Ficha 1: Introdução à Inteligência Artificial e aos Agentes Inteligentes

1. Considere a lista de tarefas seguinte:
   1) Jogar razoavelmente um jogo de Ténis (ou Ping-Pong).
   2) Jogar um jogo de Poker (jogo de cartas) como um profissional.
   3) Conduzir um automóvel numa auto-estrada pouco movimentada.
   4) Conduzir um automóvel no centro de Lisboa.
   5) Inventar uma anedota com piada.
   6) Traduzir de Português para Inglês em tempo real.
   7) Fornecer aconselhamento legal numa dada área.
   8) Discutir política internacional com o primeiro ministro de Portugal.

   1.1. Para cada uma das tarefas anteriores discuta se podem, actualmente, ser executadas por computadores.

   1.2. Para cada uma das tarefas anteriores argumente quais as dificuldades que indicou que não podem ser executadas por computadores.

2. Um programa em Inteligência Artificial pode ser encarado como (escolha a opção mais apropriada da lista seguinte):
   2.1. Uma rede neuronal artificial ou biológica a processar informação sob a forma de impulsos nervosos.
   2.2. Uma fórmula.
   2.3. Um agente inteligente.

3. O ambiente do jogo das damas pode ser considerado (escolha sim/não para cada opção da lista seguinte):
   3.1. Parcialmente Observável.
   3.2. Estocástico.
   3.3. Contínuo.
   3.4. Adverso (ambiente multi-agente com agentes competitivos).

4. O ambiente do jogo do poker pode ser considerado (escolha sim/não para cada opção da lista seguinte):
   4.1. Parcialmente Observável.
   4.2. Estocástico.
   4.3. Contínuo.
   4.4. Adverso (ambiente multi-agente com agentes competitivos).

5. O ambiente correspondente à condução autónoma de um automóvel por parte de um agente artificial pode ser considerado (escolha sim/não para cada opção da lista seguinte):
   5.1. Parcialmente Observável.
   5.2. Estocástico.
   5.3. Contínuo.
   5.4. Adverso (ambiente multi-agente com agentes competitivos).

6. Especifique, usando pseudo-código, um agente simples reflexo, TEMPERATUREX, para controlar a temperatura de uma sala.

6.1. Especifique o agente, usando pseudo-código.
Suponha que dispõe das seguintes percepções:
- T1 e T2 correspondentes à temperatura da sala e à temperatura exterior.

Suponha que dispõe das seguintes acções:
- AQ ligar o aquecedor, NAQ – Desligar o aquecedor;
- AC – ligar o ar frio, NAC – Desligar o ar frio;
- AJ – abrir as janelas, NAJ – fechar as janelas.

Pretende-se que a temperatura da sala esteja entre os 22 e os 24 graus. Sempre que seja possível usar as janelas para controlar a temperatura (não desperdiçando energia), tal deve ser efectuado. Sempre que a temperatura da sala esteja mais de 2 graus afastada da banda desejada (ou seja se a temperatura for inferior a 20 ou superior a 26 graus), deve-se fechar as janelas e em vez disso, usar o aquecedor ou ar frio para repor a temperatura dentro da banda desejada.
Use termos linguísticos para caracterizar a situação.

6.2. Implemente o agente usando a biblioteca aima-java. Defina no simulador (o ambiente), por exemplo, leis lineares de variação da temperatura da sala em função da temperatura exterior e das condições restantes como aquecedor ligado ou desligado e outras. Um exemplo de uma lei poderá ser: caso a janela esteja fechada, ar frio desligado e aquecedor ligado a temperatura irá subir 0,5ºC por cada ciclo de simulação.

6.3. Sugira como poderia construir um agente um pouco mais inteligente para este problema (que tipo de agente, percepções, estado do mundo, etc., usar)?

7. O agente Bouncex move-se num espaço 2D onde podem existir obstáculos do tipo parede. O espaço é limitado. Quando não choca com nenhum obstáculo o agente desloca-se em linha recta. Caso choque com um obstáculo, afasta-se na direcção que faz um ângulo simétrico com o ângulo de incidência (exemplo: se ângulo de incidência = 30º então ângulo de saída = -30º). O agente pode percepcionar o espaço à sua frente a uma unidade de distância determinado a existência ou não de um obstáculo. As suas acções resumem-se a avançar ou rodar *x* graus caso encontre um obstáculo. Modelize o comportamento deste agente por meio de um sistema de produção (regras). Acha que poderá ocorrer alguma situação/comportamento bizarro?

8. O agente Bulimex vive num mundo dividido em células. Tem um apetite insaciável e passa o tempo à procura de comida. As suas capacidades sensoriais permitem-lhe determinar a quantidade de comida (número inteiro eventualmente nulo) que se encontra nas suas oito células vizinhas. As acções que pode efectuar são deslocar-se em frente, rodar 45º para a esquerda ou para a direita e comer. Cada acção que efectua consome uma unidade de energia. Quando come a sua energia é aumentada da quantidade de comida que ingeriu. Veio ao mundo com uma energia *n*. Especifique um sistema de produção para o agente Bulimex de modo a satisfazer o seu apetite voraz. Em que medida a solução obtida procura maximizar a quantidade de energia do agente?

