

# Inteligência Artificial

João Vale, Luís Barros, Flávio Sousa, André Carvalho

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal

e-mail: {a100715, a100818, a100697, a100693}@alunos.uminho.pt

## 1 Introdução

No âmbito da Unidade Curricular de Inteligência Artificial foi-nos apresentado um trabalho prático que visa a implementação de diversos algoritmos de procura, considerando a perspetiva da sustentabilidade. Assim, o objetivo principal centra-se na otimização, de forma eficiente, dos melhores percursos para uma empresa de distribuição de encomendas.

Apesar das várias empresas de distribuição que conhecemos usufruírem de algoritmos bem desenvolvidos para este objetivo, nada tira a complexidade do sistema a desenvolver. Como tal, ao longo deste relatório pretendemos apresentar todas as adversidades e dilemas que enfrentámos durante a implementação do nosso serviço.

## 2 Formulação do Problema

Com o objetivo de garantir o sucesso na futura implementação, acreditamos ser uma boa ideia começar por formular o problema base, iniciando com a sua representação computacional. Após uma longa análise, decidimos ser mais eficiente a elaboração de um grafo. Para esse efeito, recorremos à biblioteca *networkx* para facilitar a sua criação e a respetiva representação visual.

Para o nosso *grafo*, decidimos que cada *nodo* representaria uma cidade e, cada aresta, o respetivo peso calculado pela distância dos *nodos* (*cidades*), onde as suas posições *x* e *y* seriam estáticas e previamente definidas com a biblioteca *networkx*.

$$\text{Peso da aresta} = \sqrt{(x_2^2 - x_1^2) + (y_2^2 - y_1^2)} * 3$$

Foi criada, então, a função *Peso da Aresta* que calcula o pesos entre dois *nodos*, permitindo, ao utilizador, a alteração do mapa, por outras palavras, permite que novos caminhos entre freguesias sejam criados ou eliminados para modificação de rotas.

Figura 1: Mapa no seu estado inicial.

A escolha da divisão das freguesias por zonas será ainda detalhadamente explicada ao longo do relatório.

### 3 Tipos de Dados

Antes de iniciar a implementação dos algoritmos, decidimos estabelecer uma representação formal para cada tipo de dado, bem como desenvolver uma estrutura de base de dados correspondente.

#### 3.1 Estafeta

O estafeta seria simplesmente definido pelo seu *id*, *nome*, *avaliação* e *encomendas*, onde as encomendas entregues são uma lista, inicialmente vazia, preenchida por cada encomenda que o mesmo entrega e a avaliação é atualizada também aquando a entrega de cada encomenda.

#### 3.2 Veículo

Para o veículo, definimos que o mesmo iria ser composto pelo *peso máximo*, *volume máximo*, *velocidade média*, *perda de velocidade* e *preço*. Aqui, cada um dos três tipos de veículo teria todos os seus valores estipulados previamente, tornando-se portanto muito simples a implementação de funções para calcular, por exemplo, a alteração da velocidade média com base no peso que o mesmo leva, bem como, alterar o preço de uma encomenda com base no veículo que a transporta.

#### 3.3 Encomenda

No caso da encomenda, a mesma é caracterizada pelo seu *id*, *morada*, *data limite*, *data de entrega*, *peso*, *volume*, *preço*, *avaliação* e *status*. Desta forma, a data de entrega e a avaliação são apenas atualizados aquando da respetiva entrega; já o status apenas indica se a mesma já foi entregue ou não.

#### 3.4 Rota

Acreditando que a rota seja o dado mais importante para o trabalho, passamos para a abordagem da mesma. Sendo representada por *id*, *estafeta*, *veículo*, *distância*, *peso*, *volume*, *zona* e *encomendas*. Desta forma, o volume e o peso (inicializados a 0), são automaticamente calculados com base nas encomendas atribuídas a essa rota, podendo posteriormente ser definido também o melhor veículo para esses atributos. Como seria de esperar, a distância dessa rota é apenas calculada posteriormente ao uso dos algoritmos.

Quando falamos de zonas, é importante referir que as mesmas são úteis para obter uma melhor sustentabilidade, direcionando para uma rota apenas as encomendas pertencentes à mesma zona. Desta forma, permitimos que a área de cada rota seja mais pequena e mais sustentável.

