Introdução à Inteligência Artificial
Ano 2005/06 – Época normal
3 horas / com consulta

Grupo 1 (Pesquisa)

A gestão atempada dos prazos é um problema recorrente na vida real. Considere-se um conjunto de tarefas **T**, em que cada tarefa **t** de **T** é caracterizada por uma duração **d(t)**, um tempo limite de conclusão **l(t)** e uma penalização por atrasos **p(t)**, todos eles números inteiros positivos. O objectivo é encontrar um escalonamento das tarefas que minimiza a penalização total, sendo que um escalonamento consiste em encontrar para cada tarefa **t** o instante de tempo **s(t)** em que esta se inicia (este instante é também um inteiro positivo). Cada tarefa **t** é executada entre o tempo **[s(t),s(t)+d(t)[** e não podem estar duas tarefas simultaneamente em execução. As tarefas são independentes e portanto podem ser executadas por qualquer ordem.
Cada tarefa que ultrapasse o limite estipulado dá origem à respectiva penalização: se **s(t)** + **d(t)** > **l(t)** então o custo (penalização) da tarefa **t** é de **p(t)**. A penalização total é obtida pela soma da penalização de cada tarefa atrasada. Considere-se a seguinte instância:

Tarefa	Duração	Limite	Penalização
t1	**d(t1)**=4	**l(t1)**=10	**p(t1)**=2
t2	**d(t2)**=3	**l(t2)**=20	**p(t2)**=3
t3	**d(t3)**=3	**l(t3)**=6	**p(t3)**=2
t4	**d(t4)**=2	**l(t4)**=5	**p(t4)**=4
t5	**d(t5)**=10	**l(t5)**=12	**p(t5)**=1

Por exemplo, o escalonamento **s(t1)**=3, **s(t2)**=10, **s(t3)**=7, **s(t4)**=1, **s(t5)**=13 tem uma penalização total de 3, originada pela não conclusão atempada das tarefas **t3** e **t5**.

1a) Formule claramente o problema para ser resolvido recorrendo a algoritmos de pesquisa em espaço de estados, indicando o estado inicial, teste de estado objectivo e função que devolve os sucessores de um estado, não esquecendo de indicar o custo dos operadores. Indique também o número total de estados objectivo, quando existem **n** tarefas para escalonar.

1b) Apresente uma heurística não constante que garanta a obtenção de uma solução óptima pelo algoritmo A*. Não é necessária a apresentação de qualquer expressão matemática, desde que fique absolutamente claro a forma como a heurística pode ser obtida. Justifique adequadamente.

1c) Considere a variante do problema anterior em que estamos interessados em saber somente se existe, ou não, um escalonamento sem penalização (penalização total 0). Indique se o problema pode ser entendido como um problema de satisfação de restrições (CSP), indicando a sua formulação caso o seja.

Grupo 2 (Redes de Bayes)

O "copianço" é uma pecha de qualquer sistema educativo, tendo sido alvo de estudos recentes. Sabe-se que 10% da população estudantil frequenta o Ensino Universitário, 30% o Ensino Secundário, e os restantes o Ensino Básico. Dos alunos que frequentam a Universidade 60% copiam, sendo a percentagem dos alunos que copiam no Ensino Secundário de 80%; os alunos do Básico não copiam. Todos os alunos do Secundário já viram os seus colegas a copiar alguma vez, descendo esta percentagem para 80% no caso dos Universitários. Quanto aos alunos do Básico, 10 % afirma que já viram os seus colegas a copiar (os alunos do básico às vezes mentem). Os alunos do básico nunca estudam para as provas, enquanto que 50% dos universitários e 50% dos alunos do secundário diz que o faz. Os alunos que copiam e estudam para as provas sentem-se penalizados na nota em 10% dos casos, descendo esta percentagem para 1% para os alunos que estudam mas não copiam. Os alunos que não estudam para as provas nunca se sentem penalizados na nota.

2a) Modele a situação anterior com uma rede de Bayes, indicando as variáveis aleatórias, seus domínios, topologia da rede e tabelas de probabilidade condicionada.