9. O agente Covardex vive num mundo 2D, partilhado com outro agentes, dividido em células rectangulares. Todos possuem energia que se vai dissipando por cada acção tomada. O nosso agente tem capacidade para percepcionar a sua energia e a célula à sua frente, determinando neste caso a existência ou não de agentes nessa células e respectiva energia. Se a energia for maior do que a sua o Covardex foge, se for menor Covardex toma de assalto a célula e apodera-se da energia do ocupante, que morre

entretanto. As suas acções primárias são avançar para a célula para onde está virado, rodar 90º para a direita ou para a esquerda e comer o agente da célula para onde se desloca. Modelize o comportamento do agente através de um sistema de produção. Identifique algumas características particulares do nosso agente.

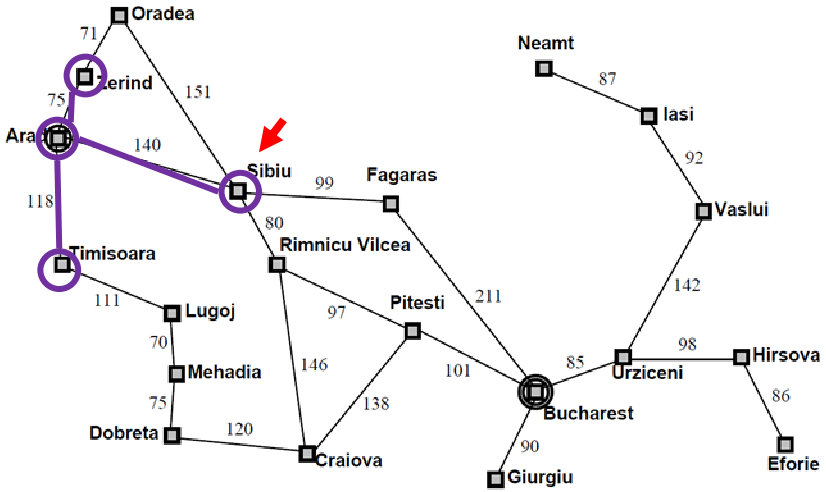
10. O agente Hercos vive num mundo 2D dividido em células rectangulares. O seu único propósito é passear no seu mundo evitando os obstáculos encontrados. A sua vizinhança é formada apenas pela célula que está à sua frente. As suas acções primárias são avançar para a célula para onde está virado (A), rodar 90º para a direita (D) ou para a esquerda (E). Tem ainda uma curta memória que lhe permite saber o que aconteceu no instante anterior. Pode actuar na memória alterando o seu conteúdo. As suas capacidades de percepção limitam-se a detectar a presença ou ausência de obstáculos. Modelize o comportamento do agente através de um sistema de produção de modo a que caminhe no seu mundo contornando os obstáculos. Em que medida a solução obtida evita que o Hercos não ande aos círculos.

11. Para cada um dos seguintes agentes, desenvolva uma descrição PEAS (Performance, *Environment*, Actuadores e Sensores):
    11.1. Robot jogador de futebol.
    11.2. Agente de compra de livros através da Internet.
    11.3. Veículo autónomo de exploração de Marte.
    11.4. Assistente para prova de teoremas matemáticos.

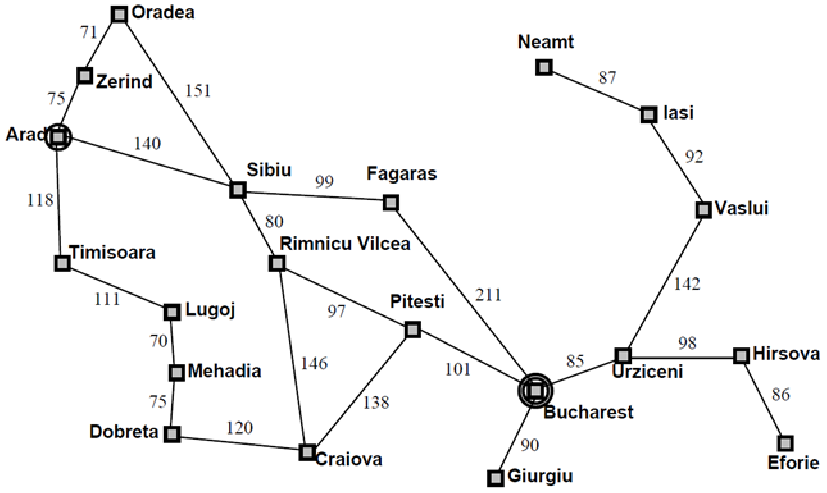
12. Implemente e teste um agente reflexo simples para o caso do mundo Vacuum do agente aspirador. (nota: considere o código aima-java disponível com o livro AIMA)
    12.1. Experimente alterar as medidas de desempenho do ambiente e faça testes com os vários agentes.

13. Altere o ambiente/mundo Vacuum de modo a que tenha quatro células em formato de grelha de tamanho 2x2. Cada célula pode estar limpa ou suja e deve considerar mais duas acções possíveis para o agente: ir para cima (ACTION_MOVE_TOP) e ir para baixo (ACTION_MOVE_BOTTOM).
    13.1. Implemente e teste um agente reflexo e um agente baseado em modelos usando o código aima-java.