## 4 Geração de Dados

De modo a disponibilizar dados para a execução do programa achamos, por bem, que a criação de um simples programa de geração dos mesmos poderia ser útil para efeito de testes. Assim, criamos dois ficheiros de texto - um para as encomendas e outro para os estafetas, que seriam povoados de forma aleatória. Posteriormente, preenchemos a classe denominada de *Data*, sendo esta, composta pelo nosso *grafo* com o mapa, uma lista de *estafetas*, um dicionário de listas *encomendas* - onde cada *key* representa uma zona - e, por fim, uma lista de *rotas* povoada também de forma aleatória por um conjunto de uma a quatro encomendas.

## 5 Algoritmos

De forma a obter o melhor trajeto possível que passasse por todas as freguesias com encomenda atribuída, decidimos usar diferentes algoritmos de procura:

### 5.1 BFS

Para além de procurar um dos trajeto possíveis, este algoritmo apresenta outra funcionalidade bastante relevante para o nosso projeto. Aquando o *upload* dos dados, o mesmo é utilizado para percorrer todas as freguesias e garantir que todas tenham pelo menos um caminho possível, impossibilitando assim a existência de encomendas impossíveis de entregar.

### 5.2 DFS

Tendo sido também implementado este algoritmo denotamos que, da mesma forma que o BFS, nenhum dos dois nos trazia resultados muito otimistas, ficando assim muito à quem daquilo que seria o trajeto mais sustentável a seguir.

### 5.3 AStar e Dijkstra

Para obter sempre o melhor resultado possível, acreditamos que o nosso algoritmo *dijkstra*, que atualiza de forma adequada as *heurísticas*, em conjunto com o algoritmo *AStar*, para definir o trajeto mais curto, seja a melhor opção a escolher. Deste modo, utilizamos como pré-definição que o trajeto seria sempre escolhido por este conjunto de algoritmos.

### 5.4 Traveling Salesman

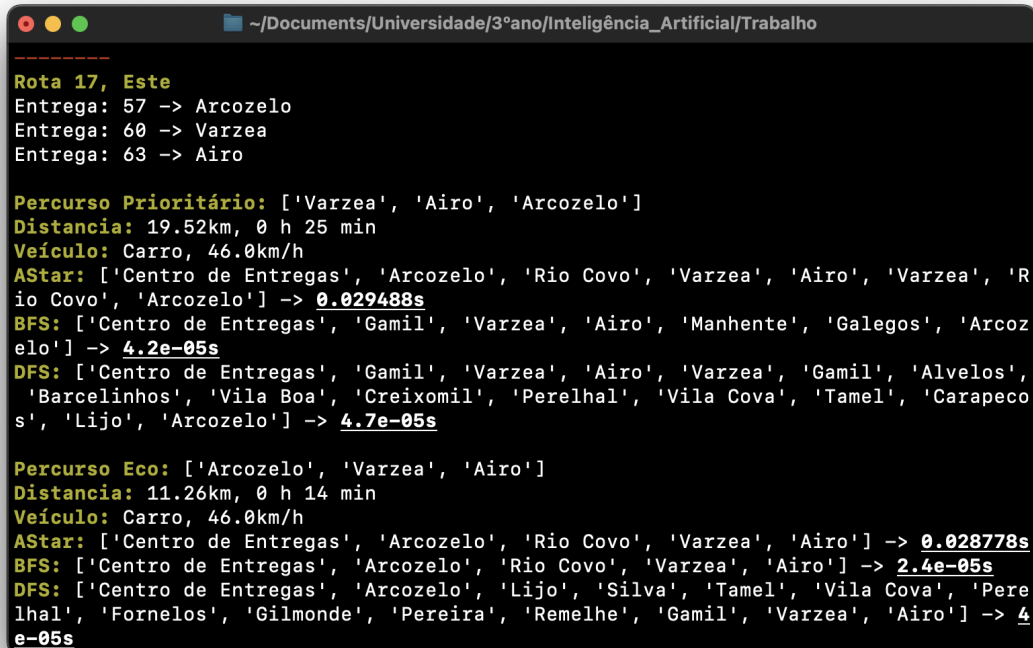
Para além de ser importante garantir o menor trajeto entre duas freguesias para uma melhor sustentabilidade, é também muito conveniente procurar a melhor ordem de passagem entre freguesias, de modo a garantir a

sustentabilidade máxima. Por outras palavras, se um estafeta precisa de fazer uma entrega em x, y e z, o mesmo deve procurar a melhor ordem de passagem para garantir que faz o melhor trajeto a passar por todos os pontos e não só o melhor trajeto de x a y ou de x a z, por exemplo.

