Universidade do Minho

Escola de Engenharia

Licenciatura em Engenharia informática

Resolução de Problemas - Algoritmos de procura

Inteligência Artificial

Ano Letivo de 2023/2024

André Pimentel Filipe, A96890

Manuel Vasconcelos Amaral Dos Santos Fernandes, A93213

Marcelo Araújo De Sousa, A81971

Tiago Granja Rodrigues, A100827

Avaliação pelos pares.

A96890 André Pimentel Filipe DELTA = 0

A93213 Manuel Vasconcelos Amaral Dos Santos Fernandes DELTA = 0

A81971 Marcelo Araújo De Sousa DELTA = 0

A100827 Tiago Granja Rodrigues DELTA = 0

1 Índice

2 Introdução

No âmbito da disciplina de Inteligência Artificial da Licenciatura em Engenharia Informática da Universidade do Minho, foi proposto trabalho que coloca em prática conceitos essenciais de resolução de problemas através da aplicação de algoritmos de procura.

Este trabalho prático tem como foco principal a implementação e análise de diversas estratégias de procura em grafos, visando abordar um problema específico que simula uma situação real e atual: a otimização de rotas de entrega para uma empresa de distribuição com um forte compromisso com a sustentabilidade.

Ao longo deste relatório, iremos detalhar todo o processo de implementação dos algoritmos escolhidos. Serão abordados os passos para a formulação do problema, a representação dos dados em forma de grafo, e a descrição detalhada de cada estratégia de procura utilizada, incluindo os algoritmos não informados e informados.

Além disso, faremos uma comparação entre os resultados obtidos por cada um dos algoritmos, analisando seu desempenho em termos de eficiência, eficácia e adequação às exigências do cenário proposto.

3 Descrição do problema

O problema consiste em alocar um conjunto de encomendas a um grupo de estafetas, cada um utilizando um meio de transporte específico, de forma que as entregas sejam realizadas dentro dos prazos estabelecidos, minimizando o seu custo operacional e considerando a sustentabilidade ambiental.

Os estafetas podem escolher entre bicicletas, motos e carros, cada um com características próprias de capacidade de carga e velocidade. A seleção do veículo tem um impacto direto no custo e na pegada ecológica da entrega. As bicicletas representam a opção mais ecológica, enquanto os carros, apesar de sua maior capacidade de carga, têm um impacto ambiental mais significativo.

O cenário de entrega é representado por um grafo, onde os nodos correspondem a pontos de entrega e as arestas representam as ruas, com seus respetivos custos (distâncias).

O objetivo principal do problema é o planejamento eficiente das rotas de entrega. Para isso, são implementados e avaliados diversos algoritmos de procura, que variam desde métodos não informados, como Busca em Largura (BFS) e Busca em Profundidade (DFS), até métodos informados, como o algoritmo A* e a Busca Gulosa (Greedy). Cada algoritmo tem suas particularidades, vantagens e desvantagens, sendo crucial analisar qual se adapta melhor às necessidades e restrições do serviço de entregas.

4 Formulação do problema

Tendo em conta que o agente sabe exatamente o estado em que estará, e a solução será uma sequência de ações, podemos concluir que este é um problema de estado único.

- **Representação do estado:** Grafo orientado onde cada aresta representa as ruas de uma cidade e cada nodo representa as interseções das ruas (pontos de entrega).
- **Estado inicial:** Estafeta com as entregas atribuídas no ponto de recolha.
- **Estado/teste objetivo:** Todas as encomendas são entregues.
- **Operadores:** Deslocar entre os nodos e entregar encomendas.
- **Solução:** Um caminho válido (sequência de ações) que leva à entrega de todas as encomendas.
- **Custo da solução:** Distância total percorrida pelo estafeta.

5 Funcionamento do Programa

5.1 Grafo

O processo de criação do grafo para o nosso projeto, iniciou-se com a utilização da ferramenta OSMnx, que permite extrair, modelar e analisar redes de ruas de cidades reais do OpenStreetMap. Este processo envolveu a seleção de uma área geográfica específica, a cidade de Braga, e a obtenção da rede de ruas e estradas associadas a essa área. Após a obtenção dos dados de rua do OpenStreetMap através do OSMnx, procedemos à conversão desses dados em um formato mais adaptável e compatível com o nosso sistema. Este passo implicou a exportação dos dados para ficheiros CSV, que contêm informações detalhadas sobre os nós (interseções de ruas) e as arestas (segmentos de rua).

5.1.1 Representação do Grafo

O grafo foi representado através de duas classes principais: **Node** e **Edge**. Cada instância da classe **Node** representa um node no grafo, identificado por um **node_id** único e suas coordenadas geográficas. Por outro lado, a classe **Edge** representa as arestas do grafo, cada uma ligando dois nodes (**u** e **v**). As arestas incluem informações sobre se a via é de sentido único (**oneway**), o comprimento da via (**length**), a geometria da via (uma representação das coordenadas do caminho, que será útil para o desenho do grafo) e o nome da rua (**name**).