1. Na figura seguinte (mapa da Roménia) está-se a utilizar a pesquisa em profundidade para resolver o problema de encontrar o caminho de Arad (estado inicial) até Bucharest (estado objectivo). Suponha que o nó a ser expandido neste momento é Sibui.
   1.1. Indique quais os nós que irão ser adicionados à estrutura fronteira no passo seguinte. Desenhe a árvore de expansão.
   1.2. Depois do passo anterior, indique quais os nós candidatos a serem expandidos no passo subsequente.
      1.2.1. Escolha um dos nós possíveis que respondem à alínea anterior e volte a expandir a árvore de dois modos distintos: usando uma função de pesquisa em árvore (não detecta expansão de estados múltiplas vezes); usando uma função de pesquisa em grafo (detecta e evita a expansão de estados múltiplas vezes).
   1.3. Continue a expandir a árvore de pesquisa para o problema em questão até chegar ao nó objectivo usando uma função de pesquisa em grafo.



2. Pretende-se agora utilizar a estratégia de pesquisa uniforme para encontrar o caminho de Arad (estado inicial) até Bucharest (estado objectivo) usando uma função de pesquisa em grafo. Faça a expansão da árvore de pesquisa até ao final.



3. O problema clássico das torres de Hanoi consiste em três hastes designadas por esquerda (E), média (M) e direita(D). Na haste E estão empilhados vários discos de diâmetro decrescente. Pretende-se mover os discos de E para D, de tal forma que, após qualquer movimento, nunca fique um disco de diâmetro maior sobre outro de diâmetro menor em nenhuma das hastes.
   3.1. Formule o problema anterior como um problema de pesquisa, definindo o estado inicial, os estados possíveis, os operadores, o teste objectivo e o custo da solução.

4. Suponha que um Pastor leva consigo um Lobo, um Carneiro e uma Couve. Chegado ao Rio, dispõe unicamente de um barco capaz de o transportar e transportar mais um item. O objectivo do pastor é passar para o outro lado levando os três itens mas se deixar sozinhos numa das margens, o lobo e o carneiro ou o carneiro e a couve vai ter problemas!
   4.1. Formule o problema anterior como um problema de pesquisa, definindo o estado inicial, os estados possíveis, os operadores, o teste objectivo e o custo da solução.
   4.2. Resolva o problema.
   4.3. Implemente o problema mencionado em software usando, por exemplo, a linguagem Java e o código de software disponibilizado pelo livro "Artificial Intelligence: A Modern Approach".

5. 3 missionários e 3 canibais estão numa das margens do rio com um barco que só leva 2 pessoas. Encontrar uma forma de levar os 6 para a outra margem do rio sem nunca deixar mais canibais do que missionários numa das margens durante o processo!
   5.1. Formule o problema anterior como um problema de pesquisa, definindo o estado inicial, os estados possíveis, os operadores, o teste objectivo e o custo da solução.
   5.2. Resolva o problema.
   5.3. Implemente o problema mencionado em software usando, por exemplo, a linguagem Java e o código de software disponibilizado pelo livro "Artificial Intelligence: A Modern Approach".

6. Considere uma matriz 3×3 onde serão colocados algarismos entre 1 e 9, de modo que em cada linha, coluna ou diagonal a soma seja sempre a mesma. Este problema designa-se por quadrado mágico de lado *n*, sendo com *n*=3 para o exercício proposto. A soma de cada linha, coluna ou diagonal é dada pela expressão $M = \frac{1}{2}n(n^2 + 1)$ que depende apenas do valor de *n* (ver: http://mathworld.wolfram.com/MagicSquare.html).
   6.1. Formule o problema anterior como um problema de pesquisa num espaço de estados.
   6.2. Discuta qual o método de pesquisa mais adequado para resolver este problema.

7. Considere o seguinte problema: dado o quadrado apresentado na figura, ligar o A com o A, o B com o B, o C com o C e o D com o D sem cruzar nenhuma linha!

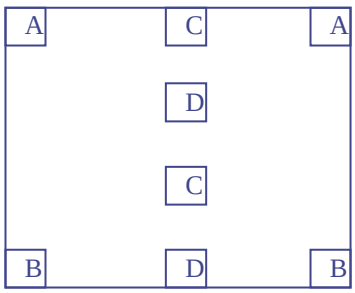


Ilustração 2.1 – problema do quadrado

7.1. Formule o problema anterior como um problema de pesquisa, definindo o estado inicial, os estados possíveis, os operadores, o teste objectivo e o custo da solução.
7.2. Resolva o problema.

8. Considere um espaço de estados onde o estado inicial é 1 e a função sucessora para o estado *n* produz dois estados, os números 2*n* e 2*n*+1.
   8.1. Desenhe a porção do espaço de estados para os estados 1 a 15.
   8.2. Suponha que o estado objectivo é 11. Liste a ordem pela qual os nós serão visitados para BFS, DFS com limite de pesquisa 3 e pesquisa de aprofundamento iterativo.
   8.3. Será, a pesquisa bidireccional, apropriada para este problema? Justifique.
   8.4. Qual é o factor de ramificação *b* (*branching factor*) em cada uma das direcções da pesquisa bidireccional neste problema?
   8.5. Em função da resposta nas alíneas anteriores, como é que poderia reformular o problema para ir do estado inicial a, virtualmente, qualquer estado quase sem usar pesquisa.