Assim, após uma pequena procura na *web*, encontramos o problema do caixeiro viajante, *traveling salesman*, e tentámos uma adaptação mais simplista. Sucintamente, tendo uma lista de freguesias pela qual o estafeta precisa passar, o nosso algoritmo começa por procurar a que se encontra mais próxima ao seu ponto de partida tornando-se, esse o novo ponto de partida e repetindo, assim, o processo. Tendo, finalmente, o melhor conjunto de passagens, o resto do trabalho fica a cargo dos nossos algoritmos mencionados anteriormente.

Consideramos ainda relevante mencionar a implementação de outra alternativa ao *traveling salesman*. Semelhante a este, procuramos organizar a ordem de passagem das freguesias, não em função da distância para o melhor percurso, mas sim em função da ordem de prioridade das encomendas, ou seja, em função das datas limite de entregas.

Segue-se, então, um exemplo da aplicação de todos estes algoritmos:



```
-----
Rota 17, Este
Entrega: 57 -> Arcozelo
Entrega: 60 -> Varzea
Entrega: 63 -> Airo

Percurso Prioritário: ['Varzea', 'Airo', 'Arcozelo']
Distancia: 19.52km, 0 h 25 min
Veículo: Carro, 46.0km/h
AStar: ['Centro de Entregas', 'Arcozelo', 'Rio Covo', 'Varzea', 'Airo', 'Varzea', 'Rio Covo', 'Arcozelo'] -> 0.029488s
BFS: ['Centro de Entregas', 'Gamil', 'Varzea', 'Airo', 'Manhente', 'Galegos', 'Arcozelo'] -> 4.2e-05s
DFS: ['Centro de Entregas', 'Gamil', 'Varzea', 'Airo', 'Varzea', 'Gamil', 'Alvelos', 'Barcelinhos', 'Vila Boa', 'Creixomil', 'Perelhal', 'Vila Cova', 'Tamel', 'Carapeços', 'Lijo', 'Arcozelo'] -> 4.7e-05s

Percurso Eco: ['Arcozelo', 'Varzea', 'Airo']
Distancia: 11.26km, 0 h 14 min
Veículo: Carro, 46.0km/h
AStar: ['Centro de Entregas', 'Arcozelo', 'Rio Covo', 'Varzea', 'Airo'] -> 0.028778s
BFS: ['Centro de Entregas', 'Arcozelo', 'Rio Covo', 'Varzea', 'Airo'] -> 2.4e-05s
DFS: ['Centro de Entregas', 'Arcozelo', 'Lijo', 'Silva', 'Tamel', 'Vila Cova', 'Perelhal', 'Fornelos', 'Gilmonde', 'Pereira', 'Remelhe', 'Gamil', 'Varzea', 'Airo'] -> 4e-05s
```

Figura 2: Demonstração dos algoritmos.

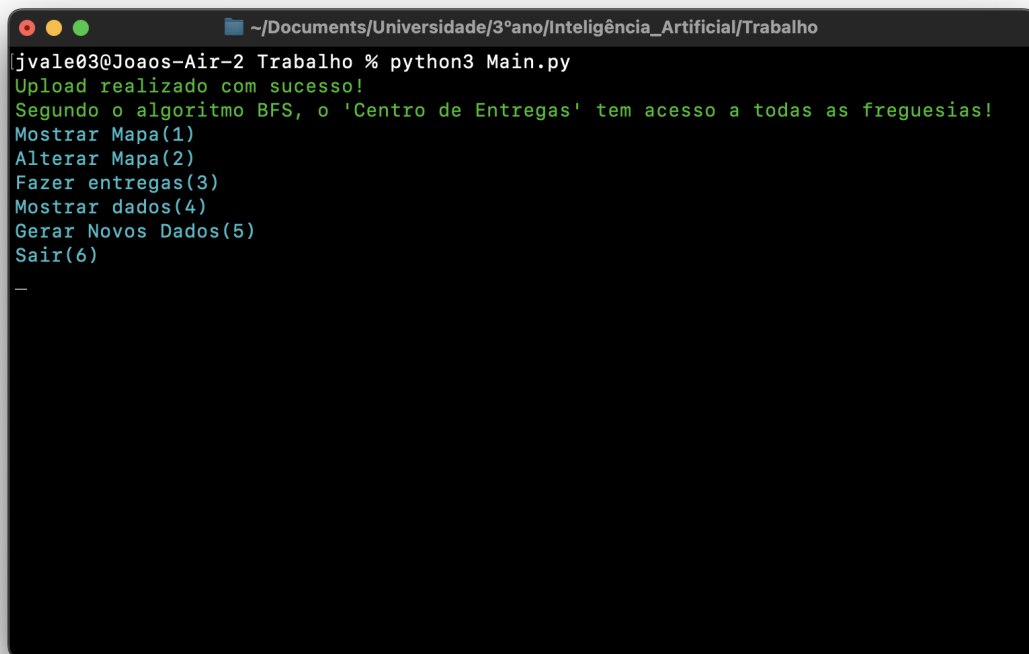
Desta forma, conseguimos acompanhar o uso de todos os diferentes algoritmos utilizados, assim como a vantagem de obter dois tipos de percurso, o percurso ecológico que obtém o menor trajeto possível, 11.26km e o percurso prioritário onde prevalecem as datas limite, 19.52km.

