicáveis $A(s)$, a função de transição $T(s, a)$ e a função de custo $C(a_1, \ldots, a_n)$.

Podemos representar um estado como um par ordenado (x, X) onde $x \in \{a, b, c\}$ e $X \in \{A, B, C\}$. Por simplicidade, escrevemos também xX para denotar (x, X). Dessa forma o **conjunto de estados** é

$$S = \{a, b, c\} \times \{A, B, C\} - bB\,,$$

onde $\times$ representa o produto direto (Cartesiano). O **estado inicial** é

$$s_0 = cB\,,$$

e os **estados-meta** é

$$M = \{aB\}\,.$$

As **ações** são c, b, d, e representando, nessa ordem, deslocamentos para célula acima, abaixo, à esquerda e à direita da posição atual (estado) do agente. As **ações aplicáveis** são:

$A(s) =$

	A	B	C
a	c,d	–	b,d
b	e,d	–	e,d
c	e,c	c,b	e,b

A **função de transição** é dada por

$$T(xX, e) = (x - 1, X) \qquad T(xX, d) = (x + 1, X)$$
$$T(xX, c) = (x, X + 1) \qquad T(xX, b) = (x, X - 1)$$

onde $x + 1$ representa a próxima letra e de forma similar para outras expressões.
Por fim, a **função de custo** indica o número de ações: $C(a_1, \ldots, a_n) = n$.

Questão 2 Assuma agora que o agente executa, a cada turno t, uma ação aplicável qualquer com probabilidade uniforme, que é sempre bem sucedida. Assuma também que o agente não consegue observar diretamente a posição em que se encontra, mas possui um sensor que retorna um número Y_t entre 1 e 3 com as probabilidades abaixo:

$P(Y_t = 1|X_t)$

	a	b	c
C	2/3	0	2/3
B	0		0
A	2/3	0	2/3

$P(Y_t = 2|X_t)$

	a	b	c
C	1/3	1/3	1/3
B	1/3		1/3
A	1/3	1/3	1/3

$P(Y_t = 3|X_t)$

	a	b	c
C	0	2/3	0
B	2/3		2/3
A	0	2/3	0

$P(X_0|Y_0 = 3)$

	a	b	c
C	0	0	0
B	0		1
A	0	0	0

A figura mais à direita representa a crença $P(X_0|Y_0 = 3)$ sobre a posição do agente no turno $t = 0$ (isto é, antes de tomar qualquer ação) após observar $Y_0 = 3$. A equação abaixo descreve essa distribuição em função dos parâmetros do modelo oculto de Markov do ambiente.

$$P(X_0 = x_0|Y_0 = 3) = \frac{P(Y_0 = 3|X_0 = x_0)P(X_0 = x_0)}{\sum_{x_0'} P(Y_0 = 3|X_0 = x_0')P(X_0 = x_0')}\,.$$

Complete as equações abaixo em função dos parâmetros do modelo, e preencha as figuras com os valores de probabilidades correspondentes.

	a	b	c
C	0	0	1/2
B	0		0
A	0	0	1/2

$P(X_1|Y_0 = 3) = \sum_{x_0} P(X_1|X_0 = x_0)P(X_0 = x_0|Y_0 = 3)$

Na equação acima, usamos o fato de que X_1 e Y_0 são condicionalmente independentes dado X_0. Como o agente se move com probabilidade uniforme para célula vizinha, e iniciamente se encontra em cB de acordo com $P(X_0|Y_0)$, a distribuição acima é não nula apenas em cC e cA

	a	b	c
C	0	0	1/2
B	0		0
A	0	0	1/2

$P(X_1|Y_1 = 1, Y_0 = 3) = \frac{P(Y_1=1|X_1)P(X_1|Y_0=3)}{P(Y_1=1|Y_0=3)} = \frac{P(Y_1=1|X_1)P(X_1|Y_0=3)}{\sum_{x_1'} P(Y_1=1|X_1=x_1')P(X_1=x_1'|Y_0)}$

Para ober a distribuição na figura, multiplicamos $P(X_1|Y_0 = 3)$ obtido acima por $P(Y_1 = 1|X_1)$ indicado no enunciado e então *renormalizamos* para que a soma seja igual a um. Como apenas as posições cC e cA são não nulas, e o valor de $P(Y_1 = 1|X_1) = 2/3$ nelas, chegamos à distribuição ao lado.