9. Dois baldes, de capacidades 4 litros e 3 litros, respectivamente, estão inicialmente vazios. Os baldes não possuem qualquer escala. As únicas operações que pode realizar são:
   - esvaziar um balde
   - encher (completamente) um balde
   - despejar um balde para o outro até que o segundo fique cheio
   - despejar um balde para o outro até que o primeiro fique vazio

   9.1. Quais as operações a efectuar de modo a que o primeiro balde contenha 2 litros ? Resolva este problema usando a estratégia "primeiro em profundidade".
   9.2. Quais as operações a efectuar de modo a que o primeiro balde contenha 2 litros ? Resolva este problema usando a estratégia "primeiro em largura".

10. Considere um labirinto rectangular genérico. Um agente é largado numa célula do labirinto e tem de descobrir a saída. Esse agente pode deslocar-se para cima, para baixo, para a esquerda e para a direita, caso não haja paredes.
    10.1. Formule o problema anterior como um problema de pesquisa, definindo o estado inicial, os estados possíveis, os operadores, o teste objectivo e o custo da solução.
    10.2. Implemente o problema mencionado em software utilizando o algoritmo A* para sua resolução.

11. Considere o jogo *Rush Hour* (http://en.wikipedia.org/wiki/Rush_Hour_(board_game)), um jogo solitário de tabuleiro do tipo *sliding block puzzle*.
    11.1. Formule o problema anterior como um problema de pesquisa, definindo o estado inicial, os estados possíveis, as ações, o teste objetivo e o custo da solução.
    11.2. Implemente o problema mencionado em software utilizando o algoritmo A* para sua resolução.

12. Considere o tabuleiro seguinte com 7 casas. Nele estão colocados 3 peões brancos e 3 peões pretos com a configuração inicial representada na figura.

Ilustração 2.2 – jogo das 7 casas

Pretende-se inverter a posição relativa dos peões para o que se podem usar os seguintes movimentos:

- Deslocar um peão para a casa adjacente caso esta não esteja ocupada. Esta operação tem um custo nulo.
- Deslocar um peão para a casa vazia saltando por cima de um ou (no máximo) dois peões. Operação com custo igual ao número de peões ultrapassados.

12.1. Discuta qual o método de pesquisa mais adequado para resolver este problema.
12.2. Defina funções **g** e **h** (função heurística) para o problema em questão.
12.3. Simule o algoritmo A* utilizando as funções definidas na alínea anterior.

13. No grafo seguinte, o nó A representa o estado inicial e os nós I e J os estados objectivo (finais). Os ramos indicam possíveis transições entre estados. A cada ramo está associado um custo real da transição de estado correspondente. Ao lado do nome de cada nó está indicado um valor estimado gerado por uma função heurística. Indique, justificando, qual seria a solução encontrada pelo algoritmo A*. Indique claramente a ordem pela qual os sucessivos estados são gerados.

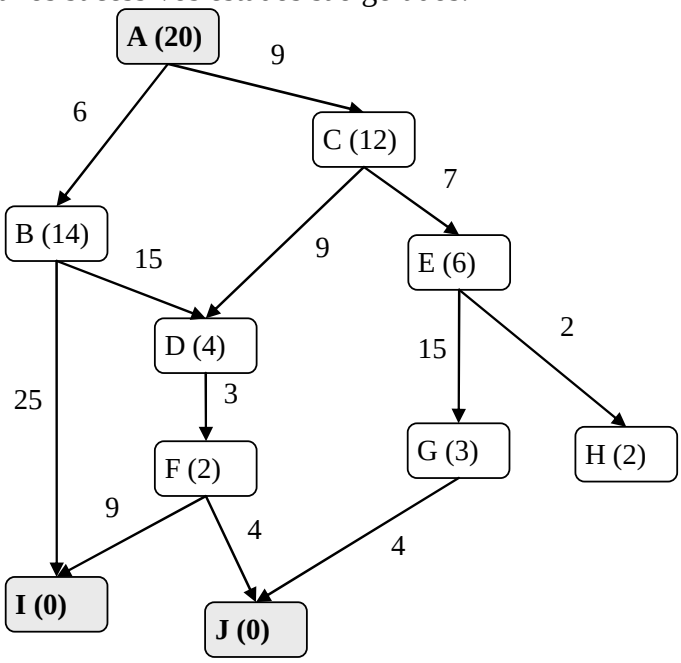


Ilustração 2.3 – grafo de pesquisa heuristica

14. Prove que quer a pesquisa de custo uniforme quer a pesquisa primeiro em largura quando aplicadas a problemas com custo por passo constante são óptimas quando usado o algoritmo de pesquisa em grafo. Dê um exemplo de espaço de estados com custo por passo variável em que a pesquisa de aprofundamento iterativo encontra uma solução subóptima.

15. Descreva um espaço de estados em que a pesquisa de aprofundamento iterativo tem um desempenho muito pior do que a pesquisa primeiro em profundidade.

16. Usando o problema do mundo do aspirador (Vacuum) como base:
    16.1. Formule-o como um problema de pesquisa, definindo o estado inicial, os estados possíveis, os operadores, o teste objectivo e o custo da solução.
    16.2. Resolva o problema e implemente-o usando a framework de software AIMA-JAVA.

16.3. Implemente variações do problema, como por exemplo, o cenário em que o mundo/ambiente Vacuum passa a ter quatro células em formato de grelha de tamanho 2x2.

17. Uma estratégia alternativa de pesquisa não informada, que poderá ser considerada, será a pesquisa em largura iterativa. Neste tipo de pesquisa, à semelhança da pesquisa de aprofundamento iterativo, também se estabelecem valores de corte no algoritmo, embora, desta feita, eles sejam aplicados para limitar, de um modo progressivamente crescente, o valor máximo do factor de ramificação em cada iteração do algoritmo. Analise este algoritmo sob o ponto de vista de completude e complexidade .