A classe **Graph** encapsula a lógica para construir o grafo. Ela mantém dois dicionários: um para os nodes (**nodes**) e outro para as arestas (**edges**)

5.1.2 Menu Grafo:

Iremos também ter o seguinte menu para que o utilizador possa interagir com o grafo

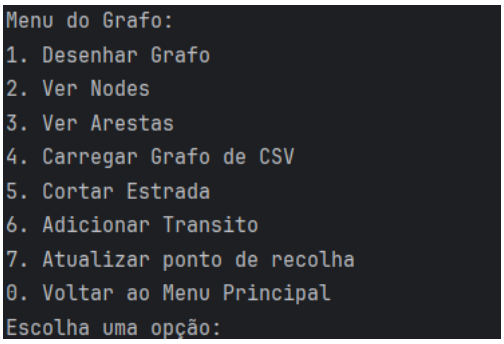

Figura 1 Menu do grafo

1. **Desenhar Grafo** – Esta opção utiliza a biblioteca Plotly para visualizar graficamente o grafo. Cada nodo e aresta é representado, permitindo uma compreensão visual clara dos caminhos e interseções. O nodo inicial (ponto de recolha) é destacado, facilitando a identificação do ponto de partida das entregas.
2. **Ver Nodes** – Exibe os nodes do grafo.
3. **Ver Arestas** –Exibe as arestas dos grafos.

4. **Carregar Grafo de CSV** – Carregar um grafo de um csv, para isso o utilizador terá de indicar o caminho para o csv dos nodes e para o csv das edges.
5. **Cortar Estrada** – Possibilita simular situações de estradas bloqueadas, removendo uma aresta específica do grafo.
6. **Adicionar Transito** – Permite aumentar o custo (distância) de uma aresta específica para simular condições de tráfego.
7. **Atualizar ponto de recolha** – Permite ao utilizador indicar o ponto de recolha das entregas.

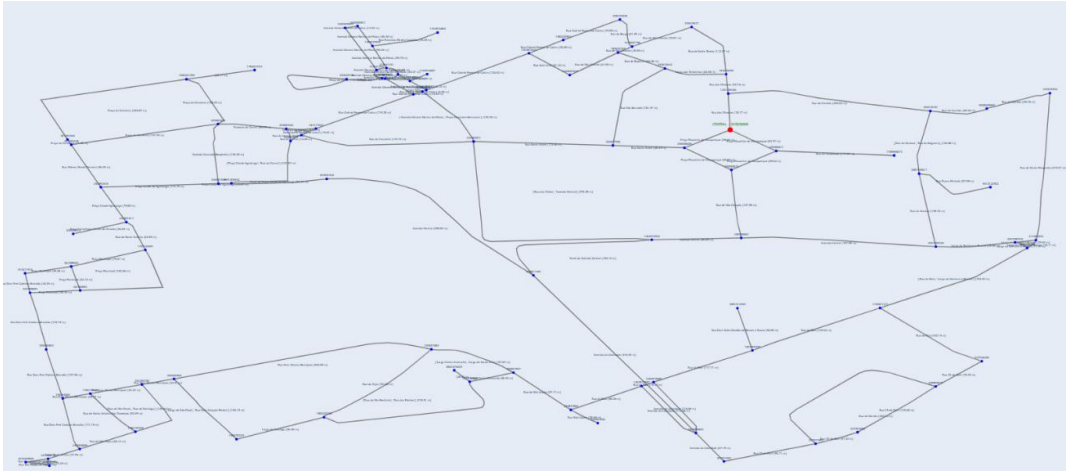

Figura 2 Exemplo de um grafo

5.2 Menu Principal

De forma ao utilizador poder interagir com o nosso sistema, foi criado o seguinte menu

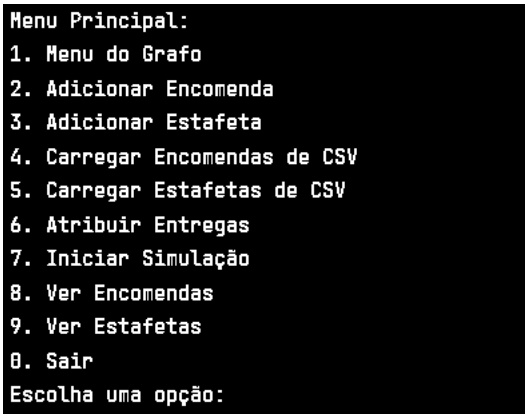

Figura 3 Menu Principal

1. **Menu do Grafo** – Permite ao utilizador aceder ao menu do grafo.
2. **Adicionar Encomenda** – Permite o registo de novas encomendas no sistema, especificando detalhes como ponto de entrega, destino, prazo, peso e volume. Para indicar o ponto de entrega o utilizador pode escolher entre digitar o mesmo ou então selecionar num grafo iterativo o node pretendido.

