Relatório IA - Grupo 18

A100661 - Daniel Henrique Cracel Rodrigues

A100691 - Francisco Manuel Afonso

A100615 - João Miguel Mendes Moura

A100655 - Roberto Lopes Capela

Dezembro 2023

Conteúdo

Introdução

Este projeto foi elaborado no contexto da unidade curricular de Inteligência Artificial. Tem como o propósito consolidar conhecimentos sobre a disciplina como por exemplo o desenvolvimento de diversos algoritmos de procura, neste caso tendo em conta a sustentabilidade num sistema de entregas.

Na realização deste projeto era pretendido sensibilizar e motivar os alunos para um assunto relevante na atualidade que é a sustentabilidade na resolução de problemas.

1.1 Objetivos

O projeto tem como objetivo realizar um sistema de entregas que utilize o meio de entrega de encomendas mais sustentável/ecológico para o planeta terra. Neste contexto, existem ao dispor vários estafetas com diferentes meios de transporte que têm diferentes níveis de consumo.

Descrição do problema

Para melhor entender o problema que nos foi proposto resolver com a execução deste projeto, é melhor antes entendê-lo descrevendo-o indicando o tipo do problema, os estados quer o inicial quer o estado objetivo, os operadores e o custo da solução.

2.1 Tipo do Problema

Para identificar o tipo do problema temos antes de saber se é um ambiente determinístico ou não e se é acessível ou não.

Para um ambiente ser determinístico tem de ser um ambiente onde qualquer ação tem um único efeito. Não determinístico é o contrário. O mundo real, para os seres humanos, é um exemplo de um ambiente não determinístico.

Para sabermos se um ambiente é ou não acessível, temos de ver se o agente consegue ou não obter informações sobre o estado do ambiente.

No caso do nosso problema deparamos-nos com um problema de estado único porque o ambiente é determinístico e acessível. O agente sabe exatamente o estado em que estará e a solução é uma sequência.

2.2 Estados

O problema pode ser resolvido através da procura de um caminho entre o estado inicial e um estado objetivo.

2.2.1 Estado Inicial

O Estado inicial é o estado em que o nosso problema se encontra no inicio do programa. No caso do nosso problema o seu estado inicial é o nodo (local onde a entrega começa).

O nosso estado inicial varia entre a Central que por base é onde todas as nossas encomendas começam a ser distribuídas e pode ser também a rua onde a encomenda anterior foi entregue, isto é, se o estafeta levou mais do que uma encomenda, na primeira ela sai da central até á rua X, logo a segunda encomenda que ele levou, vai ter como nodo inicial a rua X.

2.2.2 Estado Objetivo

O estado objetivo é o nodo onde a nossa encomenda é entregue.

2.3 Operadores

Quanto aos operadores temos de definir a pré-condição e os efeitos. No nosso problema a pré-condição é haver ruas para circular e o efeito é conduzir nas ruas para entregar a encomenda.

2.4 Custo de Solução

Para definir o nosso problema falta nos ainda falar sobre o custo de solução. O custo de solução é o quanto custa cada caminho. Neste caso o nosso custo de solução é a distância entre as duas ruas medido em quilómetros. A qualidade da solução é medida por qual é a

solução com emissões mais baixas de CO2. O calculo que é feito para definir quanto CO2 foi emitido na viagem varia de veiculo para veiculo.

No caso da bicicleta a formula que usamos para calcular o CO2, fazemos uma estimativa de que por distancia é emitido 0.001 gramas de CO2:

$$co2bicicleta = distancia * co2porkmpessoa$$

No caso da mota o calculo feito é o seguinte:

$$co2mota = (distancia/gasoleoporkmmota) * co2porkmmota$$

No caso do carro o calculo é semelhante ao da mota e é o seguinte:

$$co2carro = (distancia/gasoleoporkmcarro) * co2porkmcarro$$

Grafo

No projeto era nos pedido que fosse representado os diferentes pontos de entrega em forma de grafo.

Quanto aos nossos "mapas"nós desenvolvemos os nossos próprios começando por fazer uma mapa mais pequeno e simples e depois expandi-lo para mapas mais complexos.

Para a construção do grafo nos usamos como base o código fornecido nas aulas práticas em que temos funções que criam as ligações entre os nodos indicando também o custo de caminho entre eles os dois, que no nosso caso, são, como já foi referido antes, a distância em quilometros entre as duas. Usamos também uma função que adiciona a heurística a cada nodo indicando a mesma. Para o desenho gráfico do grafo, utilizamos a livraria *NetworkX* que faz o desenho dos grafos.

