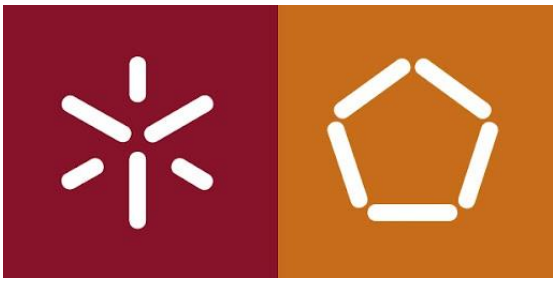

Universidade Do Minho
Departamento De Informática

# Trabalho Prático
# Inteligência Artificial

DAVID PEREIRA ALVES A93274
NUNO GONÇALO MACHADO RODRIGUES A90439
RUI MIGUEL BORGES BRAGA A93228
TIAGO LUCAS ALVES A93261

David

Gonçalo

Rui

Tiago

# Índice

# I. Introdução

Nesta 2ª fase do trabalho prático, pretendia-se a atualização do sistema realizado na 1ª fase, de modo a criar um sistema de recomendação de circuitos de entrega de encomendas para o caso anteriormente desenvolvido.

Os principais objetivos a atingir nesta parte da avaliação é o uso de técnicas de formulação de problemas, a aplicação de diversas estratégias para a resolução de problemáticas com o uso de algoritmos de procura, bem como, o desenvolvimento de mecanismos de raciocínio.

# II. Desenvolvimento do projeto

No nosso projeto optamos por dividir os ficheiros em quatro tipos, sendo que, cada tipo corresponde tem a sua função no cumprimento do objetivo proposto.

### TIPOS DE FICHEIROS:

| | |
|---|---|
| **tp.pl** | Ficheiros correspondentes as funcionalidades propostas no trabalho, incluindo, as funcionalidades extra. |
| **registos.pl** | Registos criados para realizar o caso pratico contêm a representação do conhecimento imperfeito e a evolução do conhecimento. |
| **mapa.pl** | Grafo criado com o intuito de demonstrar o caso pratico desenvolvido. |
| **helper.pl** | Ficheiro contêm predicados que servem de auxiliar as funcionalidades do *tp.pl* |

# III. Formulação do problema

| | |
|---|---|
| **TIPO DE PROBLEMA** | Problema de estado único |
| **TODOS OS ESTADOS POSSÍVEIS** | Todos os estados onde o estafeta se encontra numa cidade do mapa. |
| **ESTADO INICIAL** | Cidade de partida do estafeta. |
| **ESTADO FINAL/OBJETIVO** | Cidade de partida do estafeta tendo este já passado ponto(s) de entrega da(s) encomenda(s). |
| **OPERADORES** | Movimentar para uma cidade adjacente no mapa demorando um certo intervalo de tempo numa certa distância. |
| **CUSTO DA SOLUÇÃO** | Soma dos custos temporais e dos custos espaciais de todas as arestas que formam o caminho. |

# IV. Implementação

## A. Mapa

Este é o mapa que desenvolvemos para representar o caso prático, inspirado no mapa do jogo *Pokémon Ruby*.