3. **Adicionar Estafeta** – Esta funcionalidade é usada para adicionar novos estafetas ao sistema, definindo características como tipo de transporte, capacidade de carga e velocidade base.
4. **Carregar Encomendas de CSV** – Permite ao utilizador carregar encomendas através de arquivos csv.
5. **Carregar Estafetas de CSV** – Permite ao utilizador carregar estafetas através de arquivos csv.
6. **Atribuir Entregas** – Distribui as entregas pelos vários estafetas (vai ser explicado mais a frente como isto é feito).
7. **Iniciar Simulação** – Inicia uma simulação (vai ser explicado mais a frente como esta simulação se processa).
8. **Ver Encomendas** – Permite ao utilizador ver as encomendas, juntamente com os seus detalhes como status, avaliação, preço, …
9. **Ver Estafetas** – Permite ao utilizador ver os estafetas, assim como os seus atributos.

5.3 Distribuição de Entregas pelos Estafetas

A distribuição das entregas pelos estafetas é uma etapa crucial no nosso sistema de entrega, que visa otimizar a eficiência e a sustentabilidade. Esta distribuição é feita pela função **allocate_deliveries_to_couriers** presente no ficheiro DeliveryService.py.

O processo é realizado da seguinte forma:

1. **Ordenação das Entregas**: Inicialmente, as entregas são ordenadas com base nos seus prazos de entrega. Esta ordenação prioriza entregas com prazos mais iminentes, garantindo que as encomendas de caráter mais urgente sejam atribuídas em primeiro lugar.

2. **Seleção de Estafetas**: Para cada entrega, o sistema percorre a lista de estafetas disponíveis. A escolha do estafeta adequado para cada entrega é baseada em dois critérios principais:

    - **Capacidade de Aceitar a Entrega**: O estafeta deve ter capacidade (em termos de carga máxima) para aceitar a entrega.

    - **Pontuação de Compatibilidade**: Calcula-se uma pontuação de compatibilidade para cada estafeta, considerando o impacto ecológico e a eficiência na entrega.

Cálculo da Pontuação de Compatibilidade

A pontuação de compatibilidade entre um estafeta e uma entrega é calculada da seguinte forma:

1. **Tempo Estimado e Impacto Ecológico**: Primeiramente, calcula-se o tempo estimado de entrega e o impacto ecológico para o estafeta considerado. O impacto ecológico é avaliado com base no tipo de transporte utilizado pelo estafeta e a distância total a ser percorrida (esta distância não vai ser a real, vai ser apenas uma estimativa utilizando a heurística do grafo).

2. **Fator de Prazo**: Calcula-se um fator de prazo, que mede a adequação do tempo estimado de entrega em relação ao prazo da entrega.

3. **Cálculo da Pontuação**: A pontuação final é uma combinação ponderada do impacto ecológico e do fator de prazo. Valores menores indicam melhor compatibilidade.

Decisão Final

Após calcular a pontuação de compatibilidade para cada estafeta, o estafeta com a menor pontuação é selecionado para a entrega. Em caso de empate, o estafeta com a melhor avaliação (score) é escolhido. Uma vez selecionado, o estafeta é atribuído à entrega, e o status da entrega é atualizado para 'Atribuída'.

5.4 Cálculo dos caminhos para os estafetas:

O cálculo dos caminhos de cada estafeta, de modo aos mesmos poderem entregar as suas encomendas é feito da seguinte forma:

1. **Inicialização:**
    - Para cada estafeta, a função **calculate_route_for_courier** calcula caminhos utilizando os algoritmos de procura (BFS, DFS, Greedy, A*).
    - Cada encomenda associada ao estafeta tem um tempo de entrega estimado calculado para cada algoritmo.
2. **Execução dos Algoritmos:**
    - A função itera sobre os algoritmos de procura aplicando-os para encontrar o caminho entre o ponto atual do estafeta e o destino da encomenda.
    - O custo total, tempo de execução, espaço utilizado e o caminho percorrido são calculados para cada algoritmo.
3. **Seleção do Melhor Caminho:**
    - Após executar todos os algoritmos, a função determina qual deles oferece o caminho com o menor custo total.
    - Este algoritmo é então selecionado como o melhor para o estafeta em questão.
4. **Atribuição de Tempo de Entrega:**
    - Cada encomenda é atualizada com o tempo de entrega estimado com base no algoritmo selecionado como melhor.
5. **Visualização do Caminho:**
    - Os caminhos percorridos pelos algoritmos são desenhados num grafo, mostrando também o melhor caminho.

