UNIVERSIDADE ABERTA

# e-fólio B

Introdução a C++ Inteligência Artificial

## Caixeiro-Viajante[1]

Implemente um algoritmo para um caixeiro-viajante que circula entre vários mercados ao longo de períodos de 20 semanas, um mercado por semana, e pretende otimizar a ordem de passagem pelos diferentes mercados. Num mercado o caixeiro-viajante compra produtos que vende no segundo mercado seguinte (os produtos comprados na semana 3 são vendidos na semana 5), realizando no entanto maiores ganhos entre uns mercados que em outros, existindo até pares de mercados em que por não conseguir vender artigos nenhuns, pode perder dinheiro. A capacidade de transporte é de 20 unidades no total, que é sempre utilizada mesmo que não compense monetariamente, sendo comprado em cada mercado 10 unidades. Deve comercializar os 5 produtos pelo menos uma vez, e passar por cada um dos 20 mercados uma só vez.

Em cada mercado existem as seguintes oportunidades de compra/venda de cada um dos 5 produtos (colocar de forma estática no código):

```
int compra[MAX_MERCADOS][MAX_PRODUTOS] =
{
  { 10, 7, 3, 6, 8 },
  { 5, 7, 5, 6, 7 },
  { 8, 6, 4, 4, 10 },
  { 6, 6, 5, 5, 9 },
  { 7, 7, 4, 2, 10 },
  { 9, 6, 4, 8, 8 },
  { 5, 7, 3, 6, 10 },
  { 10, 7, 4, 6, 9 },
  { 8, 5, 4, 7, 7 },
  { 8, 6, 5, 9, 10 },
  { 9, 8, 4, 5, 6 },
  { 4, 6, 5, 6, 7 },
  { 9, 5, 4, 6, 7 },
  { 4, 6, 3, 5, 9 },
  { 7, 8, 6, 7, 8 },
  { 9, 7, 7, 5, 8 },
  { 8, 6, 4, 8, 10 },
  { 8, 7, 5, 10, 10 },
  { 8, 8, 5, 7, 9 },
  { 7, 7, 6, 7, 8 }
};
```

```
int venda[MAX_MERCADOS][MAX_PRODUTOS] =
{
  { 5, 2, 1, 3, 4 },
  { 2, 4, 3, 4, 3 },
  { 6, 3, 2, 3, 4 },
  { 2, 5, 3, 3, 3 },
  { 6, 4, 3, 1, 3 },
  { 6, 5, 3, 4, 4 },
  { 4, 6, 2, 2, 6 },
  { 5, 3, 2, 6, 5 },
  { 7, 2, 3, 3, 5 },
  { 6, 4, 4, 4, 6 },
  { 6, 2, 1, 3, 4 },
  { 4, 4, 2, 2, 3 },
  { 7, 3, 2, 3, 5 },
  { 3, 5, 3, 3, 6 },
  { 4, 5, 3, 5, 5 },
  { 7, 6, 4, 3, 4 },
  { 6, 3, 2, 5, 6 },
  { 6, 5, 3, 4, 5 },
  { 7, 5, 3, 5, 4 },
  { 6, 6, 4, 6, 7 }
};
```

Pretende-se uma ordem de mercados e lista de quantidades de produtos a adequirir, que maximize o total de retorno de uma passagem pelos 20 mercados, assumindo o preço de compra e de venda indicados nas matrizes anteriores.

Um exemplo de uma solução é a seguinte ordem de passagem pelos mercados (1º mercado é o 0):

Mercados: 19; 17; 18; 5; 1; 4; 15; 14; 0; 2; 13; 7; 12; 6; 10; 16; 11; 8; 3; 9.

[1] O problema do caixeiro viajante aqui relatado neste e-fólio, é distinto do problema do caixeiro viajante clássico

Para além da ordem dos mercados, há que saber que produto e em que quantidades adquirir. Uma possibilidade é a seguinte (1º produto é o 0, sempre em quantidades máximas de 10 unidades):

> Produtos: 0; 0; 2; 2; 0; 3; 3; 0; 2; 3; 0; 1; 0; 0; 2; 0; 1; 2; 3; 1 – 9 unidades, 4 – 1 unidade.

Apenas no último mercado há dois produtos, o produto 1 com 9 unidades, e o produto 4 com 1 unidade. Este produto teve de ser adquirido mesmo não compensando, de modo a comercializar o produto 4 que não foi adquirido em qualquer outro mercado. O custo total desta solução é 54 (portanto não compensaria dado que o custo é positivo), existindo as seguintes diferenças entre preço de compra e venda (compra-venda no 2º mercado após a aquisição):

> (Compra-Venda): (7 - 7); (8 - 6); (5 - 3); (4 - 3); (5 - 7); (2 - 5); (5 - 3); (7 - 6); (3 - 3); (4 - 6); (4 - 7); (7 - 6); (9 - 6); (5 - 6); (4 - 2); (8 - 7); (6 - 5); (4 - 4); (5 - 6); 9*(6 - 5) + 1*(10 - 5)

A compra nos dois primeiros mercados, do produto 0, é feita a preços 7 e 8 respectivamente, uma vez que o mercado inicial é o 19 (última linha), e 17 (penúltima linha), com valores 7 e 8. A venda é feita a preços 7 e 6, nos mercados 18 e 5 (dois mercados após serem adquiridos), daí os valores 7 e 6. A soma de todas as diferenças multiplicado por 10 unidades resulta no custo da solução de 54. Note-se ainda que as unidades compradas nos dois últimos mercados, neste caso 3 e 9, são vendidas nos dois primeiros mercados, 19 e 17 respectivamente.

Pretende-se a solução de menor custo, idealmente negativo, de modo a permitir viabilizar a atividade comercial.

Deve entregar:

- Relatório;
- Código fonte dos algoritmos implementados.

Critérios de correção (4 valores):

- **Análise do problema** (2 valores): Referência a aspectos importantes do problema no relatório, revelando independentemente de os implementar ou não, que tinha consciência dos mesmos.
- **Identificação de algoritmos** (1 valor): Identificação clara dos algoritmos que implementou de acordo com a nomenclatura do livro e da UC, juntamente com as configurações utilizadas, ou no caso de utilização de um algoritmo distinto, deve descrevê-lo. A utilização de outro nome para os mesmos algoritmos é possível, desde que indique a qual correspondente. A penalização para a não identificação corresponde a 0,5 valores.
- **Resultados** (1 valor): Para a qualidade da solução que conseguir encontrar (em menos de 1 minuto de CPU, modo de *release*), indicar o algoritmo/configuração, valor da solução, número de expansões, gerações e avaliações.