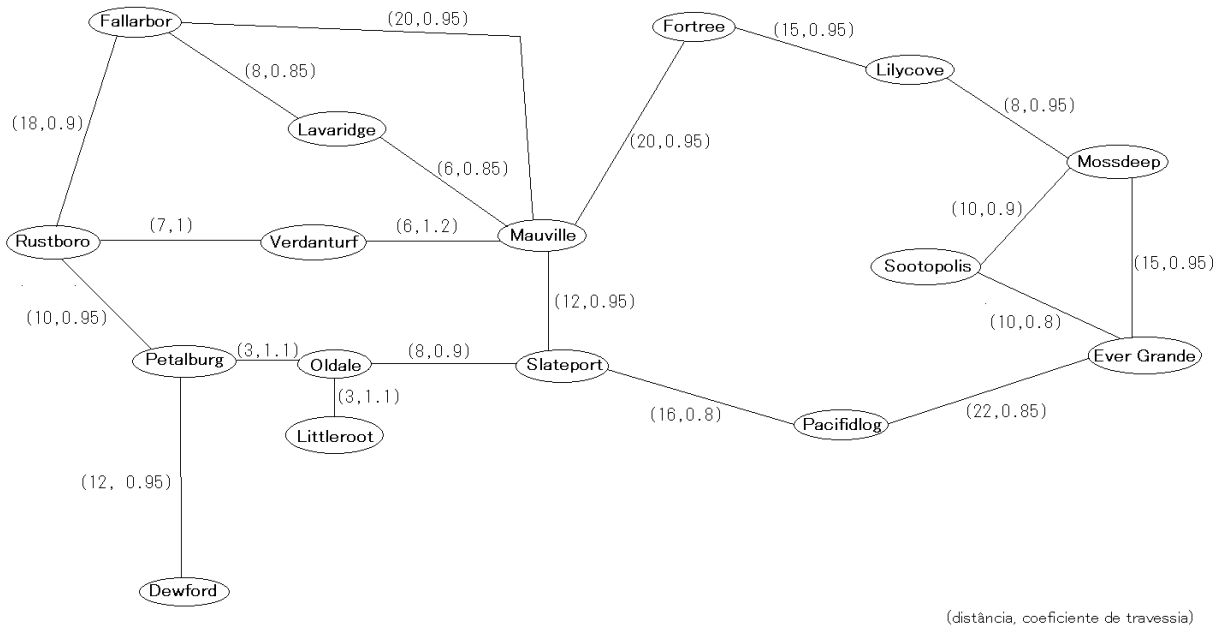


**Figura 1 – Mapa do caso prático desenvolvido**

O mapa é um grafo em que os nodos são os pontos de entrega e os seus nodos adjacentes ligam-se a eles por arestas.

Cada aresta tem a distância e o coeficiente de facilidade da travessia entre nodos (varia de 0.8 a 1.2). O coeficiente altera a velocidade média a que o veículo se desloca, na medida, em que a velocidade naquela aresta, dá-se pelo cálculo entre o coeficiente e a velocidade média.

Alteramos ligeiramente o mapa criado na 1ª fase, de modo, a melhorar os resultados obtidos, na procura de circuitos para facilitar o cálculo da velocidade na aresta.

## B. Registos

Os registos são constituídos por quatro argumentos principais: entrega, encomenda, circuito e prazo.

A entrega tem informação de quem é o estafeta e qual o transporte que utiliza, sendo que, o transporte pode ser bicicleta, moto e carro. Cada estafeta tem associado a ele 1 ou mais veículos e um ponto de partida, que pode ser, diferente de outros estafetas.

A encomenda indica quem é o cliente, local de entrega, volume da encomenda, preço de transporte e classificação (que é -1 caso a entrega não tenha sido concluída).

O circuito indica quais os locais por onde o estafeta passou para fazer a sua entrega e no seu caminho de retorno.

O prazo tem hora limite ditada pelo cliente, data limite, hora de entrega e data de entrega. A hora está representada no seguinte formato (H/M) e a data está representada no formato (D/M/A).

Para fazer o caso prático, atribuímos 5 estafetas à *Green Distribution* e registamos todas as encomendas que foram atribuídos a eles entre os dias 15/11/2021 e 19/11/2021.

```
registo(entrega(estafeta, transporte(veiculo)),
        encomenda(cliente, local, volume, preço, classificação),
        circuito([cidades]),
        prazo(hora_limite, data_limite, hora_de_entrega, data_de_entrega))
```

**Figura 2 - Formato dos registos**

## C. Circuito

Um circuito é definido como um trajeto, que começa no ponto de partida do estafeta, utilizando um do/s meio/s de transporte que tem ao seu dispor, levando uma encomenda com um certo peso, até ao ponto de entrega, onde fica 5 minutos (para fazer a entrega) e o seu regresso ao ponto de partida.

# D. Algoritmos Usados

## 1. Depth-First Search

É usado no predicado *circuitoDFS*, que calcula o trajeto entre um nodo inicial e um nodo final usando uma estratégia de procura primeiro em profundidade (não informada).

Esta estratégia consiste em expandir sempre o nó mais profundo da árvore de procura.