5.5 Simulação

A simulação de entrega de encomendas é um componente crítico na avaliação da eficácia dos algoritmos de procura implementados. Este processo é realizado da seguinte forma:

1. Execução da Simulação:

- A função **simulate** é responsável por efetuar o cálculo dos percursos para todos os estafetas. Para isso calcula os caminhos para todos os estafetas utilizando a função **calculate_route_for_courier** e regista os dados referentes aos custos, tempos e espaço utilizados por cada algoritmo de procura.

2. Análise Comparativa:

- Os resultados obtidos são submetidos a uma análise comparativa para determinar a eficiência de cada algoritmo. Para ilustrar as discrepâncias entre os algoritmos no que concerne a custos, tempo e espaço, elabora-se um gráfico de barras.

3. Atualização das Encomendas e Estafetas:

- Cada encomenda é atualizada com o seu estado final, marcado como "concluída", e o preço é calculado tendo em conta o meio de transporte empregue.
- É concedida ao utilizador a oportunidade de avaliar a encomenda, sendo-lhe exibidos detalhes pertinentes, como o tempo de entrega e o preço.
- Os estafetas recebem atualizações nos seus scores, as quais são baseadas nas avaliações efetuadas pelos clientes.

Uma das funcionalidades da nossa simulação é aplicar trânsito nas ruas do nosso grafo, este trânsito terá em conta o veículo do estafeta, sendo que as bicicletas não serão afetadas pelo trânsito, e os carros serão mais afetados que as motos.

6 Estratégias de Procura Implementadas

6.1 Procura em Largura (BFS)

A estratégia de procura em largura expande inicialmente todos os nodos de menor profundidade no grafo.

Vantagens:

- Garante a descoberta do caminho mais curto se os custos das arestas forem iguais.
- É completa, ou seja, encontra uma solução se ela existir.

Desvantagens:

- Pode ser ineficiente em termos de tempo e espaço, especialmente em grafos grandes, pois mantém todos os nodos fronteira na memória.
- Em termos de tempo, pode ser lento em grafos muito grandes ou densos.

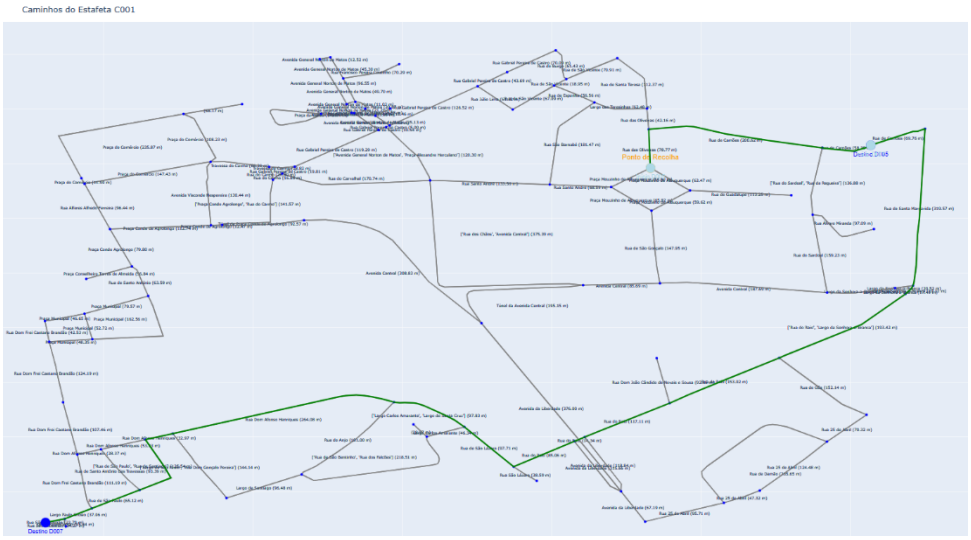

Figura 4 Exemplo de procura em largura

6.2 Procura em Profundidade (DFS)

Esta estratégia expande sempre os nós mais profundos do grafo em primeiro lugar.

Vantagens:

- Utiliza menos memória do que o BFS, pois não precisa armazenar todos os nós de fronteira.
- Pode ser mais rápido que o BFS em alguns casos, especialmente em grafos onde a solução está muito profunda.

Desvantagens:

- Não garante o caminho mais curto.
- Pode ser ineficiente em grafos muito profundos ou em casos onde não existe solução.
- Suscetível a ficar preso em ciclos (embora a nossa implementação evite isso através da verificação de nós já visitados).