2b) Calcule a probabilidade de um aluno copiar.

2c) Calcule a probabilidade de um aluno frequentar o ensino secundário dado que viu algum colega a copiar e que se sentiu penalizado na nota.

Grupo 3 (Planeamento)

O Sr. Cuidadoso vai efectuar uma viagem por Auto-Estrada entre Alverca e Santarém. O Sr. Cuidadoso recorre a um sistema de planeamento de viagens, desenvolvido por ele, que utiliza a linguagem STRIPS. As acções conhecidas pelo sistema de planeamento são as seguintes:

```
accao entrar(AE,L)
condicoes [portagem(AE,L),local(L)]
efeitos [entrou(AE,L)]
restricoes [].

accao seguir(AE,L2)
condicoes [troço(AE,L1,L2),entrou(AE,_),local(L1)]
efeitos [-local(L1),local(L2)]
restricoes [].

accao sair(AE,L)
condicoes [portagem(AE,L),local(L),entrou(AE,L1)]
efeitos [-entrou(AE,L1),fora]
restricoes [L \== L1].
```

O estado inicial contém a seguinte informação:

```
inicial[ local(alverca),fora,
         portagem(a1,alverca), portagem(a1,aveiras), portagem(a1,santarém),
         troço(a1,alverca,aveiras), troço(a1,aveiras,santarém),
         estacao(a1,serv1,aveiras)
].
```

3a) Verifique se as acções anteriores permitem obter linearizações de planos POP em que se pode entrar consecutivamente na mesma auto-estrada sem sair dela, como por exemplo nas duas sequências seguintes para atingir um estado em que **local(santarém)** e **fora** são verdadeiros:

[entrar(a1, alverca), entrar(a1, alverca), seguir(a1, aveiras), seguir(a1, santarém),sair(a1,santarém)] ou mesmo **[entrar(a1, alverca), entrar(a1, alverca), seguir(a1, aveiras), seguir(a1, santarém)]**

Justifique sucintamente como estas duas sequências poderiam ser obtidas, caso seja possível. Proponha alterações às acções apresentadas no enunciado de modo a eliminar as referidas situações indesejáveis, caso ocorram.

3b) Para almoçar numa estação de serviço, o Sr. Cuidadoso tem de estar atrasado. O Sr. Cuidadoso só se atrasa quando ocorre um acidente com outro condutor, situação capturada pela acção:

```
accao acidente
condicoes []
efeitos [ atrasado ]
restricoes [].
```

Nas Auto-Estradas só se pode almoçar nas estações de serviço, descritas no estado inicial por instâncias do tipo **estacao(AE,EST,L)**, em que **AE** é o identificador da Auto-Estrada, **EST** o identificador da estação de serviço que ocorre no nó **L** de **AE**. Para simplificar o problema, assuma que o Sr. Cuidadoso pode almoçar na estação de serviço desde que se encontre no local **L** (e esteja atrasado).
Modele a acção **almoçar(AE,EST)** na linguagem STRIPS, em que AE e EST são os identificadores da Auto-Estrada e da estação de serviço. Construa um plano com ordem parcial, como o algoritmo POP o faria, em que o Sr. Cuidadoso chega a Santarém devidamente almoçado. Apresente todas as linearizações deste plano. De preferência, utilize a representação obtida na alínea **3a**) para modelar as outras acções.

3c) Comente a seguinte afirmação: "Planos gerados pelo algoritmo POP para as acções do enunciado, estado inicial indicado e objectivos **local(santarém)** e **fora** só têm uma linearização possível". Justifique detalhadamente.