## 6 Exemplos

Por último, mas não menos importante, segue-se uma demonstração detalhada de cada uma das funcionalidades da nossa aplicação:

### 6.1 Menu Principal

Quanto a este menu, não há considerações particularmente significativas a acrescentar, sendo a sua finalidade autoevidente.

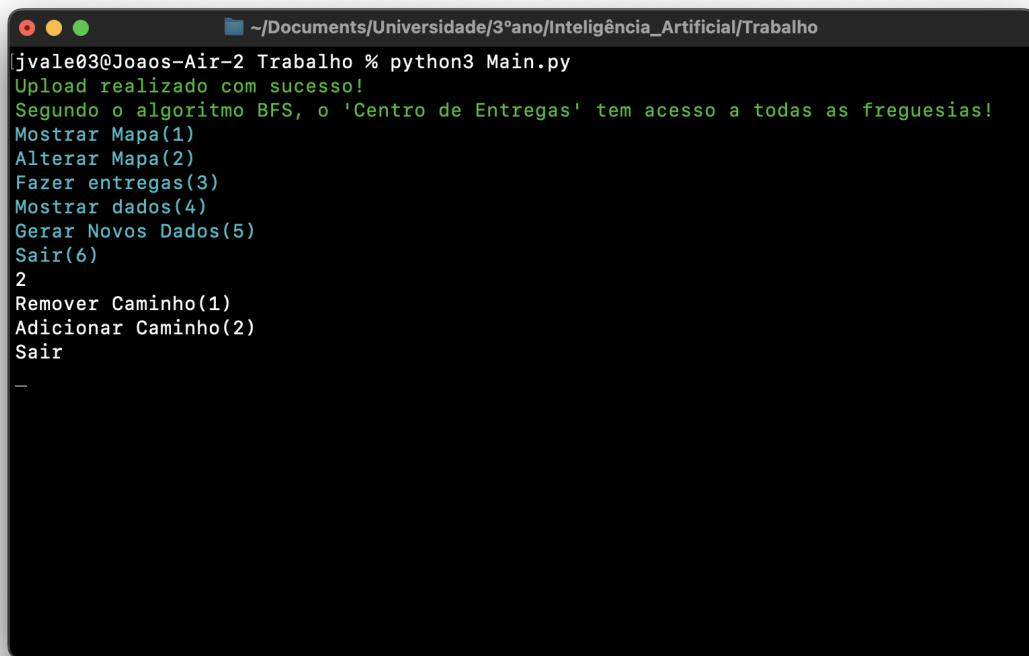
A terminal window with a dark background and light-colored text. The window title bar shows the path ~/Documents/Universidade/3ºano/Inteligência\_Artificial/Trabalho. The terminal output shows the execution of a Python script named Main.py, which displays a menu of options for the user to interact with. The options are numbered 1 through 6, corresponding to different functions of the application. The user has entered a dash '-' at the prompt, indicating they have not yet selected an option.

```
jvale03@Joaos-Air-2 Trabalho % python3 Main.py
Upload realizado com sucesso!
Segundo o algoritmo BFS, o 'Centro de Entregas' tem acesso a todas as freguesias!
Mostrar Mapa(1)
Alterar Mapa(2)
Fazer entregas(3)
Mostrar dados(4)
Gerar Novos Dados(5)
Sair(6)
-
```

Figura 3: Menu principal.