18. Implemente e teste, usando a framework de software AIMA-JAVA, a resolução do problema das N-Rainhas utilizando vários algoritmos de pesquisa local como *Hill Climbing*, *Simulated Annealing* e algoritmos genéticos.

19. No seguinte problema de planeamento de horários (*scheduling* ou *timetableing*), é necessário planear o horário de três máquinas M1, M2 e M3 que têm que realizar duas tarefas cada. Cada tarefa demora um certo tempo e ocupa um recurso. A informação está resumida na primeira tabela de baixo. Admita, como restrição adicional, que o mesmo recurso não pode ser usado em simultâneo por diferentes tarefas. Produza uma solução para o problema usando o algoritmo de *Simulated Annealing* e uma função heurística adequada ao problema. Um exemplo de solução é a apresentada na segunda tabela de baixo.

Tarefa	Máquina	Duração	Recurso
T1	M1	2	R2
T2	M1	3	R1
T3	M2	4	R1
T4	M2	3	R2
T5	M3	2	R1
T6	M3	3	R2

Tabela 2.1 – problema de geração de horários com 6 tarefas/3 máquinas/2 recursos

	1	2	3	4	5	6	7	8	9	*tempo*
M1	T1(R2)				T2(R1)					
M2	T3(R1)					T4(R2)				
M3			T6(R2)					T5(R1)		

Tabela 2.2 – uma solução possível para o problema anterior

1. Dê uma formulação precisa para o seguinte problema PSR:
   1.1. Problema da pavimentação do chão de uma sala de formato rectangular com pequenas lajes de formato rectangular: deverá seleccionar um conjunto de pequenas lajes rectangulares de modo que quando colocadas no chão não estejam sobrepostas.
   1.2. Problema de criação dos horários de uma escola: considere a existência de um número limitado de professores e de salas; uma lista de aulas que deverão funcionar e uma lista de slots temporais possíveis para essas aulas. Cada professor ficará com um conjunto de aulas que irá leccionar.

2. Dado o seguinte problema de criptoaritmética:

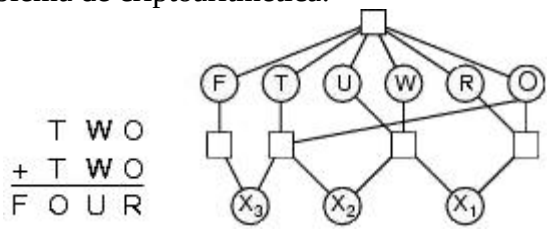


Ilustração 3.1 – problema de criptoaritmética

   2.1. Resolva-o, aplicando o algoritmo de pesquisa com retrocesso e os métodos independentes do domínio associados aos PSR.
   2.2. Implemente e resolva o mesmo problema usando, por exemplo, a *framework* de software AIMA-JAVA.

3. Use o algoritmo AC-3 para demonstrar que a consistência dos arcos é capaz de detectar inconsistência na atribuição parcial {WA = red, V = blue} no problema de coloração do mapa da Austrália.

4. Implemente e teste, usando a *framework* de software AIMA-JAVA, a resolução do problema das N-Rainhas utilizando pesquisa local e a heurística de conflitos mínimos.

1. Aplique o algoritmo Minimax e o algoritmo Minimax com cortes Alfa-Beta a uma árvore com factor de ramificação de 3 no nível superior e 3 no nível final e com os seguintes valores da função avaliação para a linha final:
   [5 3 12; 1 20 10; 2 -1 3]

2. Aplique o algoritmo Minimax a uma árvore com factor de ramificação de 4 no nível superior, 3 no segundo nível e 2 no nível final e com os seguintes valores da função avaliação para a linha final:
   [3 10 5 12 0 4 6 20 8 1 2 6 8 15 0 5 2 3 10 20 20 30 0 0]

3. Aplique o algoritmo Minimax com cortes Alfa-Beta a uma árvore com factor de ramificação de 4 no nível superior, 3 no segundo nível e 2 no nível final e com os seguintes valores da função avaliação para a linha final:
   [3 10 5 12 0 4 6 20 8 1 2 6 8 15 0 5 2 3 10 20 20 30 0 0]

4. Utilizando o algoritmo Minimax com cortes Alfa-Beta, implemente o jogo do galo (TicTacToe) para ser jogado entre um humano e o computador de um modo interactivo. Implemente o jogo usando, por exemplo, a *framework* de software AIMA-JAVA.

5. Utilizando o algoritmo Minimax com cortes Alfa-Beta, implemente variações (exemplo: Achi, Magic Square, ….) do jogo TicTacToe. Pode fazê-lo com recurso à *framework* de software AIMA-JAVA.

6. Prove a seguinte afirmação: "para toda a árvore de jogo, a utilidade obtida pelo MAX, usando decisões minimax, contra um adversário MIN subótimo nuca será menor que a utilidade obtida contra um MIN ótimo". Acha que existirão situações em que o MAX, usando uma estratégia subótima, poderá ter um desempenho ainda melhor do que se usasse a estratégia ótima, no confronto com um MIN subótimo? Se sim dê exemplos.