	a	b	c
C	0	1/4	0
B	0		1/2
A	0	1/4	0

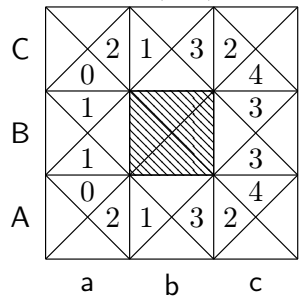
$P(X_2|Y_2 = 2, Y_1 = 1, Y_0 = 3) = \frac{P(Y_2=2|X_2)\sum_{x_1} P(X_2|X_1)P(X_1|Y_1,Y_0)}{\sum_{x_2'} P(Y_2|X_2=x_2')\sum_{x_1} P(X_2=x_2'|X_1=x_1)P(x_1|Y_1,Y_0)}$

Mais uma vez, para chegarmos a distribuição na figura ao lado, computamos a espressão no numerador da equação acima e renormalizamos para que a soma seja um.

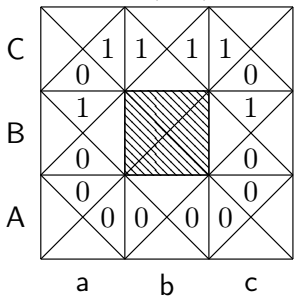
Questão 3 Assuma agora que o agente pode observar diretamente sua posição, mas desconhece os efeitos e os custos/recompensas de suas ações. Considere as seguintes funções características $f_1(s,a)$ e $f_2(s,a)$ (os espaços em branco indicam ações não aplicáveis):

$A(s)$

$f_1(s,a)$



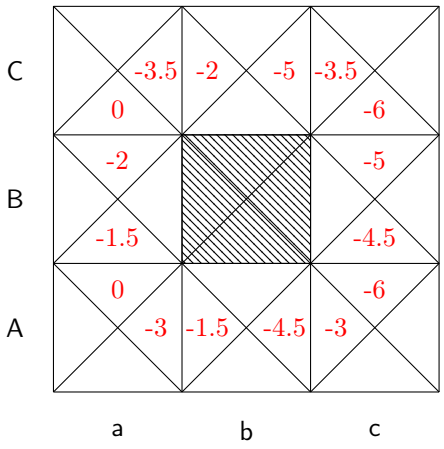
$f_2(s,a)$



t	0	1	2
S_t	cB	cC	bC
A_t	↑	←	←
R_t	-1	-1	-1

Suponha que o agente interage com o ambiente obtendo a sequência de estados, ações e recompensas (nessa ordem) listadas na tabela à direita acima. Complete as figuras abaixo com os valores de $Q^{(t)}(s,a)$ *após* cada transição acima usando o algoritmo *Q-Learning aproximado* com taxa de aprendizado $\alpha = 1/2$, fator de desconto $\gamma = 1$ e valores iniciais $w_1^{(0)} = w_2^{(0)} = 0$. Detalhe como os pesos foram atualizados nos espaços indicados.

$Q^{(1)}(s,a)$



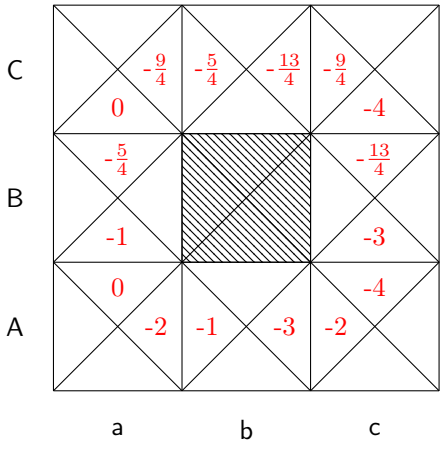
$$w_k^{(t+1)} = w_k^{(t)} + \alpha[R_t + \gamma \max_a Q^{(t)}(S_{t+1}, a) - Q^{(t)}(S_t, A_t)] f_k(S_t, A_t)$$

$$w_1^{(1)} = \underset{=0}{w_1^{(0)}} + \underset{=1/2}{\alpha}[\underset{=-1}{R_0} + \gamma \underset{=0}{Q^{(0)}}(cC, a) - \underset{=0}{Q^{(0)}}(cB, \uparrow)] \underset{=3}{f_1}(cB, \uparrow) = -3/2$$