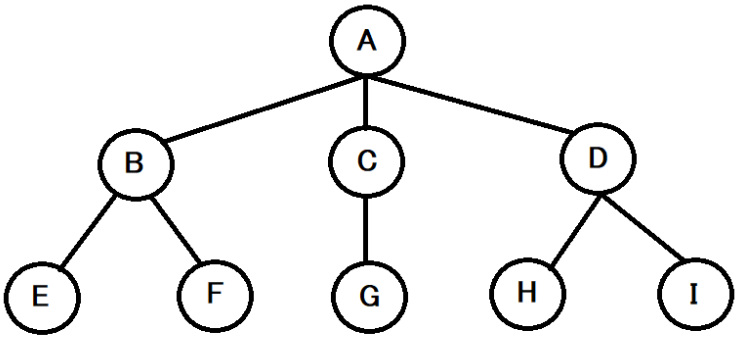


Figura 3 - Árvore de procura de exemplo

Travessia com objetivo de chegar a G:

| | |
|---|---|
| 1. | A |
| 2. | B |
| 3. | C |
| 4. | D |
| 5. | E |
| 6. | F |
| 7. | G |

## 2. Breadth-First Search

É usado no predicado *circuitoBFS*, que calcula o trajeto entre um nodo inicial e um nodo final usando uma estratégia de procura primeiro em largura (não informada).

Esta estratégia consiste em expandir primeiro os nós de menos profundidade de cada árvore.

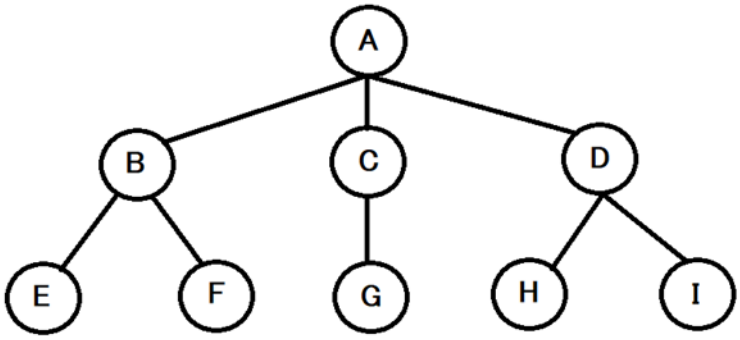


Figura 3 - Árvore de procura de exemplo

Travessia com objetivo de chegar a G:

| | |
|---|---|
| 1. | A |
| 2. | B |
| 3. | E |
| 4. | F |
| 5. | C |
| 6. | G |

## 3. Iterativa Limitada em Profundidade

É usado no predicado circuitoIDS, que calcula o trajeto entre um nodo inicial e um nodo final usando uma estratégia de procura iterativa limitada em profundidade (não informada).

Esta estratégia consiste em executar uma pesquisa primeiro em profundidade com um limite de profundidade. Limite esse, que é incrementado em cada sucessiva iteração.

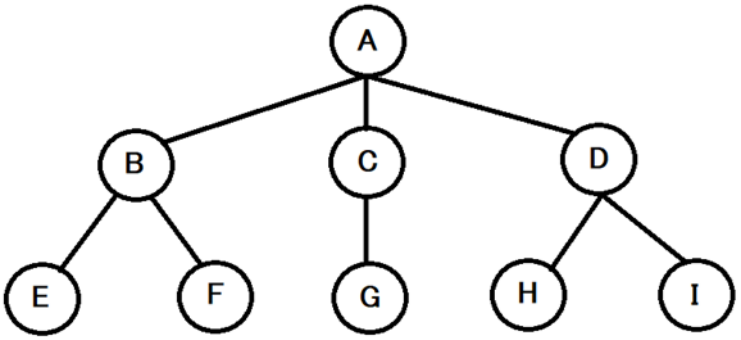


**Figura 3 - Árvore de procura de exemplo**

Travessia com objetivo de chegar a G:

| 1ª iteração | |
|---|---|
| 1. | A |
| 2. | B |
| 3. | C |
| 4. | D |

| 2ª iteração | |
|---|---|
| 1. | A |
| 2. | B |
| 3. | E |
| 4. | F |
| 5. | C |
| 6. | G |

## 4. Greedy

É aplicado nos predicados que identificam o circuito mais ecológico ou mais rápido. Calcula o trajeto entre um nodo inicial e um nodo final usando a estratégia de procura *greedy* (informada).

Dada uma heurística (heurísticas usadas referidas à frente), este algoritmo expande o nó que, segundo a heurística, está mais próximo da solução (menor custo estimado da heurística).

## 5. A*

É aplicado nos predicados que identificam o circuito mais ecológico ou mais rápido. Calcula o trajeto entre um nodo inicial e um nodo final usando a estratégia de procura A* (informada).

Dada uma heurística (heurísticas usadas referidas à frente), este algoritmo expande o nó que tenha a menor soma do custo estimado da heurística e do custo efetuado até então.