Grupo 4 (Torneio de Futebol)

Efectuou-se um torneio de futebol envolvendo os seguintes clubes de futebol: Aguiense, Belavista, Cruzense, e Dragoense. O torneio foi efectuado a uma só volta, tendo cada equipa jogado apenas uma vez com cada um dos três adversários. Foram realizados os seguintes seis jogos de futebol:

Aguiense-Dragoense	Cruzense-Aguiense
Belavista-Cruzense	Belavista-Dragoense
Dragoense-Cruzense	Aguiense-Belavista

Sabe-se ainda que:

i1. O Dragoense teve resultados iguais nos jogos que efectuou com o Aguiense e o Cruzense.
i2. O Cruzense ganhou ao vencedor do jogo Aguiense-Dragoense.
i3. O Belavista não ganhou qualquer jogo.
i4. O Aguiense perdeu com o vencedor do jogo Dragoense-Cruzense.
i5. O Aguiense empatou com o derrotado do jogo Belavista-Dragoense
i6. O Cruzense nunca empatou um jogo fora.

Dados os seus conhecimentos de programação por conjuntos de resposta, os organizadores do torneio pediram-lhe ajuda no desenvolvimento de um programa, para o sistema Smodels, que lhes permita saber o resultado de cada jogo. Considere o seguinte programa, representando alguma da informação enunciada:

```
equipa(a).              jogo(a,d).              res(1).
equipa(b).              jogo(b,c).              res(x).
equipa(c).              jogo(d,c).              res(2).
equipa(d).              jogo(c,a).
                        jogo(b,d).
                        jogo(a,b).

resultado(C,V,R) :- jogo(C,V), res(R),  not neg_resultado(C,V,R).
neg_resultado(C,V,R) :- jogo(C,V), res(R),  not resultado(C,V,R).
```

O predicado **equipa/1** lista as quatro equipas em competição, em que **a** significa Aguiense, **b** designa o clube Belavista, etc. O predicado **jogo/2** descreve os seis jogos realizados, em que o primeiro argumento é a equipa da casa e o segundo argumento a equipa visitante. O predicado **res/1** explicita os três resultados possíveis para um jogo de futebol: **1** – vitória da equipa da casa; **x** – empate; e **2** – vitória da equipa visitante (a que joga fora).

4a) Indique qual a função das duas regras que constam do programa apresentado acima. Quantos conjuntos de resposta existem para esse programa (factos mais regras) ?

4b) Adicione restrições ao programa apresentado, **que garantam** que para cada jogo exista um e um só resultado.

4c) Compare o efeito das seguintes restrições, indicando os conjuntos de resposta eliminados por cada uma delas.

a) `:- resultado(a,d,x).`
b) `:- not resultado(a,d,1), not resultado(a,d,2).`

4d) Apresente uma ou mais restrições que capturem a informação descrita na afirmação **i1** do enunciado (repare que num dos jogos o Dragoense joga em casa e noutro é visitante).

4e) Para representar as afirmações **i2** a **i5**, sugere-se a construção do predicado **vencedor/3**, em que uma instância **vencedor(C,V,X)** pertence a um conjunto de resposta quando **X** é o vencedor do jogo (**C**,**V**). Defina o predicado **vencedor/3**. Estando definido o predicado **vencedor/3**, o predicado **derrotado/3** pode ser definido com as seguintes regras:

```
derrotado(C,V,V) :- jogo(C,V), vencedor(C,V,C).
derrotado(C,V,C) :- jogo(C,V), vencedor(C,V,V).
```

4f) Apresente restrições que capturem a informação descrita nas afirmações **i2** a **i6**, podendo utilizar os predicados seguintes e quaisquer dos definidos anteriormente, de maneira a que o (único) conjunto de resposta do programa obtido corresponda à (única) solução do problema.

```
ganhou(E1,E2) :- jogo(E1,E2), resultado(E1,E2,1).
ganhou(E1,E2) :- jogo(E2,E1), resultado(E2,E1,2).

perdeu(E1,E2) :- jogo(E1,E2), resultado(E1,E2,2).
perdeu(E1,E2) :- jogo(E2,E1), resultado(E2,E1,1).

empatou(E1,E2) :- jogo(E1,E2), resultado(E1,E2,x).
empatou(E1,E2) :- jogo(E2,E1), resultado(E2,E1,x).
```

FIM