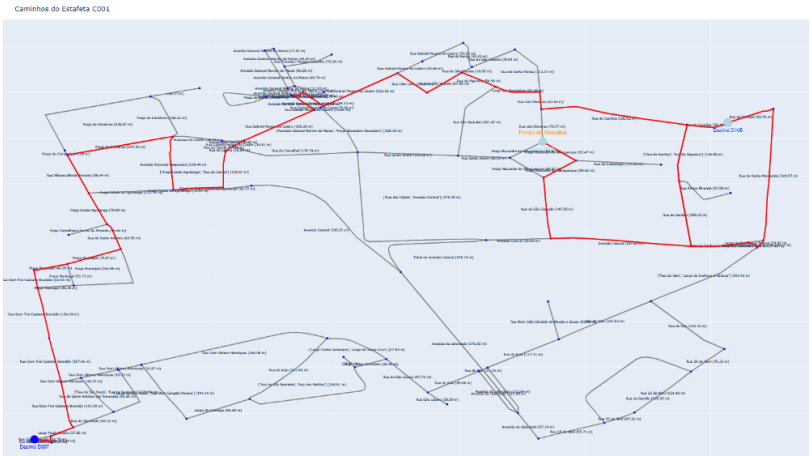

Figura 5 Exemplo de procura em profundidade

6.3 Procura Gulosa (Greedy-Search)

Este algoritmo seleciona o próximo passo com base numa heurística que estima a melhor opção para se aproximar do objetivo, sem considerar o custo acumulado até ao momento.

Vantagens:

- Rápido para encontrar uma solução, mesmo que não seja a ótima, em muitos casos.
- Particularmente eficaz em problemas onde a heurística é bem representativa da distância ao objetivo.

Desvantagens:

- Não garante a solução mais ótima, podendo ser enganado por caminhos que parecem promissores, mas que não levam à solução.

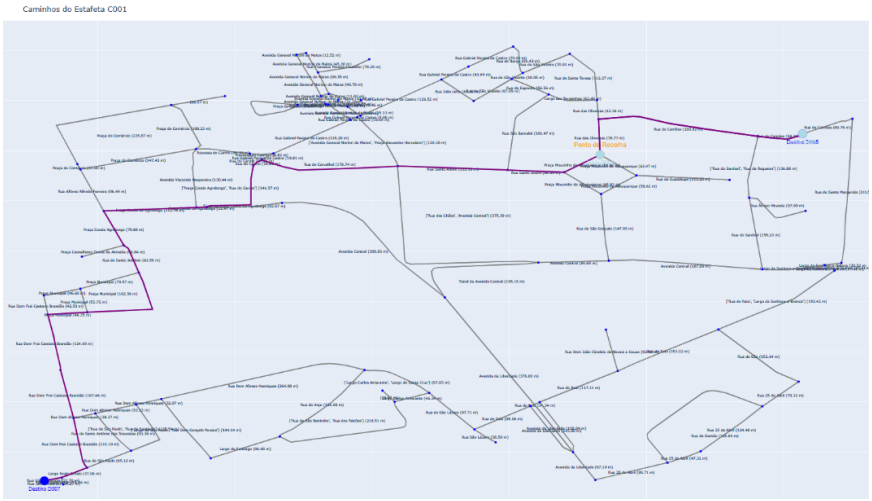

Figura 6 Exemplo de procura utilizando o Greedy

6.4 Procura A*

Este algoritmo combina características da procura gulosa e da procura de custo uniforme. Utiliza uma função **f(n) = g(n) + h(n)**, onde **g(n)** é o custo do caminho do nó inicial até **n**, e **h(n)** é o custo estimado de **n** até o objetivo. Irá assim selecionar o caminho que minimiza **f(n)**.

Vantagens:

- Eficiente e eficaz na localização do caminho mais curto.
- Garante a solução ótima se a heurística for admissível (nunca superestima o custo real até ao objetivo).
- É uma das melhores escolhas para muitos problemas de procura no caminho mais curto.

Desvantagens:

- A eficácia depende da qualidade da heurística.
- Pode ser ineficiente em termos de memória e tempo se a heurística não for bem escolhida ou se o espaço de procura for muito grande.