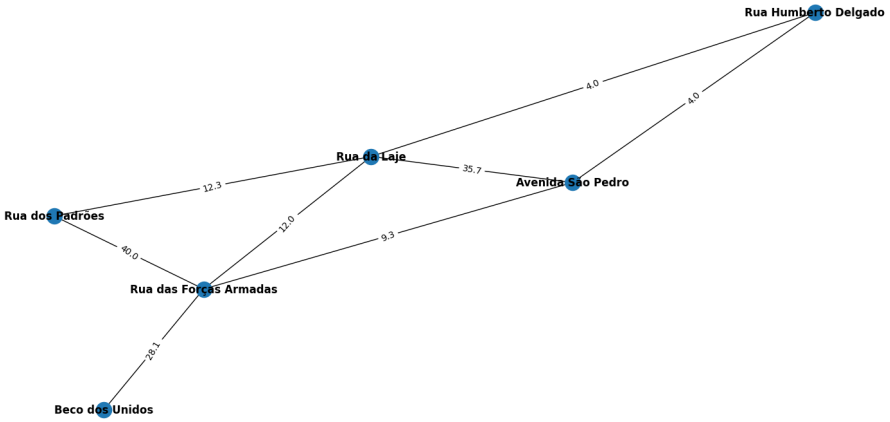

Figura 3.1: Mapa 1

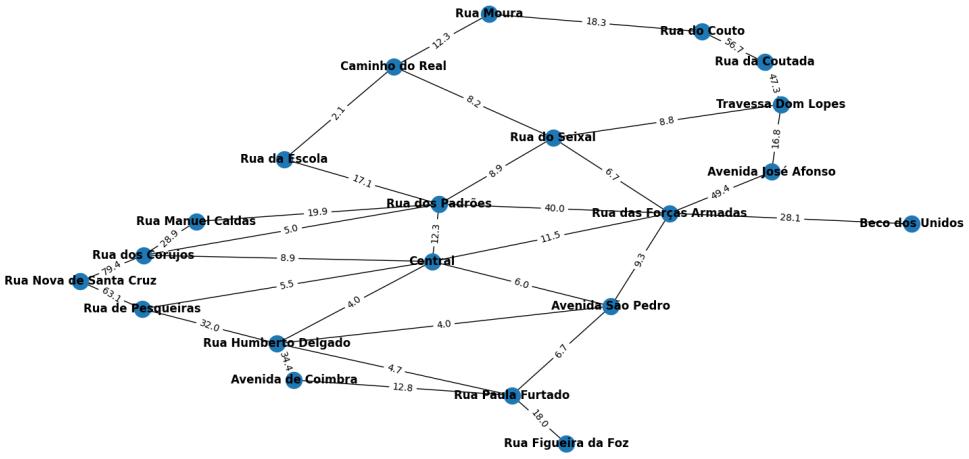

Figura 3.2: Mapa 5

Algoritmos de Procura

Quando falamos de algoritmos de procura, estamos-nos a referir á forma como é feita a ordem de expansão do nó.

Para escolher qual algoritmo usamos para cada encomenda, fazemos a pergunta ao utilizador, de se pretende escolher o algoritmo pelo que dá o caminho com a distância menor ou pelo que tem menor emissão de CO2.

No nosso projeto utilizamos dois tipos de algoritmos de procura. Algoritmos de procura não-informada (cega) e algoritmos de procura informada (heurística).

4.1 Procura Não-Informada

Estratégias de procura não informada usam apenas as informações disponíveis na definição do problema. Neste caso usam a distancia entre as ruas.

Neste tipo de procura, no nosso projeto, usamos os algoritmos: Primeiro em Largura (BFS), Primeiro em Profundidade (DFS) e Custo Uniforme.

4.1.1 Primeiro em Largura (BFS)

Esta estratégia como o nome indica faz a expansão pelos nós de menor profundidade. Por um lado é uma boa estratégia porque faz uma procura sistemática, mas por outro é má porque normalmente demora muito tempo e ocupa muito espaço.

Quanto ás suas propriedades ela é completa se for finito. No que toca ao espaço, guarda cada nó em memória é ótima se o custo for 1. A níveis de tempo supondo fator de ramificação b então

$$n = 1 + b + b2 + b3 + \ldots + b^n = O(b^d)$$

é exponencial em d.