## 6.2 Alterar Mapa

Esta funcionalidade permite-nos escolher duas opções: *adicionar* e *remover* um caminho. Desta forma, ambas as opções solicitam duas freguesias à qual desejamos, respetivamente, adicionar ou remover um caminho, calculando o peso do mesmo de forma automática.



```
jvale03@Joaos-Air-2 Trabalho % python3 Main.py
Upload realizado com sucesso!
Segundo o algoritmo BFS, o 'Centro de Entregas' tem acesso a todas as freguesias!
Mostrar Mapa(1)
Alterar Mapa(2)
Fazer entregas(3)
Mostrar dados(4)
Gerar Novos Dados(5)
Sair(6)
2
Remover Caminho(1)
Adicionar Caminho(2)
Sair
—
```

Figura 4: Alterar Mapa.

## 6.3 Fazer Entregas

Referindo agora o momento mais importante da aplicação, o detalhe da demonstração do mesmo será crucial.

Como podemos observar, é nos permitido que vários estafetas saiam do seu posto para realizar as suas respetivas rotas, no entanto, nunca será permitido que o mesmo estafeta realize duas rotas simultaneamente.

```
~/Documents/Universidade/3ºano/Inteligência_Artificial/Trabalho
jvale030Joaos-Air-2 Trabalho % python3 Main.py
Upload realizado com sucesso!
Segundo o algoritmo BFS, o 'Centro de Entregas' tem acesso a todas as freguesias!
Mostrar Mapa(1)
Alterar Mapa(2)
Fazer entregas(3)
Mostrar dados(4)
Gerar Novos Dados(5)
Sair(6)
3
Insira o id de um estafeta: 35
Insira o id de um estafeta: 39
Insira o id de um estafeta: 35
Estafeta inválido!
Insira o id de um estafeta: 43
Insira o id de um estafeta: _
```

Figura 5: Entrega.

```
~/Documents/Universidade/3ºano/Inteligência_Artificial/Trabalho
Rota 39, Norte
Entrega: 73 -> Silva
Entrega: 74 -> Lijo
Entrega: 76 -> Alvito
Entrega: 77 -> Roriz

Percurso Prioritário: ['Lijo', 'Roriz', 'Silva', 'Alvito']
Distancia: 20.13km, 0 h 27 min
Veículo: Carro, 43.78km/h
AStar: ['Centro de Entregas', 'Lijo', 'Roriz', 'Lijo', 'Silva', 'Lijo', 'Alvito'] -> 0.034411s
BFS: ['Centro de Entregas', 'Lijo', 'Roriz', 'Lijo', 'Silva', 'Carapecos', 'Alvito'] -> 3.5e-05s
DFS: ['Centro de Entregas', 'Lijo', 'Roriz', 'Galegos', 'Manhente', 'Airo', 'Varzea', 'Gamil', 'Alvelos', 'Barcelinhos', 'Vila Boa', 'Creixomil', 'Perelhal', 'Vila Cova', 'Tamel', 'Silva', 'Tamel', 'Vila Cova', 'Perelhal', 'Fornelos', 'Gilmonde', 'Pereira', 'Remelhe', 'Gamil', 'Rio Covo', 'Arcozelo', 'Galegos', 'Roriz', 'Alvito'] -> 9.5e-05s

Percurso Eco: ['Silva', 'Lijo', 'Alvito', 'Roriz']
Distancia: 11.82km, 0 h 16 min
Veículo: Carro, 43.78km/h
AStar: ['Centro de Entregas', 'Silva', 'Lijo', 'Alvito', 'Roriz'] -> 0.034717s
BFS: ['Centro de Entregas', 'Silva', 'Lijo', 'Alvito', 'Roriz'] -> 2.3e-05s
DFS: ['Centro de Entregas', 'Silva', 'Lijo', 'Alvito', 'Roriz'] -> 1.8e-05s
```