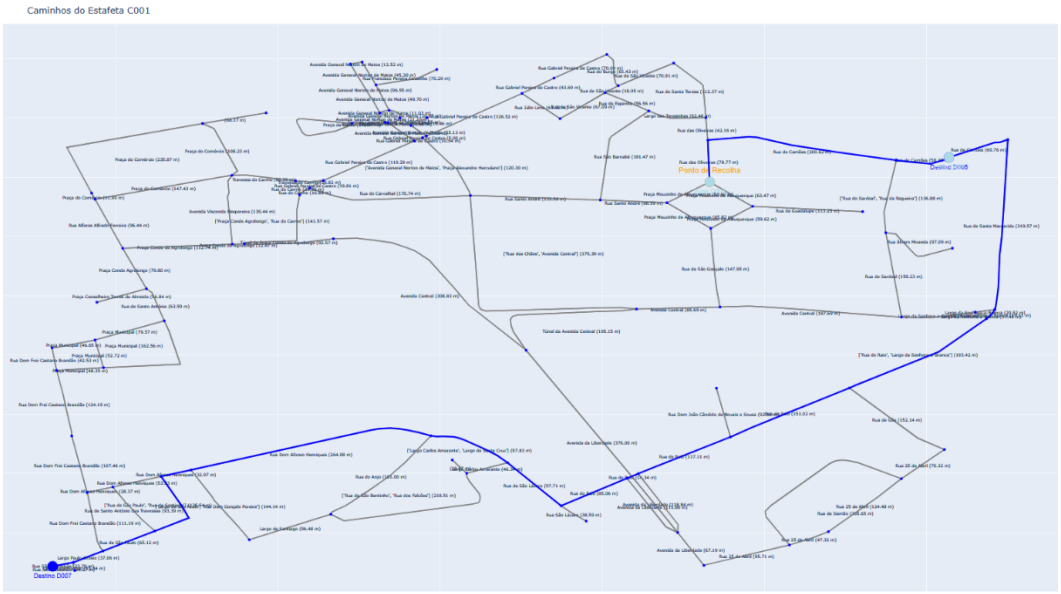

Figura 7 Exemplo de procura utilizando o A*

6.5 Heurística

A heurística adotada no nosso sistema é a Fórmula de Haversine. Esta fórmula é utilizada para calcular a distância geodésica, isto é, a distância em linha reta, entre dois pontos na superfície terrestre, com base nas suas coordenadas de latitude e longitude.

A Fórmula de Haversine revela-se particularmente apropriada para o cálculo de distâncias no âmbito da distribuição de encomendas. Ela fornece uma estimativa precisa da distância efetiva entre dois pontos, o que é fundamental para o planeamento eficiente dos caminhos. Esta heurística assume um papel crucial em algoritmos de procura informada, como o A* e a Busca Gulosa, nos quais o objetivo primordial é minimizar o custo total do percurso. Ao oferecer uma estimativa do custo necessário para alcançar o objetivo (neste caso, a distância até o ponto de entrega), a heurística permite que o algoritmo faça escolhas estratégicas, priorizando trajetos que aparentem conduzir de forma mais rápida ao destino.

Importa salientar que a heurística de Haversine é admissível e consistente para o problema em análise. Ela nunca superestima a distância real entre dois pontos, aspeto crucial para assegurar que os algoritmos de procura informada identifiquem o caminho ótimo.

7 Análise de Resultados

Nesta secção iremos analisar o desempenho dos vários algoritmos. Para isso teremos em conta parâmetros como o custo da solução, o tempo de execução e o espaço (calculado tendo em conta as estruturas de cada algoritmo) utilizado por cada algoritmo. Importante destacar que o cálculo do tempo pode não ser o mais preciso, pois pode ser influenciado por fatores externos como a carga do sistema operativo no momento da execução do algoritmo.

Para podermos comparar o desempenho de forma justa, foram utilizados dois grafos, um grafo com pequeno com cerca de 100 nodes, e um grafo grande com cerca de 5000 nodes.