4.1.2 Primeiro em Profundidade (DFS)

Nesta estratégia é expandido sempre um dos nós mais profundos da árvore. Por um lado esta estratégia é boa porque é necessária pouca memória, mas por um lado é má porque não pode ser usada em árvores com profundidade infinita. No nosso caso utilizamos este algoritmo de procura porque temos uma amostra finita.

Quanto ás suas propriedades, não é completa porque falha em espaços de profundidade infinita. Quanto ao espaço é espaço linear e não é ótima.

4.1.3 Custo Uniforme

Nesta estratégia é expandido sempre para o nó com o menor custo da lista de estados não expandidos que é medido pelo custo de solução (distância entre as ruas em quilómetros). Esta estratégia será igual á procura BFS, se todos os custos forem iguais.

Quanto ás suas propriedades, esta procura é completa, se o custo da etapa for maior que alguma constante positiva. É uma procura ótima

Criterion	Breadth-First	Uniform-Cost	Depth-First
Complete?	Yes	Yes	No
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$
Optimal?	Yes	Yes	No

Figura 4.1: Comparação entre estratégias de procura não-informada

4.2 Procura Informada

Na procura informada, é usada informação sobre o problema para evitar que o algoritmo de procura fique "perdido". Para isso é utilizada a heurística.

4.2.1 Heurística

As heurísticas são especificas a cada problema. Funcionam basicamente como um atalho na procura. Implementar uma heurística é desafiador, porque avaliar com precisão o valor de uma solução e medir certos conhecimentos de maneira que permita uma análise matemática do seu impacto no processo de busca é complexo.

Na escolha da nossa heurística foi tomado em conta o facto de grande parte das encomendas saírem da central, por isso, achamos adequado que a nossa heurística fosse a distância em linha reta até á central. Os únicos casos em que a encomenda não sai da central são nas que o estafeta leva mais do que uma encomenda para fazer. Isto fará com que a segunda encomenda começará na rua em que foi entregue a encomenda anterior. Mas ainda assim o local de onde o estafeta sai é a central.

4.2.2 Procura Gulosa (Greedy-Search)

Nesta estratégia de procura, é expandido o nó que parece estar mais perto da solução. Isto significa que o algoritmo expandirá para o nodo que tem heurística menor. Neste caso para o nodo mais perto da central em linha reta.

A procura gulosa não é uma procura completa porque pode entrar num ciclo logo fica suscetível a falsos arranques. Na complexidade no tempo e de espaço é

$$O(b^m)$$

mas com uma boa função heurística pode diminuir consideravelmente. Quanto ao espaço ele matem todos os nós em memória. Não é uma procura ótima porque não encontra sempre a melhor solução.

4.2.3 Procura A*

Esta estratégia de procura combina a procura gulosa com a procura uniforme. A procura A* soma o custo total, até á altura, para chegar ao estado n, com o custo estimado para chegar ao objetivo (heurística).

A procura A* é um algoritmo completo e ótimo.