## E. Gerar Todos os Circuitos

Para gerar todos os circuitos de entrega que cubram um determinado território no sistema, basta introduzir o ponto de entrega da encomenda e o peso da mesma no predicado *gera_todos*/3.

O que este predicado faz é, numa fase inicial, filtra todos os estafetas existentes no sistema elegíveis para transportar aquela encomenda (de acordo com o/s veículo/s que possuem e o peso da encomenda) e, de seguida, atravessa a lista de estafetas elegíveis e gera (usando o *findall*) todos os circuitos possíveis entre o ponto de partida do estafeta e o ponto de entrega, usando uma pesquisa primeiro em profundidade para os encontrar (a escolha da estratégia de procura usado foi aleatória) e faz o mesmo no sentido inverso para completar todas as possibilidades.

## F. Identificar os circuitos com mais entregas (por volume e peso)

A identificação do circuito com maior número de entregas por volume ou peso no sistema é feita, usando, inicialmente, o predicado *findall* para guardar numa lista todos os circuitos existentes nos registos (no *circuitosMaisPeso* também é necessário guardar o peso presento no registo). Depois, percorre essa lista de circuitos, mudando os valores de duas outras listas, uma com circuitos (TRACKS), outra com valores (VALORES).

Caso o circuito não exista na lista TRACKS, é lá adicionado e é adicionado 1 à lista VALORES (no caso das entregas) ou o peso da encomenda (no caso do peso). Caso o circuito já exista, limita-se a modificar o seu elemento respetivo na lista VALORES, incrementando o seu número de entregas ou somando o peso da nova encomenda.

## G. Comparar Circuitos

Circuitos podem ser avaliados de acordo com 2 indicadores de produtividade: distância e tempo.

Para comparar 2 circuitos segundo a distância, limita-se a aferir a distância que cada um dos trajetos faz. O trajeto que percorrer menos distância, é naturalmente o melhor. Em caso de igualdade, é escolhido o transporte mais ecológico. Sendo assim, dá-se prioridade à bicicleta, de seguida à moto e, por último, ao carro.

Para comparar 2 circuitos segundo o tempo, é definido primeiro o tempo máximo (recebido como argumento no predicado), caso um dos circuitos demore mais que esse tempo, o outro será imediatamente melhor. Não sendo esse o caso e ambos os circuitos demorarem menos tempo a percorrer que o tempo máximo (ou ambos demorarem mais), é

dada prioridade ao circuito percorrido por bicicleta. Na alternativa dos circuitos terem sido percorridos pelos outros 2 veículos, é escolhido qual é que demorou menos tempo a percorrer.

## H. Circuito Mais Rápido

Para identificar o caminho mais rápido, depois de filtrados todos os estafetas elegíveis de transportar a encomenda, é calculada a heurística utilizada para cada nodo. Como o critério para identificar o caminho mais rápido é minimizar a distância, a heurística que utilizamos neste caso é a distância de cada nodo até ao ponto de entrega. A distância é estimada através da dedução do trajeto entre um nodo e o ponto de entrega, usando uma procura primeiro em largura que, como os custos associados às arestas são diferentes de 1, nem sempre deduz a melhor solução.

Depois de calculada a heurística, verificamos qual o estafeta que, segundo a nossa estimativa, está em melhores condições para obter a melhor solução e a partir do ponto de partida deste estafeta inferimos o percurso ideal usando os algoritmos *greedy* ou A*. Caso mais que um estafeta esteja em condições idênticas no início, é inferido o percurso para todos esses e no final é escolhido o circuito com menor custo.

## I. Circuito Mais Ecológico

Para identificar o caminho mais ecológico, depois de filtrados todos os estafetas elegíveis de transportar a encomenda, é calculada a heurística utilizada para cada nodo 3 vezes, variando o veículo utilizado. Como o critério para identificar o caminho mais ecológico é conseguir cumprir com o tempo máximo de entrega escalonado pelo cliente, usando o veículo mais ecológico possível, a heurística que utilizamos nesta situação, é o tempo de cada nodo até ao ponto de entrega, usando um certo veículo que carregue um determinado peso. Deduzimos o trajeto entre um nodo e o ponto de entrega, usando uma procura primeiro em largura que nem sempre deduz a melhor solução, inferindo depois o tempo demorado, tendo em conta a velocidade média a que o veículo se desloca.

Depois de calculada a heurística, inferimos o percurso ideal usando os algoritmos *greedy* ou A* para todos os estafetas definidos como elegíveis. Finalmente, verificámos qual o circuito que demore menos tempo entre todos e é esse que é retornado como solução.