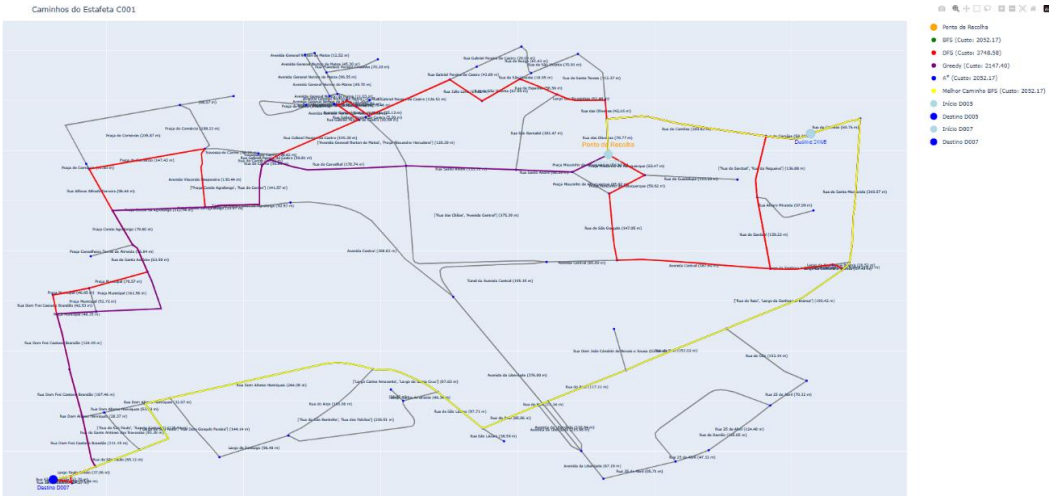

Figura 8 Exemplo de caminhos obtidos pelos algoritmos num grafo pequeno

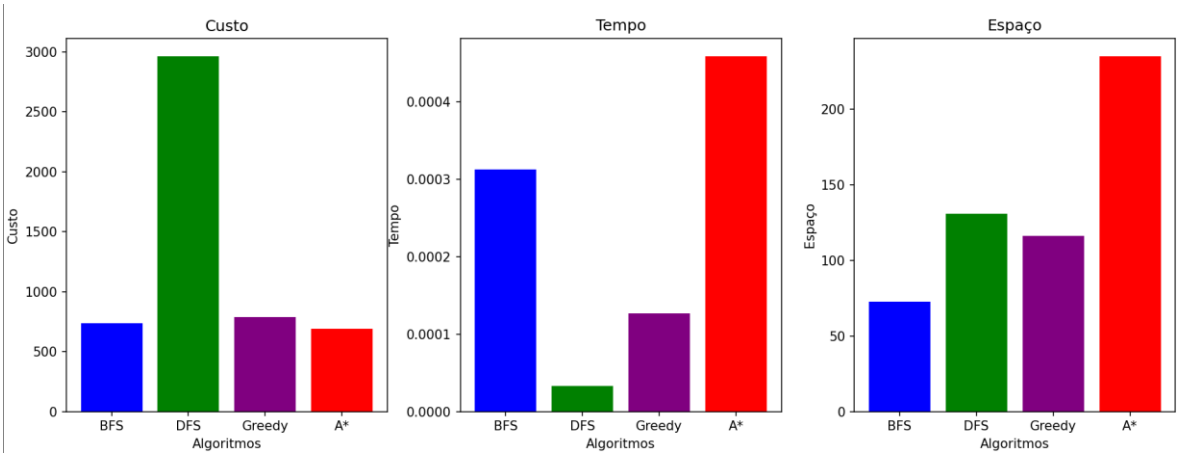

Figura 9 Resultados obtidos para grafo pequeno

Observando os resultados obtidos para o grafo pequeno, podemos verificar que em termos de custo o DFS tem o pior desempenho, o que já era esperado considerando a sua natureza de exploração em profundidade sem ter em conta as heurísticas ou o custo acumulado. Por outro lado, o A* demonstrou ser o mais eficaz em termos de custo, dado que este algoritmo integra tanto o custo já acumulado quanto a estimativa heurística para o objetivo, levando a uma procura mais direcionada e eficiente.

Apesar de não assegurar a solução ótima, o BFS também registou um bom desempenho, uma vez que explora de forma uniforme em todas as direções a partir do ponto inicial, sendo provável que encontre um caminho mais curto quando as entregas se situam relativamente próximas. O algoritmo Greedy, por sua vez, também conseguiu identificar uma solução próxima da ótima.

Em termos de tempo o DFS foi o mais rápido a encontrar uma solução, mesmo que esta não seja a solução ótima. Por outro lado, o A* e o BFS foram os mais ineficientes.

Já em termos de espaço, podemos verificar que para grafos pequenos onde os pontos de entrega estão bastante próximo do node inicial, o BFS destaca-se. O A* foi o pior algoritmo em termos

de espaço porque precisa de armazenar simultaneamente as múltiplas fronteiras de nós não explorados e a sua informação associada na heap.