7. Considere a seguinte variante simplificada do jogo das 7 casas. Nesta variante apenas existem 4 casas e cada um dos jogadores, jogador 1 e jogador 2, apenas tem um peão que pretende colocar na outra extremidade do tabuleiro unidimensional. Quando isso acontece termina o jogo e ganha o jogador que alcançar esse objectivo. O jogo desenvolve-se com os jogadores a efectuarem jogadas alternadamente. Os jogadores podem efectuar os seguintes movimentos:
   - Deslocar um peão para a casa adjacente caso esta não esteja ocupada.
   - Deslocar um peão para a casa vazia saltando por cima do peão adversário.

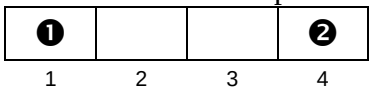


Ilustração 4.1 – jogo das 4 casas

   7.1. Desenhe a árvore do jogo marcando cada estado com o par de variáveis (s_1, s_2) que se referem, respectivamente, à casa onde está o peão do jogador 1 e à casa onde está o peão do jogador 2. Cada estado terminal deve estar assinalado com uma

caixa rectangular em seu redor. Assinale os estados repetidos em ciclo com uma caixa rectangular dupla em seu redor. Considere que quando o jogador 1 vence o valor é atribuído +1 e quando o jogador 2 vence o valor é atribuído -1.

1. Construa uma tabela de verdade com as variáveis lógicas P e Q e com as seguintes três frases como colunas adicionais: $P \wedge (P \Rightarrow Q)$; $\neg(\neg P \vee \neg Q)$; $(P \wedge (P \Rightarrow Q)) \Leftrightarrow (\neg(\neg P \vee \neg Q))$

2. Prove cada uma das asserções seguintes:
   2.1. A frase α é valida se e só se $True \models \alpha$
   2.2. Para qualquer frase α, $False \models \alpha$
   2.3. $\propto \vDash \beta$ se e só se a fórmula $(\propto \Rightarrow \beta)$ for válida
   2.4. $\propto \equiv \beta$ se e só se a fórmula $(\propto \Leftrightarrow \beta)$ for válida
   2.5. $\propto \vDash \beta$ se e só se a fórmula $(\propto \wedge \neg\beta)$ for inconsistente

3. Considere um vocabulário com apenas quatro proposições: A, B, C e D. Quantos modelos existem para cada uma das frases seguintes?
   3.1. $(A \wedge B) \vee (B \wedge C)$
   3.2. $A \vee B$
   3.3. $A \Leftrightarrow B \Leftrightarrow C$

4. Para cada uma das frases seguintes em lógica proposicional, indique se é válida, consistente ou inconsistente:
   4.1. $P \vee \neg P$
   4.2. $P \wedge \neg P$
   4.3. $P \vee Q \vee (P \Leftrightarrow Q)$
   4.4. $(P \Rightarrow Q) \vee (Q \Rightarrow P)$
   4.5. $\big((Comida \Rightarrow Festa) \vee (Bebida \Rightarrow Festa)\big) \Rightarrow ((Comida \wedge Bebida) \Rightarrow Festa)$

5. Representar as seguintes frases em lógica de primeira ordem:
   5.1. Todos os alunos desta disciplina estudaram Matemática.
   5.2. Alguns alunos desta disciplina estudaram Matemática.
   5.3. Nenhum aluno desta disciplina estudou Matemática.
   5.4. Alguns alunos desta disciplina não estudaram Matemática.
   5.5. Toda a gente tem um pai.
   5.6. O pai do meu irmão é meu pai também.
   5.7. Se duas pessoas afirmam serem ambas progenitores da mesma criança ou são de sexos diferentes ou uma delas está a mentir.

6. Mostre, utilizando tabelas de verdade, que as seguintes regras de inferência são consistentes:
   6.1. $\frac{P, P \Rightarrow Q}{Q}$
   6.2. $\frac{P \wedge Q}{P}$
   6.3. $\frac{P \Leftrightarrow Q, Q \Rightarrow R}{P \Rightarrow R}$

7. Representar as seguintes frases em lógica de primeira ordem:
   7.1. Só um aluno chumbou a História.

7.2. Nem todos os estudantes se inscreveram simultaneamente a Inteligência Artificial e Sistemas de Apoio à Decisão.
7.3. Só um aluno chumbou a História e a Biologia.
7.4. A melhor nota a História foi mais elevada do que a melhor nota a Biologia.
7.5. Todos os Portistas são espertos.
7.6. Todas as pessoas que não gostam de Portistas são burras.
7.7. Nenhuma pessoa gosta de um Portista burro.
7.8. Existe um Sportinguista que gosta de todos os Benfiquistas que não são espertos.
7.9. Existe um Barbeiro que barbeia toda a gente menos ele próprio.

1. Escreva o ficheiro *rel.pl* com o seguinte conteúdo:

```
parente(a,b).
parente(a,c).
parente(b,d).
parente(e,f).
homem(b).
irmão(X,Y) :- parente(P1,X), parente(P1,Y), homem(X), X\=Y.
```

   1.1. Carregue o programa anterior e experimente as seguintes perguntas:
   `irmão(a,b).  irmão(b,c).  irmão(c,b).`
   1.2. São os resultados os esperados? Aumente o número de factos e execute mais perguntas!
   1.3. Escreva regras para (e outras relações familiares):
   `irmã(X,Y).  pai(X,Y).  mãe(X,Y).  avó(X,Y).  avô(X,Y).`
   1.4. Execute perguntas para as novas regras inseridas.