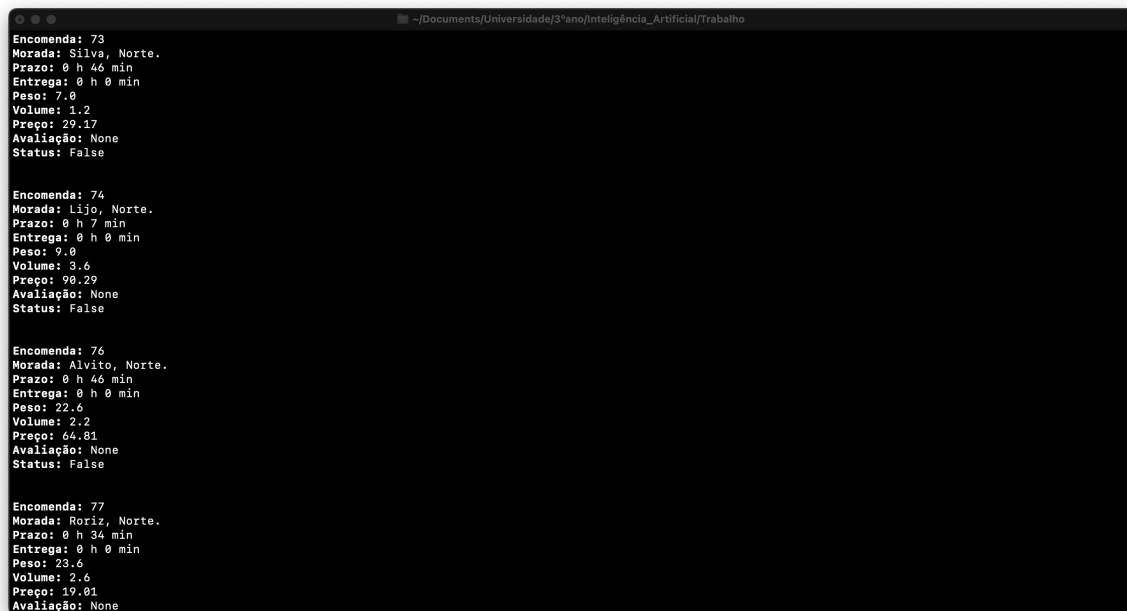
Figura 6: Rota 39.



Para acompanhar melhor o exemplo desta rota, precisamos também analisar as 4 encomendas que o mesmo está encarregar, 73,74,76 e 77.

Desta forma, conseguimos perceber que o *percurso prioritário* está devidamente organizado com base no prazo e que o *percurso eco* ordena devidamente com base na distância da rota. É portanto ainda possível observar que a escolha do *veículo* é também adequada e que a sua velocidade média desce conforme o previsto.

Infelizmente, para este exemplo, no percurso Eco observamos que, coincidentemente, todos os algoritmos obtêm um resultado ótimo, no entanto, positivamente para nós, no percurso prioritário, conseguimos observar que para aquela ordem de passagens, a nossa combinação do *AStar* com o *Dijkstra* é sem dúvida a mais otimista.



```
Encomenda: 73
Morada: Silva, Norte.
Prazo: 0 h 46 min
Entrega: 0 h 0 min
Peso: 7.0
Volume: 1.2
Preço: 29.17
Avaliação: None
Status: False

Encomenda: 74
Morada: Lijo, Norte.
Prazo: 0 h 7 min
Entrega: 0 h 0 min
Peso: 9.0
Volume: 3.6
Preço: 90.29
Avaliação: None
Status: False

Encomenda: 76
Morada: Alvito, Norte.
Prazo: 0 h 46 min
Entrega: 0 h 0 min
Peso: 22.6
Volume: 2.2
Preço: 64.81
Avaliação: None
Status: False

Encomenda: 77
Morada: Roriz, Norte.
Prazo: 0 h 34 min
Entrega: 0 h 0 min
Peso: 23.6
Volume: 2.6
Preço: 19.01
Avaliação: None
```

Figura 7: Encomendas no seu estado inicial.

**Funcionalidade Extra** Nesta fase das entregas é nos ainda permitido realizar cada um destes percursos passo a passo e com tempo entre paragens. Deste modo, torna-se possível cortar ou adicionar estradas a meio do percurso, levando o estafeta a recalcular o seu percurso e procurar um novo caminho mais próximo.