## J. Avaliar Impacto de Múltiplas Entregas

Para avaliar as alterações e o impacto de cada estafeta poder passar a efetuar mais que uma entrega em cada circuito, geramos os circuitos necessários para fazer todas as entregas sequencialmente e somamos os custos associados a esses circuitos (tempo e distância), que são comparados com a alternativa de fazer todas as entregas simultaneamente.

Para encontrar o circuito de entregas simultâneas, usamos uma estratégia de procura não informada (*breadth-first search*). O circuito é iniciado no ponto de partida definido e o algoritmo vai à procura de todos os pontos de entrega das encomendas que tem a realizar sem qualquer ordem definida. Quando são feitas todas as entregas, é novamente calculado um trajeto, este que vai do último ponto de entrega até ao ponto de partida do estafeta.

```
?- cmpMultiplasEntregas(dewford, [encomenda(oldale, 2), encomenda(littleroot, 2)], bicicleta, D1, T1, DM, TM).
TIME ENTREGAS SIMULTÂNEAS: 265.534661177256
DIST ENTREGAS SIMULTÂNEAS: 36
TIME ENTREGAS SEQUENCIAIS: 444.00689885390005
DIST ENTREGAS SEQUENCIAIS: 66
```

**Figura 4 - Diferença com duas entregas**

```
?- cmpMultiplasEntregas(dewford, [encomenda(evergrande, 40), encomenda(lilycove, 20), encomenda(sootopolis, 30)], carro, D1, T1, DM, TM).
TIME ENTREGAS SIMULTÂNEAS: 604.560230790881
DIST ENTREGAS SIMULTÂNEAS: 174
TIME ENTREGAS SEQUENCIAIS: 1176.5991247555878
DIST ENTREGAS SEQUENCIAIS: 404
```

**Figura 5 - Diferença com três entregas**

```
?- cmpMultiplasEntregas(dewford, [encomenda(oldale, 40), encomenda(slateport, 20), encomenda(sootopolis, 30), encomenda(evergrande, 10)], carro, D1, T1, DM, TM).
TIME ENTREGAS SIMULTÂNEAS: 473.0011526141558
DIST ENTREGAS SIMULTÂNEAS: 142
TIME ENTREGAS SEQUENCIAIS: 989.4977131625892
DIST ENTREGAS SEQUENCIAIS: 340
```

**Figura 6 - Diferença com quatro entregas**

Como podemos ver o impacto é gritante e a diferença aumenta quanto mais encomendas forem feitas em simultâneo.

## K. Evolução do Conhecimento e Conhecimento Imperfeito

De forma a evoluir o conhecimento, o grupo implementou diversos métodos com o objetivo de introduzir informação correta na aplicação. Para validar essa informação, criaram-se vários invariantes estruturais e referenciais. Os invariantes estruturais impedem erros na estrutura da base de dados tais como repetições de informação. Os invariantes referenciais limitam a introdução de informação que não respeite certos critérios tais como o estafeta não estar registado ou a cidade de entrega não estar no mapa. De forma a inserir novos registos, o grupo desenvolveu o método *inserirRegisto* que, introduzindo os parâmetros certos (indicar os parâmetros se acharem necessário) insere um novo registo na base de dados. A equipa criou também um método *inserirEstafeta* com o mesmo intuito. Para registar a conclusão de uma encomenda, implementaram o método *registarEntrega* que dado o estafeta, a nota da entrega, a hora e data de entrega, altera o registo do estafeta, dando-o por concluído.

Em termos de conhecimento imperfeito, o grupo desenvolveu um sistema de inferência indica o valor lógico de uma dada questão. O valor lógico pode ser verdadeiro, falso ou desconhecido. Deste modo, teve também de formalizar o PMF criando métodos de negação forte. Por fim, criou-se um método para identificar exceções dos métodos nas quais o sistema deve responder com o valor lógico negativo.