$$w_2^{(1)} = \underset{=0}{w_2^{(0)}} + \alpha[R_0 + \gamma Q^{(0)}(cC, a) - Q^{(0)}(cB, \uparrow)] \underset{=1}{f_2}(cB, \uparrow) = -1/2$$

$$Q^{(1)}(s,a) = -\tfrac{3}{2} f_1(s,a) - \tfrac{1}{2} f_2(s,a)$$

$Q^{(2)}(s,a)$

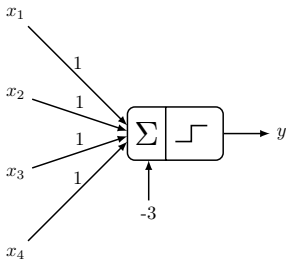


$$w_1^{(2)} = -1.5 + \tfrac{1}{2}\underbrace{[-1 + 1 \cdot \max(-2, -5) - (-3.5)]}_{=0.5} \cdot 2 = -1$$

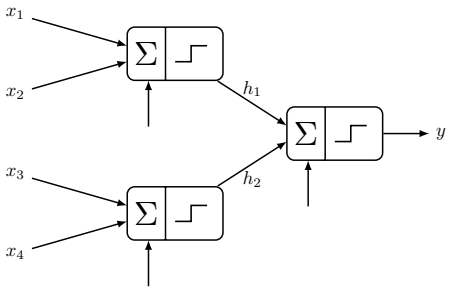
$$w_2^{(2)} = -0.5 + \tfrac{1}{2} \cdot 0.5 \cdot 1 = -1/4$$

$$Q^{(2)}(s,a) = -f_1(s,a) - \tfrac{1}{4} f_2(s,a)$$

Questão 4 As redes neurais abaixo usam função de ativação $\text{sgn}(x) = 1$ se $x \geq 0$ e $\text{sgn}(x) = -1$ se $x < 0$ e recebem entradas $(x_1, x_2, x_3, x_4) \in \{0,1\}^4$. Explique porque a rede B não consegue computar a função computada pela rede A, apesar de B possuir mais camadas que A. Tente ser o mais formal possível na sua explicação.



(A)



(B)

ERRATA: A questão original especificava a função de ativição erroneamente como $\text{sgn}(x) = \max(0, \min(1, x))$, fazendo com que a rede (A) computasse o equivalente a uma função de conjunção booleana. Tal função é claramente computável por uma rede com a estrutura da rede (B), especificando cada neurônio para computar uma conjunção de suas entradas. O erro está em assumir que $\text{sgn}(0) = 0$ e não 1. A solução abaixo assume a função de ativação do enunciado acima, que retifica esse erro. Para efeitos de correção, foram consideradas corretas soluções que indicaram a falha, ou que apontaram que a rede não poderia computar uma função de 4 variáveis através da combinação de funções de 2 variáveis (ainda que isto esteja incorreto).

A rede (A) computa a função

$$f_A(x_1, x_2, x_3, x_4) = \begin{cases} 1 & \text{caso } \sum_i x_i \geq 3, \\ 0 & \text{caso } \sum_i x_i < 3. \end{cases}$$

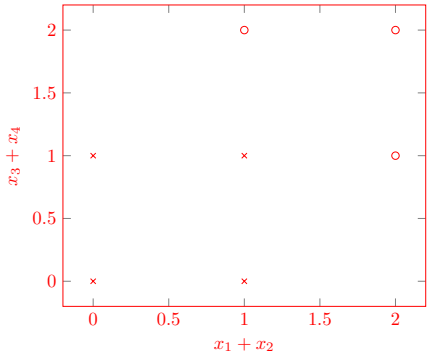
Chame de $h_1(x_1, x_2)$ e $h_2(x_3, x_4)$ as saídas dos neurônios ocultos da rede (B) e de $y_B(h_1, h_2)$ a saída da rede. Defina

$$\begin{array}{llll} a_1 = h_1(0,0) & b_1 = h_1(1,0) & c_1 = h_1(0,1) & d_1 = h_1(1,1) \\ a_2 = h_2(0,0) & b_2 = h_2(1,0) & c_2 = h_2(0,1) & d_2 = h_2(1,1) \end{array}$$

onde $a_i, b_i, c_i, d_i \in \{0,1\}, i = 1, 2$. Por simplicidade, assuma que $a_1 = a_2 = a$, $b_1 = b_2 = b$ e assim por diante (isso não é necessário). Para que a rede (B) compute a função f_A, devemos ter

$$y_B(b,d) = y_B(d,b) = y(d,d) = 1 \qquad \text{e} \qquad y_B(a,a) = y_B(b,b) = y_b(a,b) = y_B(b,a) = y_B(a,d) = y_B(d,a) = 0\,.$$