Para permitir estas funcionalidades foi necessário recorrer ao uso de *threads*, alocados individualmente a cada estafeta.

```
~/Documents/Universidade/3ºano/Inteligência_Artificial/Trabalho
BFS: ['Centro de Entregas', 'Silva', 'Lijo', 'Alvito', 'Roriz'] -> 2.1e-05s
DFS: ['Centro de Entregas', 'Silva', 'Lijo', 'Alvito', 'Roriz'] -> 1.8e-05s

Realizar rota prioritária(1)
Realizar rota eco(2)
Sair
2
Remover Caminho(1)
Adicionar Caminho(2)
Sair

Driver 39: ['Centro de Entregas', 'Silva'] -> A tempo
1
Inserir freguesia: Lijo
Inserir freguesia: Alvito
Remover Caminho(1)
Adicionar Caminho(2)
Sair

Driver 39: ['Silva', 'Lijo'] -> Fora de tempo

Driver 39: ['Lijo', 'Roriz', 'Alvito'] -> A tempo

Driver 39: ['Alvito', 'Roriz'] -> A tempo
```

Figura 8: Percurso Eco.

Apesar de termos noção de que a visualização possa à primeira vista parecer confusa, o que sucedeu aqui foi a remoção da estrada que liga Lijo a Alvito em concorrência com a apresentação do caminho percorrido pelo estafeta 39.

Com isto podemos observar que seria suposto seguir o percurso:

*Path* = Silva -> Lijo -> Alvito -> Roriz

Mas com a alteração, o percurso seguido foi:

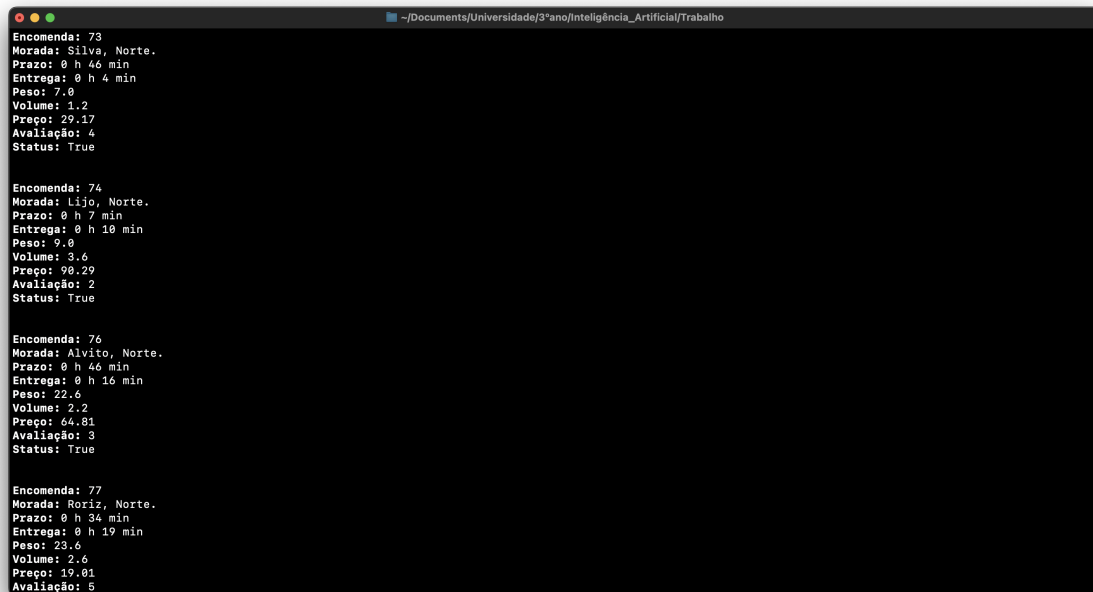
*Path* = Silva -> Lijo -> Roriz -> Alvito -> Roriz

É ainda nesta fase demonstrado se o estafeta consegue ou não fazer a sua entrega a tempo. Com base na *figura 8*, conseguimos perceber que apenas a encomenda de Lijo foi entregue fora de tempo e que portanto receberá uma penalização na sua avaliação.

## 6.4 Dados

Selecionando a opção de *Mostrar dados*, deparamo-nos posteriormente com a possibilidade de mostrar os *estafetas*, as *encomendas* ou as *rotas*.