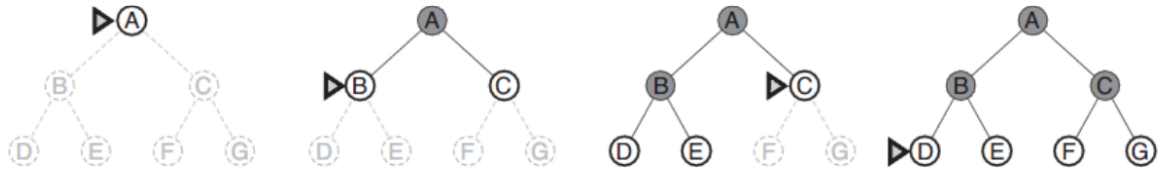

Figura 4.2: Procura BFS

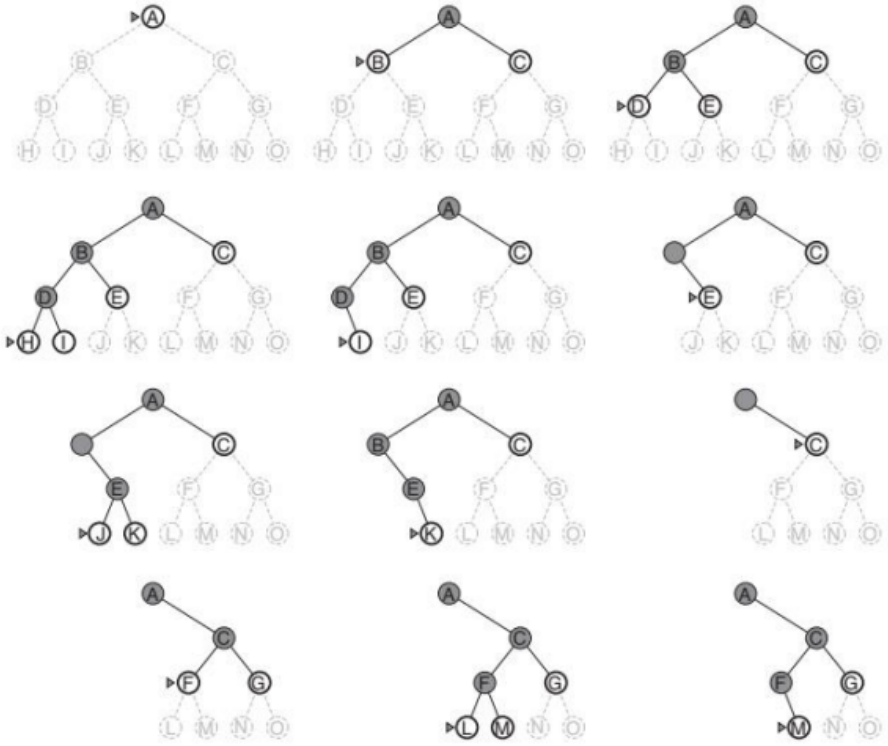

Figura 4.3: Procura DFS

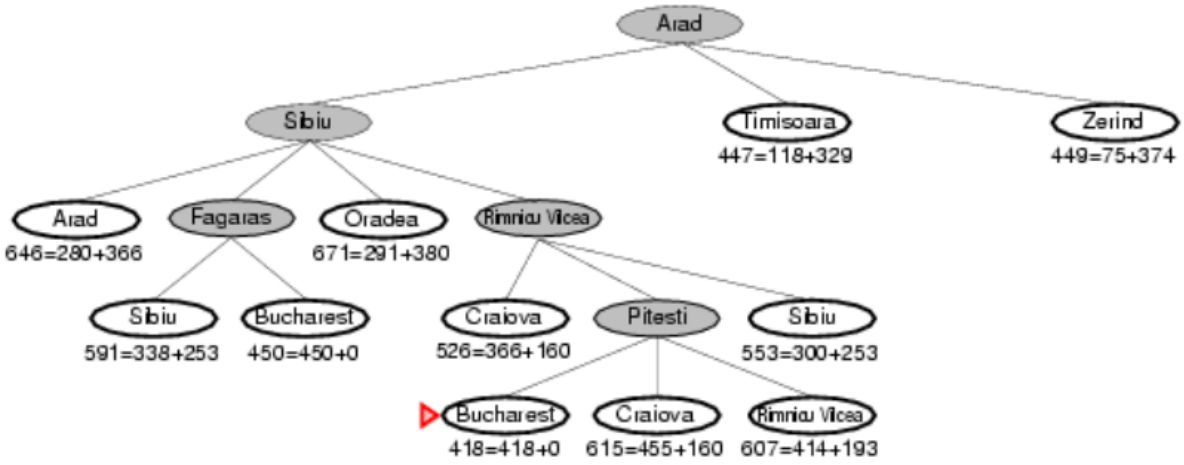

Figura 4.4: Procura A*

Algoritmo	Completo	Otimal	Tempo complexidade	Espaço complexidade
Primeiro em Largura (BFS)	Sim	Se todos os custos escalonados forem iguais	$O(b^d)$	$O(b^d)$
Primeiro em Profundidade (DFS)	Não	Não	$O(b^m)$	$O(bm)$
Iterativa	Sim	Se todos os custos escalonados forem iguais	$O(b^d)$	$O(bd)$
Custo Uniforme	Sim	Sim	Nº de nodos com $g(n) \leq C^*$	
Gulosa	Não	Não	no pior caso : $O(b^m)$ No melhor caso: $O(bd)$	
A*	Sim	Sim (se a heurística for admissível)	Nº de nodos com $g(n)+h(n) \leq C^*$	

Figura 4.5: Comparação entre algoritmos de procura informada e não-informada

Funcionalidades

Para o bom funcionamento do nosso projeto nos implementamos várias funcionalidades.