As equações acima não possuem solução. Por exemplo, com $y(a,a) \neq y(d,d)$, devemos ter $a \neq d$. E como $y(a,b) \neq y(b,b)$ devemos ter $a \neq b$. Isso implica em $b = d$ o que viola a desigualdade $y(b,b) \neq y_B(d,d)$.
Para valores quaisquer $a_1, a_2, \ldots$ note que precisamos distinguir quando a soma das entradas de h_1 ou h_2 é 0, 1 e 2, mas h_1 e h_2 só tomam valores 0 e 1. A figura abaixo ilustra o raciocínio. Pontos "o" representam saída 1 e pontos "x" representam saída 0. Qualquer forma de mapear valores 0, 1 e 2 em valores 0 e 1 faz com que pontos de classes distintas se sobreponham (e portanto não possam ser separados por reta).



Questão 5 Uma modificação comum ao classificador Perceptron Probabilístico é encontrar pesos $w = (w_0, w_1, \ldots, w_d)$ que minimizam a função objetivo

$$J(w) = -\sum_{j=1}^{N} \ln \Pr(Y = y_j | X = x_j) + \lambda(w \cdot w)\,,$$

onde $\lambda > 0$ é um parâmetro a ser especificado, $\Pr(Y = y|x) = 1/(1 + \exp(-y(w \cdot x)))$ e $\cdot$ indica produto escalar.

1. Obtenha as equações de atualização dos pesos para um algoritmo de aprendizado por descida do gradiente em lote (*batch gradient descent*) que otimiza a função objetivo J.

2. Para um conjunto de treinamento linearmente separável, esse algoritmo sempre aprende um classificador que classifica corretamente todos os dados de treinamento? Justifique.
   Dica: Pense o que acontece para o conjunto $x_1 = -1, y_1 = -1, x_2 = 1, y_2 = 1$.

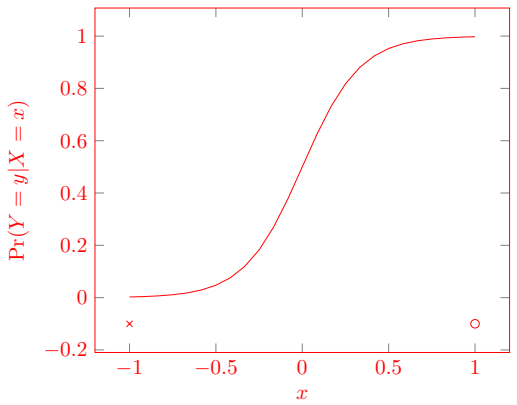
1. O algoritmo de descida do gradiente atualiza os pesos por:

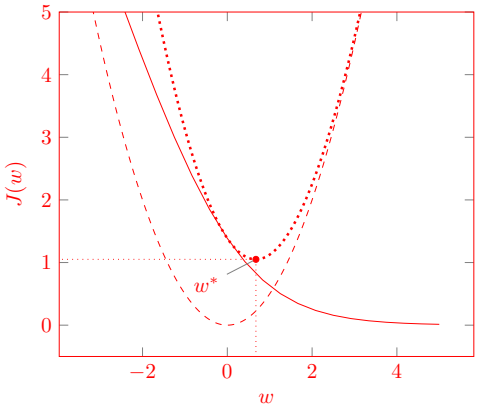
$$w_k^{(i+1)} = w_k^{(i)} - \alpha \left.\frac{\partial J(w)}{\partial w_k}\right|_{w=w^{(i)}}$$

onde $k = 1, \ldots, d$, α é taxa de aprendizado e

$$\frac{\partial J(w)}{\partial w_k} = -\sum_{j=1}^{N} \frac{e^{-y_j(w \cdot x_j)}}{1 + e^{-y_j(w \cdot x_j)}}(y_j x_{jk}) + 2\lambda w_k = -\sum_{j=1}^{N} [1 - \Pr_w(Y = y_j | X = x_j)] y_j x_{jk} + 2\lambda w_k\,.$$