2. Escreva, em Prolog, uma relação **ascendente(X, Y)**, para determinar se X é ascendente de Y (sucede se X for pai, mãe, avô, avó, bisavô, … de Y). Use **parente(X, Y)** como relação auxiliar.

3. Escreva, em Prolog, um predicado **nat(N)**, para listar todos os números naturais menores ou iguais ao número N.

4. Escreva no ficheiro *lista.pl* as definições das relações **member(X, L)** e **subset(L1, L2)** e teste-as com as seguintes perguntas:

```
member(a,[b,c,a,d]).
member(a,[b,c,f,d]).
member(20, [15,16, 17, 18, 19, 20]).
member(miguel, [ana, paulo, maria, joana]).
subset([a,b,c],[x,a,f,b,g,h,c]).
subset([1,2,100],[100,2,1,3]).
subset([1,2,100],[100,2]).
```

5. Escreva, em Prolog, uma relação **inversão(L1, L2)**, tal que a lista L2 represente a lista L1 pela ordem inversa. Por exemplo, **inversão([1,2,3,4], [4,3,2,1]).** sucede.

6. Escreva, em Prolog, uma relação **junção(L1, L2, L3)**, tal que a lista L3 representa a junção da lista L1 com a lista L2, isto é, a concatenação das listas o que permite a existência de elementos repetidos. Por exemplo, **junção([1,3,2], [3,4], [1,3,2,3,4]).** sucede.

7. Escreva, em Prolog, uma relação **intersecção(L1, L2, L3)**, tal que a lista L3 representa a intersecção da lista L1 com a lista L2. Por exemplo, **intersecção([1,2,3,4,6], [2,3,5], [2,3]).** sucede.

8. Escreva, em Prolog, uma relação **união(L1, L2, L3)**, tal que a lista L3 representa a operação de união da lista L1 com a lista L2. Por exemplo, **união([1,2,3,4,6], [2,3,5], [1,4,6,2,3,5]).** sucede.

9. Escreva, em Prolog, uma relação **rodar(L1, L2)**, tal que a lista L2 represente a lista L1 rodada de um para a esquerda.

```
?- rodar([1,2,3,4], L).
L = [2,3,4,1]
```

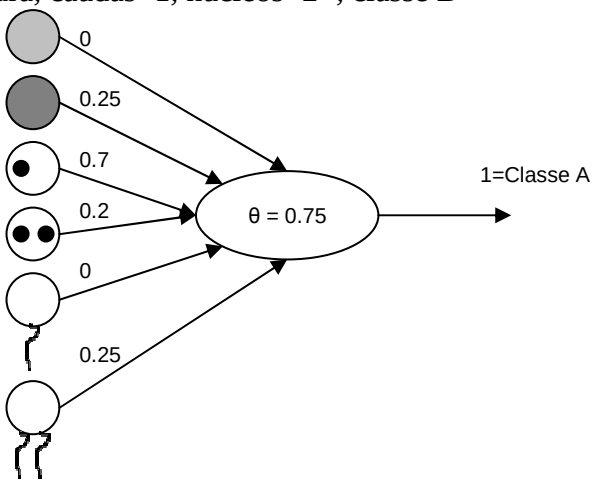
10. Escreva, em Prolog, uma relação **traduz(L1, L2)**, que traduz uma lista L1 de algarismos de 0 a 9 para uma lista L2 de palavras correspondentes. Use a relação **algarismo(0, zero)**, …, **algarismo(9, nove)** como relação auxiliar. Por exemplo, **traduz([2,3,5], [dois, três, cinco]).** sucede.

11. Escreva, em Prolog, uma relação **del(X, L1, L2)**, que elimina todos os elementos X duma lista L1 produzindo a lista L2 como resultado. Por exemplo, **del(2, [2,3,2,6,4,5], [3,6,4,5]).** sucede.

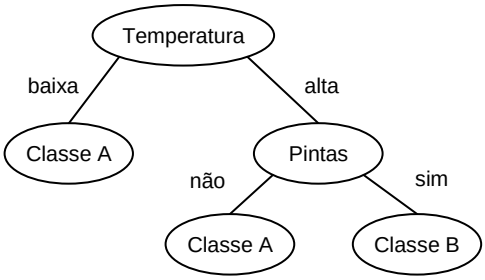
12. Escreva, em Prolog, uma relação **separa(L, LP, LN)**, tal que dada uma lista de números, L, a separa em duas listas: LP só com números positivos e LN só com números negativos. Por exemplo, **separa([1,-2,0,-4,6], [1,0,6], [-2,-4]).** sucede.

1. Dadas as hipóteses seguintes de descrição de conceitos:
   h_1 = <?,?,normal,?,quente,?>
   h_2 = <?,?,normal,?,?,?>
   h_3 = <?,?,normal,forte,?,?>
   h_4 = <?,?,normal,forte,quente,?>
   1.1. Indique as relações de ordem (se aplicável) existentes entre elas.