5.1 Load e Save

A primeira funcionalidade com que o utilizador se depara na execução do nosso projeto é a função de *load* e por consequência a função de *save*.

Consideramos que estas funcionalidades eram necessárias porque desta forma os ficheiros `JSON` apenas são alterados no fim da execução do programa.

5.1.1 Load

A função de *load* guarda toda a informação contida nos ficheiros `JSON` em catálogos na memória.

5.1.2 Save

A função *save* guarda tudo o que tem nos catálogos nos ficheiros respetivos.

5.2 LogIn e SignUp

Com a implementação de um cliente, criamos as opções de *login* e *signup*.

5.2.1 LogIn

Ao dar *login*, se for o *admin*, este tem permissão para realizar as encomendas, isto é, para expedi-las.

Se o *login* for dado com outro cliente pode fazer tudo menos realizar as encomendas.

5.2.2 SignUp

Para dar *SignUp* é pedido que o utilizador introduza o seu *username* e a *password*. De seguida o sistema regista o utilizador na base de dados.

5.3 Imprime

Esta funcionalidade como o nome indica, imprime cada catalogo tal e qual ele está na memória.

Desta forma o utilizador consegue ver informações sobre quais as encomendas que faltam realizar, quais já foram realizadas, com informação sobre a encomenda, o tempo que o utilizador estabeleceu, a classificação do estafeta e a classificação que foi dada pela encomenda e ainda imprime o *ranking* dos estafetas.

5.4 Registar Estafeta

Com a opção de registar um estafeta, o sistema regista o estafeta com o nome dado no login. Automaticamente o sistema regista o estafeta no *ranking* de estafetas com o seu nome e o seu *ranking*.

5.5 Criar Encomenda

Ao criar uma encomenda, é pedido ao utilizador que indique o peso da encomenda em quilos, o local onde quer que a encomenda seja entregue (o nome da rua) e o tempo limite que pretende que a encomenda seja feita (em minutos).

De seguida, o sistema regista a encomenda no catálogo das entregas por realizar gerando o id da encomenda automaticamente, atribuindo um estafeta que esteja registado no *ranking*, escreve o nome do cliente que realizou a encomenda e atribui, tendo em conta o peso da encomenda, um veiculo. Se a encomenda pesar entre um quilo e cinco, é atribuída uma bicicleta, se pesar entre cinco quilos e vinte, atribui uma mota e entre vinte e cem quilos é atribuído um carro. Desta forma nós priorizamos a sustentabilidade á velocidade da entrega.

5.6 Realizar Encomenda

Aqui como o nome indica, são realizadas as encomendas.

A única pessoa que pode realizar as encomendas é o admin.

Primeiro o programa pergunta a maneira como o utilizador pretende que o algoritmo seja escolhido, se pretende que seja por distância ou por CO2.

5.6.1 Várias encomendas

De seguida é pedido que o utilizador introduza o id da encomenda que pretende que já realizada. Aqui o programa vê qual foi a encomenda pedida e tendo em conta o peso da encomenda e o veiculo em que vai ser transportada, vê se é possível levar mais alguma encomenda. Se for possível, o sistema informa o utilizador que vai levar a encomenda solicitada e as outras encomendas e altera na base de dados o estafeta que vai realizar a encomenda que é o estafeta da encomenda que o utilizador indicou.

Caso tenha sido levada mais do que uma encomenda, o sistema passa a fazer a próxima encomenda, saindo do local onde entregou a encomenda anterior.

5.6.2 Informações Sobre a Encomenda

De seguida o sistema realiza a primeira encomenda por ordem da que tem o id mais pequeno. O sistema, de seguida, indica qual o algoritmo que foi usado e o tempo que este demorou a chegar ao caminho final, o caminho desde o local de inicio até á entrega e a distância que percorreu em quilómetros. O sistema indica também o tempo em minutos que demorou a fazer a entrega, quantas gramas de CO2 foram emitidas durante a realização da encomenda e quanto a viagem custou.

5.6.3 Classificar Estafeta

Se a encomenda foi feita dentro do tempo indicado o sistema diz que a entrega foi feita a tempo. Se não, o sistema informa o utilizador e automaticamente faz uma dedução de 0.5 na avaliação feita pelo utilizador.

Após feita a classificação do estafeta o sistema atualiza no *ranking* a sua classificação.