# V. Resultados

| Estratégia | Tempo (Milissegundos) | Complexidade Temporal | Complexidade Espacial | Indicador/Custo | Encontrou a melhor solução |
|---|---|---|---|---|---|
| DFS | 1 | $O(b^d)$ | $O(b^d)$ | Distância/Tempo | Não |
| BFS | 1 | $O(b^m)$ | O(bm) | Distância/Tempo | Não |
| IDS | > 1 | $O(b^d)$ | O(bd) | Distância/Tempo | Não |
| Greedy | 9 | Melhor caso: O(bd)<br>Pior caso: $O(b^m)$ | Melhor caso: O(bd)<br>Pior caso: $O(b^m)$ | Distância/Tempo | Não |
| A* | 9 | Pior caso: $O(b^d)$ | Pior caso: $O(b^d)$ | Distância/Tempo | Sim |

## LEGENDA:

**b** O máximo fator de ramificação (o número máximo de sucessores de um nó) da árvore de pesquisa

**d** A profundidade da melhor solução

**m** A máxima profundidade do espaço de estados.

## EXECUÇÕES:

```
?- statistics(runtime,[Start|_]), circuitoDFS(dewford, verdanturf, bicicleta, 4, S), statistics(runtime,[Stop|_]), Runtime is Stop - Start.
Start = 346,
S = circuito(bicicleta, 4, [dewford, petalburg, rustboro, verdanturf, rustboro, petalburg, dewford]),
Stop = 347,
Runtime = 1 .
```

**Figura 7 - Execução da pesquisa DFS**

```
?- statistics(runtime,[Start|_]), circuitoBFS(dewford, verdanturf, bicicleta, 4, S), statistics(runtime,[Stop|_]), Runtime is Stop - Start.
Start = 522,
S = circuito(bicicleta, 4, [dewford, petalburg, rustboro, verdanturf, rustboro, petalburg, dewford]),
Stop = 523,
Runtime = 1 .
```

**Figura 8 - Execução da pesquisa BFS**

```
?- statistics(runtime,[Start|_]), circuitoIDS(dewford, verdanturf, bicicleta, 4, S), statistics(runtime,[Stop|_]), Runtime is Stop - Start.
Start = Stop, Stop = 375,
S = circuito(bicicleta, 4, [dewford, petalburg, rustboro, verdanturf, rustboro, petalburg, dewford]),
Runtime = 0 .
```

**Figura 9 - Execução da pesquisa IDS**

```
?- statistics(runtime,[Start|_]), distHeuristica(dewford, H), getHeuristicaOfNodo(nodo(verdanturf), H, RH), greedy(dewford, nodo(verdanturf), H, RH, S), statistics(runtime,[Stop|_]), Runtime is Stop - Start.
Start = 486,
H = [nodo(dewford, 0), nodo(petalburg, 24), nodo(oldale, 30), nodo(littleroot, 36), nodo(slateport, 46), nodo(rustboro, 44), nodo(verdanturf, 58), nodo(fallarbor, 80), nodo(..., ...)|...],
RH = 58,
S = [verdanturf, rustboro, petalburg, dewford]/29,
Stop = 495,
Runtime = 9 .
```

**Figura 10 - Execução da pesquisa greedy**

```
?- statistics(runtime,[Start|_]), distHeuristica(dewford, H), getHeuristicaOfNodo(nodo(verdanturf), H, RH), aEstrela(dewford, nodo(verdanturf), H, RH, S), statistics(runtime,[Stop|_]), Runtime is Stop - Start.
Start = 509,
H = [nodo(dewford, 0), nodo(petalburg, 24), nodo(oldale, 30), nodo(littleroot, 36), nodo(slateport, 46), nodo(rustboro, 44), nodo(verdanturf, 58), nodo(fallarbor, 80), nodo(..., ...)|...],
RH = 58,
S = [verdanturf, rustboro, petalburg, dewford]/29,
Stop = 518,
Runtime = 9 .
```

**Figura 11 - Execução da pesquisa A***

# VI. Conclusão

Procuramos neste trabalho testar a nossa capacidade e destreza em programação lógica, bem como, a estruturação de problemas no sentido da representação de conhecimento e mecanismos de raciocínio logico.

Acreditamos ter desenvolvido todas as funcionalidades pedidas pelo enunciado do projeto, com sucesso, incluindo algumas das funcionalidades opcionais, que entendemos serem relevantes para obter diferentes resultados para o problema tratado, como por exemplo, os estafetas poderem ter pontos de partida diferentes uns dos outros ou a adição de novos estafetas.

Ainda assim o grupo admite que existem aspetos a melhorar do seu trabalho. Podíamos ter mais além e ter implementado a obtenção dos circuitos mais ecológicos ou mais rápidos quando os estafetas fazem várias entregas no mesmo percurso. Uma das heurísticas que podia ser usada, nesse caso, seria o mínimo da proximidade da localização aos pontos de entrega restantes.