Para demonstrar as atualizações da rota realizada neste exemplo, iremos da mesma forma mostrar as encomendas 73, 74, 76 e 77 devidamente atualizadas.



```
Encomenda: 73
Morada: Silva, Norte.
Prazo: 0 h 46 min
Entrega: 0 h 4 min
Peso: 7.0
Volume: 1.2
Preço: 29.17
Avaliação: 4
Status: True

Encomenda: 74
Morada: Lijo, Norte.
Prazo: 0 h 7 min
Entrega: 0 h 10 min
Peso: 9.0
Volume: 3.6
Preço: 90.29
Avaliação: 2
Status: True

Encomenda: 76
Morada: Alvito, Norte.
Prazo: 0 h 46 min
Entrega: 0 h 16 min
Peso: 22.6
Volume: 2.2
Preço: 64.81
Avaliação: 3
Status: True

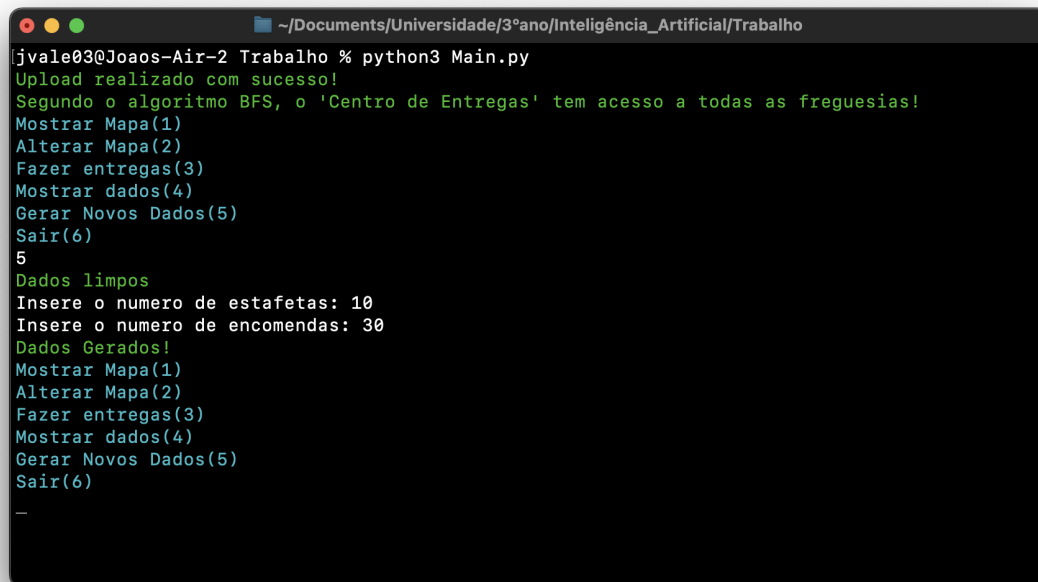
Encomenda: 77
Morada: Roriz, Norte.
Prazo: 0 h 34 min
Entrega: 0 h 19 min
Peso: 23.6
Volume: 2.6
Preço: 19.01
Avaliação: 5
```

Figura 9: Encomendas Finalizadas.

Com este exemplo é possível observar que a *entrega*, *avaliação* e o *status* foram devidamente atualizados, sendo a avaliação atribuída de forma randomizada, com uma penalização de 1 valor caso não cumpra o prazo.

## 6.5 Geração de dados

Por fim, a nossa última funcionalidade tem como objetivo povoar novamente a base de dados com novos dados, exatamente como mencionado anteriormente. Desta forma podemos escolher a quantidade de informação que queremos criar.



```
jvale03@Joao-Air-2 Trabalho % python3 Main.py
Upload realizado com sucesso!
Segundo o algoritmo BFS, o 'Centro de Entregas' tem acesso a todas as freguesias!
Mostrar Mapa(1)
Alterar Mapa(2)
Fazer entregas(3)
Mostrar dados(4)
Gerar Novos Dados(5)
Sair(6)
5
Dados limpos
Insere o numero de estafetas: 10
Insere o numero de encomendas: 30
Dados Gerados!
Mostrar Mapa(1)
Alterar Mapa(2)
Fazer entregas(3)
Mostrar dados(4)
Gerar Novos Dados(5)
Sair(6)
—
```

Figura 10: Gerar dados.

## 7 Conclusão

Ao longo deste trabalho prático encontrámos alguns obstáculos que nos obrigaram a pensar com um certo espírito crítico e a refletir sobre quais seriam as melhores estratégias para cada situação, nomeadamente na definição das *heurísticas*.

Em suma, apesar de alguns contratemplos, julgamos que o desenvolvimento deste projeto tenha sido bastante útil no sentido de consolidar aquilo que seriam os algoritmos estudados nas aulas, além disso, abriu-nos os olhos para a complexidade existente em aplicações tão familiares, como o *Google Maps*.