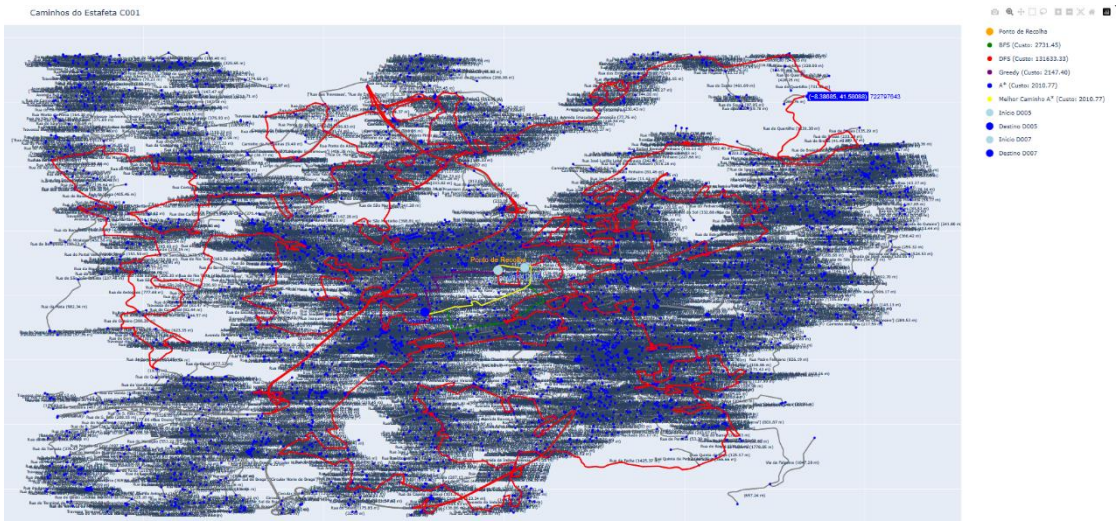

Figura 10 Exemplo de caminhos obtidos pelos algoritmos num grafo grande

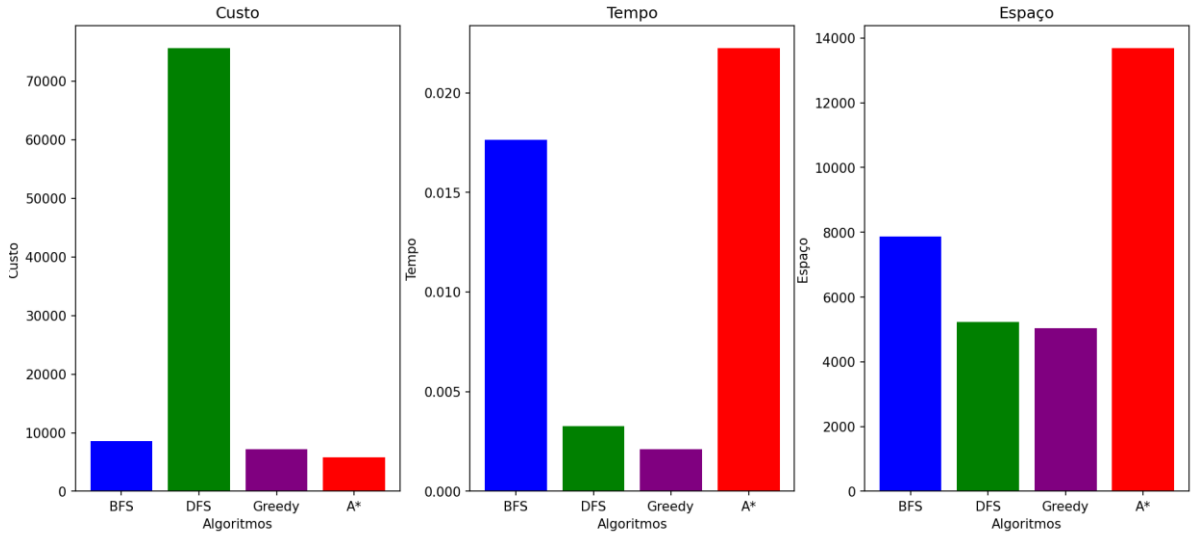

Figura 11 Resultados obtidos para grafos grandes

Observando agora o desempenho para grafos de grande dimensão, constata-se que o DFS se mantém como o menos eficiente em termos de custo, enquanto o A* continua a destacar-se pela sua eficiência.

Em termos de tempo, podemos verificar que o tempo de execução aumentou para o BFS, o que nos leva a concluir que o tempo deste algoritmo aumenta significativamente com o tamanho do grafo. No entanto, o A* continua a ser o mais lento.

Relativamente ao uso de espaço, tanto o DFS como o Greedy apresentam os melhores desempenhos. Verifica-se um acréscimo considerável no uso de espaço pelo BFS, tornando-o agora menos eficiente neste parâmetro do que o DFS e o Greedy. O A*, por sua vez, continua a ser o mais ineficiente em termos de espaço.

Após uma análise dos vários algoritmos concluímos que os algoritmos de procura informada (Greedy e A*) são mais eficientes na identificação da solução ótima. Considerando as especificidades do nosso problema, o algoritmo A* revela-se o mais adequado. Apesar de não ser o mais eficiente quanto ao tempo de execução e complexidade espacial, este algoritmo oferece a solução de menor custo, o que é fundamental para minimizar o impacto ecológico e assegurar o cumprimento dos prazos de entrega.

8 Conclusão

Em resumo, no âmbito deste projeto, desenvolvemos um sistema destinado à distribuição de encomendas por estafetas. Este sistema leva em consideração diversos fatores, tais como os prazos de entrega estipulados, o peso das encomendas e a pegada ecológica associada a cada veículo utilizado pelos estafetas. Para tal, foi construído um modelo de grafo representativo de uma cidade, baseando-nos em dados reais para estabelecer um sistema eficaz de pontos de entrega.

No decorrer do projeto, implementámos uma variedade de algoritmos de procura, abordando tanto os informados como os não informados, com o objetivo de definir os percursos mais eficientes para os estafetas. A comparação entre estes algoritmos baseou-se em três parâmetros fundamentais: custo, tempo e uso de espaço.

Este trabalho proporcionou-nos uma compreensão aprofundada sobre o funcionamento de cada algoritmo de procura, permitindo-nos assim identificar os aspetos positivos e negativos inerentes a cada um deles, e saber qual deles é mais adequado para o nosso problema.