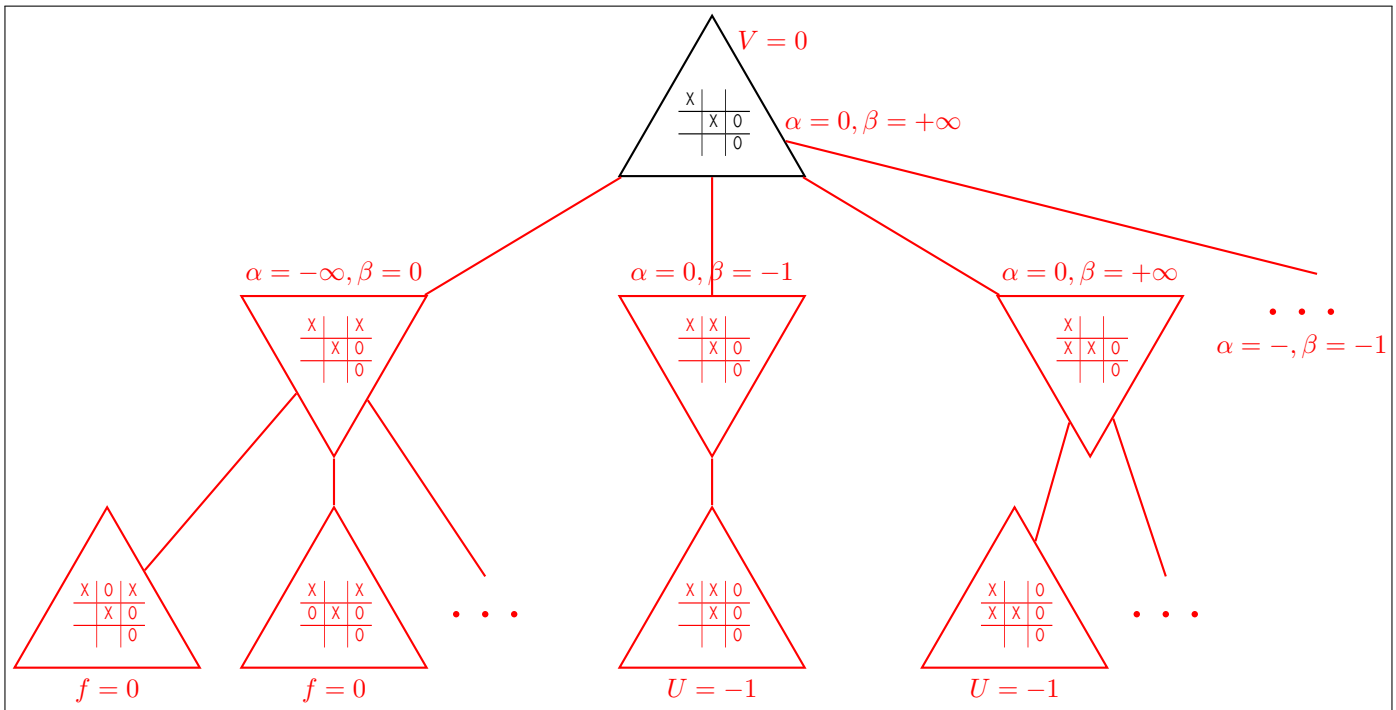
2. Considere o conjunto de pontos $\{(-1,-1),(1,1)\}$, ilustrado na figura abaixo à esquerda (junto com a função de classificação do perceptron probabilístico, com viés $w_0 = 0$ e $w_1 = 6$). Esse conjunto é claramente separavél por qualquer hiperplano com $w_0 = 0$ e $w_1 > 0$. A função objetivo para esse conjunto assumindo pesos $(0, w)$ é $J(w) = 2\log(1 + e^{-w}) + \lambda w^2$. Essa função é exibida na figura à direita abaixo na curva pontilhada. A curva sólida representa o primeiro termo, e a curva tracejada representa o segundo termo (com $\lambda = 0.5$). Intuitivamente, para $\lambda = 0$, o peso ótimo é $w \to \infty$, o que faz com que os pontos sejam classificados com probabilidade 1. Como o termo quadrático λw^2 é uma função par (i.e., simétrica com relação ao eixo das ordenadas), a função $J(w)$ é minimizada para $w > 0$, porém finito, como pode ser visto no exemplo no gráfico à direita abaixo. O valor ótima depende do valor de λ; quanto maior λ, menor o valor ótimo de w, mas sempre positivo.