2. Verifique que a TLU da figura, usando uma função de activação do tipo degrau, classifica correctamente os cinco exemplos de treino seguintes:
   #1 = <<cor=escura, caudas=2, núcleos=2>, classe A>
   #2 = <<cor=clara, caudas=1, núcleos=1>, classe A>
   #3 = <<cor=escura, caudas=2, núcleos=1>, classe A>
   #4 = <<cor=clara, caudas=2, núcleos=2>, classe B>
   #5 = <<cor=escura, caudas=1, núcleos=2>, classe B>



3. Considere a árvore de decisão da figura seguinte produzida pelo algoritmo ID3:
   3.1. Defina o conceito correspondente à classe A.
   3.2. Indique, justificando, um conjunto de treino possível que levou à construção desta árvore.



4. Usando o algoritmo de árvores de decisão ID3, para problemas com duas classes {+, –}, escolha o atributo mais discriminante dos dois seguintes *a*0 e *a*1, para um conjunto de treino composto por 75 instâncias [30+, 45–]:
   - @atributo *a*0 com domínio {V [21+, 2–], F [9+, 43–]}

- @atributo *a*1 com domínio {V [14+, 25–], F [16+, 20–]}

5. Usando o algoritmo de árvores de decisão ID3, para problemas com duas classes {+, –}, construa a árvore de decisão considerando os atributos e exemplos de treino seguintes.

   Atributos:
   - @atributo *Tamanho* com domínio {grande, médio, pequeno}
   - @atributo *Peso* com domínio {pesado, normal, leve}
   - @atributo *Forma* com domínio {cubo, pirâmide, esfera, paralelepípedo}

   Exemplos de treino:
   - #1 = << médio, pesado, cubo>, +>
   - #2 = << pequeno, normal, pirâmide >, –>
   - #3 = << pequeno, normal, esfera >, +>
   - #4 = << grande, normal, pirâmide >, –>
   - #5 = << grande, leve, paralelepípedo >, +>
   - #6 = << grande, normal, paralelepípedo >, –>
   - #7 = << grande, leve, esfera >, +>

6. Considere a sequência seguinte de exemplos de treino (+ e -) que descreve o conceito "pares de pessoas que vivem na mesma casa".
   - #1 = (<masculino, castanho, alto, americana>, <feminino, preto, baixo, americana>), +
   - #2 = (<masculino, castanho, baixo, francesa>, <feminino, preto, baixo, americana>), +
   - #3 = (<feminino, castanho, alto, alemã>, <feminino, preto, baixo, indiana>), -
   - #4 = (<masculino, castanho, alto, irlandesa>, <feminino, castanho, baixo, irlandesa>), +

   Os atributos considerados e os respectivos valores do domínio são:
   sexo = {masculino, feminino}
   cor_cabelo = {preto, castanho, louro}
   altura = {alto, normal, baixo}
   nacionalidade = {americana, francesa, alemã, irlandesa, indiana, japonesa, portuguesa}

   6.1. Se cada exemplo de treino descrever um par ordenado de pessoas, simule o funcionamento do algoritmo de Eliminação de Candidatos para a sequência apresentada.

7. Suponha que quer usar uma rede neuronal para aprender uma função conhecidos os seguintes exemplos:

x_1	x_2	x_3	o
0	1	1	1
1	1	0	0
1	1	1	1
1	0	1	0

   Admita que usa apenas uma TLU com função de activação degrau binária e aprendizagem usando a regra delta. Admita também que os pesos e o limiar θ são inicializados a zero e o ritmo de aprendizagem vale $\eta = 0.25$.
   Nestas condições determine a rede resultante após ter processado uma vez cada um dos quatro exemplos.

8. Considere a seguinte função booleana conhecida por **todos**:

x_1	x_2	o
0	0	1
0	1	1
1	0	1
1	1	1

   Implemente esta função por meio de TLU, definindo valores aleatórios para os pesos e limiar e simulando a regra delta para uma etapa de treino envolvendo as quatro possibilidades de entrada.

9. Leia documentação introdutória ao software de *data mining* WEKA, incluindo a descrição sobre o formato *Attribute-Relation File Format* (ARFF), acessível através do site de documentação no url: http://www.cs.waikato.ac.nz/~ml/weka/

10. Utilizando a aplicação *Weka Explorer* abra o *dataset* "weather.arff"
    10.1. Faça uma análise preliminar ao *dataset* no painel de preprocessamento.
    10.2. Experimente aplicar diversas filtragens ao *dataset*, grave-o em ficheiros com diferentes nomes e verifique quais as alterações introduzidas usando um editor de texto.
    10.3. Aplique o algoritmo de aprendizagem de árvores de decisão J48 (uma versão melhorada do C4.5) ao *dataset*.

11. Utilizando a aplicação *Weka Explorer* abra o dataset "zoo.arff"
    11.1. Faça uma análise preliminar ao dataset no painel de preprocessamento como por exemplo, quantos animais contém.
    11.2. Aplique o algoritmo de aprendizagem de árvores de decisão J48 ao *dataset*.
    11.3. Experimente aplicar outros classificadores ao dataset, nomeadamente:
        Decision stump - weka.classifiers.DecisionStump
        OneR - weka.classifiers.OneR
        Decision table - weka.classifiers.DecisionTable -R
        PART - weka.classifiers.j48.PART

12. Utilizando a aplicação *Weka Explorer* abra o dataset "bolts.arff"
    12.1. Experimente aplicar os seguintes classificadores ao dataset (não use o atributo TIME)
        Decision stump - weka.classifiers.DecisionStump
        Decision table - weka.classifiers.DecisionTable -R
        Linear regression - weka.classifiers.LinearRegression