5.6.4 Encomendas Entregues

Após serem feitas as encomendas, o sistema retira-as do catálogo de encomendas por fazer e escreve-o no catálogo de encomendas já feitas.

Conclusão

No desenvolvimento deste projeto adquirimos um largo espetro de conhecimentos tanto relacionados com a `UC`, como por exemplo, a aplicação de algoritmos de procura mas também consolidamos aprendizagem da linguagem de programação *Python*.

Implementamos com sucesso um sistema capaz de realizar entregas suportadas em diversos algoritmos de procura.

Acreditamos ter atingidos os objetivos que nos foram colocados para a realização deste projeto.

6.1 Análise dos Resultados

Ao longo do desenvolvimento e testes do nosso projeto deparamos-nos com o facto de que em 44 simulações de encomendas que foram feitas, em 22 delas foi escolhido o algoritmo de procura *Greedy*, em 13 o `A*`, em 5 o `BFS`, em 2 o `DFS`, e em 2 o Custo Uniforme.

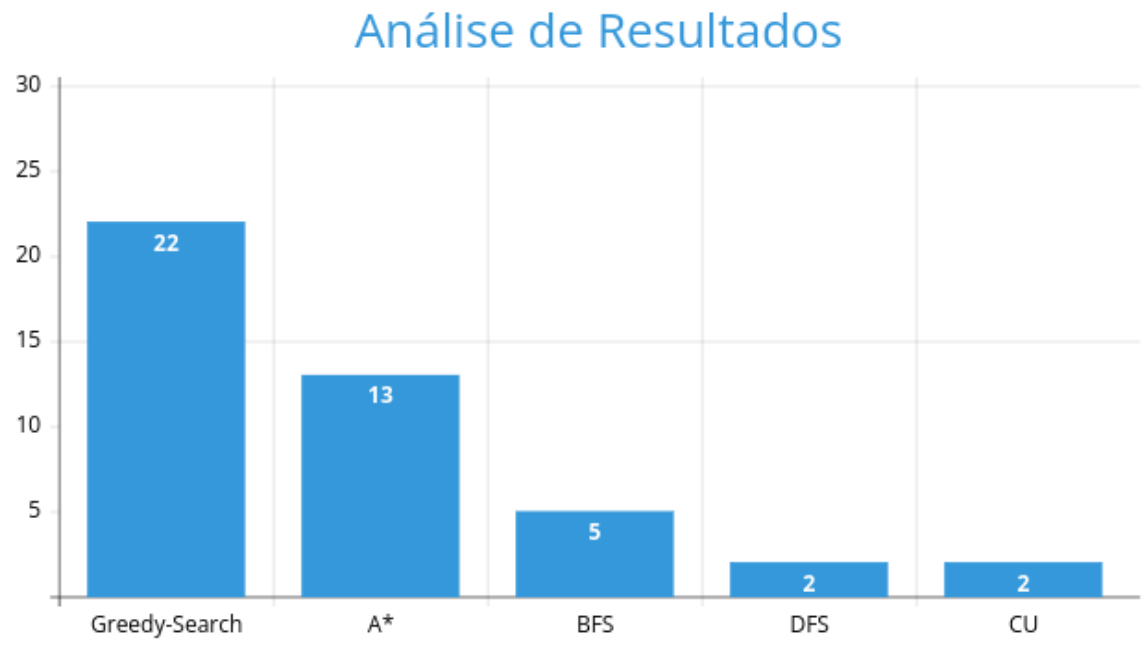

Figura 6.1: Algoritmos escolhidos

Portanto podemos verificar que em 79.5 porcento das vezes é escolhido um algoritmo de procura informada.

Com estes resultados podemos concluir que os algoritmos de procura informada, no nosso sistema, chegam a caminhos com distâncias mais pequenas e por consequência com emissões de CO2 menor na maior parte das vezes.

6.2 Trabalho Futuro

Por ultimo, apresentamos algumas funcionalidades que poderia ainda ser implementado.

Por exemplo, uma funcionalidade que permita que mais do que um estafeta esteja a entregar encomendas em simultâneo. Outra funcionalidade que podíamos implementar é a noção de trânsito, isto é, simular congestionamento fazendo o estafeta demorar mais na entrega.

No entanto, decidimos não implementar esta funcionalidade de trânsito porque consideramos que seria uma implementação complexa para o tempo que tínhamos para realizar o projeto. Optamos assim por priorizar as funcionalidades que nos foram propostas realizar no enunciado do projeto.