De forma mais geral, se os dados são linearmente separáveis, então sempre existe uma configuração de pesos w que classificam todos os pontos com corretamente. Ademais, se w é um hiperplano separador, então $c \cdot w/(w \cdot w)$ para $c > 0$ é um hiperplano separador de norma c. Isso indica que podemos reduzir a norma do vetor de pesos sem alterar suas classificações. Dessa forma minimizamos tanto o primeiro termo (que procura maximizar a norma de w para aumentar a probabilidade das classificações) e o segundo termo (que minimiza a norma de w) quando selecionamos um vetor w que separa os pontos. A função $J(w)$ é minimizada portanto em um hiperplano separador w que equilibra a norma do vetor com o negativo da verossimilhança.
O algoritmo de descida do gradiente encontra o valor ótimo de w^* (para uma taxa adequada de aprendizado) pois a função $J(w)$ é convexa. Isso pode ser constatado notando que o termo *regularizador* $\lambda w \cdot w$ é quadrático (portanto convexo), e o negativo da verossimilhança é convexo (visto em aula); a soma de duas funções convexas é também convexa.

Questão 6 Considere um jogador para o jogo da velha que escolhe suas ações através de uma busca minimax com poda (ou seja, busca alfa-beta) e profundidade de árvore limitada a 2. Assuma que jogamos como X. Os nós terminais possuem utilidade $+1$ se X é o vencedor, -1 se O é o vencedor e 0 em caso de empate. Nós não terminais são avaliados através da função de avaliação $f(n) = C(\mathsf{X}) - C(\mathsf{O})$, onde $C(\mathsf{X})$ denota o número de Xs e $C(\mathsf{O})$ denota o número de Os. Por exemplo, o nó n abaixo possui $f(n) = 2 - 2 = 0$. Desenhe a árvore de busca para uma busca alfa-beta de profundidade 2 iniciando no nó abaixo. Assuma que os nós são expandidos da esquerda para a direita, e escolha a ordem de expansão a fim de podar o maior número de nós. Você pode agrupar conjuntos de folhas com o mesmo valor para poupar espaço.



A única expansão que não gera um nó terminal de valor $U = -1$ é quando colocamos X na coluna da direita, obtendo um nós não terminais de valor $f = 0$. Portanto, para maximizarmos o número de podas devemos primeiro expandir o nó mecionado completamente, para que o valor α do nó raíz seja atualizado para 0. Ao expandir os demais nós, devemos sempre expandir primeiro o que coloca O na coluna da direita, o que nos faz podar as restantes expansões locais pois $\alpha > \beta$ nesse instante.

Questão 7 Considere um problema de busca determinístico dinâmico, ou seja, onde a especificação do modelo é alterada durante a execução do algoritmo de busca (por exemplo, quando desejamos encontrar a rota mais curta e os tempos de trânsito são atualizados constantemente). Explique como podemos modificar o algoritmo A^* para que ele expanda o menor número de nós quando uma única transição tem seu valor de custo de passo $C(s, a)$ aumentado. Use a formulação do A^* que lhe for mais conveniente.

Vamos assumir um algoritmo A^* em árvore (ou seja, sem memória de nós repetidos), e que a árvore de busca é mantida.
Vamos primeiro assumir que a busca tinha chegado ao fim (isto é, um caminho ótimo havia sido encontrado), quando o custo foi modificado. Temos portanto dois casos.
Caso 1: O custo de passo aumentado não faz parte do caminho ótimo encontrado. Nesse caso a busca não precisa ser refeita, pois o caminho ótimo não é alterado.
Caso 2: O custo de passo aumentado se encontra no caminho ótimo. Nesse caso devemos remover todos os nós obtidos a partir da expansão de s, e recomeçar a busca com a fronteira definida pelos nós folhas.
Para o caso no qual a busca ainda não havia terminado, devemos executar o caso 2 se o estado s estava associado a algum nó árvore, caso contrário simplesmente continuamos com a busca.