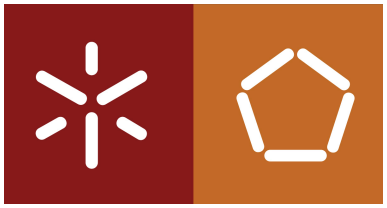

Inteligência Artificial - Instrumento de Avaliação em Grupo

UberUM

11.01.2026

Grupo 5: Gabriel Dantas (a107291), José Fernandes (a106937), Luis Ferreira (a98286), Simão Oliveira(a107322)

Universidade do Minho - Licenciatura em Engenharia Informática, Inteligência Artificial

Índice

1. Avaliação pelos pares

Da análise coletiva, concluiu-se que todos os discentes do grupo contribuíram e esforçaram-se de maneira similar e equilibrada. Dessa maneira, atribuir-se-ão os valores de delta, δ, da mesma maneira por todos os integrantes:

- Gabriel Dantas (A107291), $\delta = 0.00$;
- José Fernandes (A106937), $\delta = 0.00$;
- Luis Ferreira (A98286), $\delta = 0.00$;
- Simão Oliveira(A107322), $\delta = 0.00$.

2. Introdução

O presente relatório tem como objetivo documentar o desenvolvimento do trabalho prático da unidade curricular de Inteligência Artificial. Este trabalho incide sobre a resolução de problemas através de algoritmos de procura, aplicando-os a um cenário realista de gestão e otimização de uma frota de táxis heterogénea.

Ao longo do desenvolvimento do projeto, formulou-se o problema de forma adequada, representou-se o ambiente sob a forma de grafo e desenvolveu-se diferentes estratégias de procura, tanto informadas como não informadas, avaliando o seu desempenho face a diversos critérios, como custo operacional, tempo de resposta e sustentabilidade ambiental. Este relatório descreve as principais decisões tomadas, a abordagem seguida e os resultados obtidos durante a implementação e simulação do sistema proposto.

3. Descrição do problema

As empresas de transporte urbano enfrentam, atualmente, desafios significativos na gestão eficiente das suas frotas, resultantes do aumento da procura por serviços rápidos, da necessidade de redução de custos operacionais e da crescente preocupação com a sustentabilidade ambiental. Neste contexto, a empresa **UberUM** pretende otimizar a gestão da sua frota de táxis urbanos, composta por veículos a combustão e veículos elétricos, com características e limitações distintas.

Cada veículo da frota possui atributos específicos, nomeadamente o tipo de motorização (elétrico, combustão ou híbrido), a autonomia média, mínima e máxima, a capacidade de passageiros, a localização atual e o seu estado de disponibilidade. Em particular, os veículos elétricos requerem recargas periódicas em estações específicas que demoram aproximadamente 30 minutos, enquanto os veículos a combustão permitem reabastecimentos que apresentam um custo temporal médio de 6 minutos, mas acarretam custos operacionais superiores e maior impacto ambiental.

Os pedidos de transporte surgem de forma dinâmica ao longo do tempo e incluem informação como a localização de origem e de destino, o número de passageiros, o horário pretendido para recolha, o nível de prioridade e a possibilidade de partilha do veículo com passageiros de outros pedidos. Em determinados

casos são ainda indicadas preferências ambientais por parte dos clientes. Assim, o sistema deve ser capaz de alocar, em tempo útil, o veículo mais adequado a cada pedido, garantindo o cumprimento das restrições existentes e maximizando a eficiência global da frota.

O problema consiste, portanto, em desenvolver uma estratégia inteligente de alocação e gestão de veículos que permita satisfazer o maior número possível de pedidos, minimizando simultaneamente o tempo de resposta, os custos operacionais e o impacto ambiental. Para tal, é necessário considerar fatores dinâmicos, como variações no trânsito, alterações na disponibilidade dos veículos e diferentes climas em certas localizações.

4. Formulação do Problema como Problema de Procura

Este problema pode ser classificado como um problema de otimização de percursos com múltiplas restrições e também um problema de procura em grafos, especificamente um problema de caminho mínimo com restrições. Pode ser classificado quanto:

- ao ambiente, sendo Dinâmico, Observável, Determinístico, Multi-agente e Sequencial;
- à natureza da solução, de Caminho (a solução é uma sequência de ações que nos levam ao objetivo),Otimização (procura-se a melhor solução possível), com restrições temporais.

4.1. Estado

O estado do sistema pode ser representado por uma tupla: $S = \langle T, V, R, E, C, G \rangle$, onde:

- T: Tempo - representa o instante atual da simulação:
  - Tempo decorrido (em minutos/segundos);
  - Passo da simulação (step) atual.
- V: Veículos - lista de tuplas que representa a frota. Contêm:
  - Identificador: ID único do veículo (int);
  - Tipo de Motorização: Elétrico, Combustão ou Híbrido;
  - Posição Atual: Coordenadas cartesianas (x,y) ou ID do nó no grafo;
  - Energia:
    - Nível de Bateria atual e máxima (kWh) (para Elétricos/Híbridos);
    - Nível de Combustível atual e máximo (Litros) (para Combustão/Híbridos);
  - Capacidade: Número máximo de passageiros suportados;
  - Condutor: Nome/ID do condutor;
  - Estado: `stopped` (parado), `moving_to_pickup` (em movimento para ir buscar), `moving_to_dropoff` (em movimento para deixar), `waiting` (em espera), `charging` (a carregar), `refueling` (em reabastecimento);
  - Velocidade: Velocidade média dependente da via onde circula.

- R: Pedidos (Requests) - lista de tuplas representando as solicitações de transporte, contendo:
  - Identificador: ID único do pedido;
  - Origem e Destino: Posições (x,y) de recolha e entrega;
  - Passageiros: Quantidade de pessoas a transportar;
  - Estado: `pending` (pendente), `picked_up` (em viagem), `completed` (concluído);
  - Tempo: Hora da solicitação;
  - Premium : Se o utilizador é *premium* ou não;
  - Preferência ambiental.
- E: Estações e Infraestrutura - lista de locais de serviço:
  - Estações de Carregamento: Localização e capacidade de atendimento;
  - Bombas de Combustível: Localização e capacidade;
  - Garagem (Depot): Pontos de início/fim de turno.
- C: Condições e Eventos - Fatores dinâmicos que alteram o grafo:
  - Eventos Climáticos: Condições (ex: chuva) que podem reduzir a velocidade média;
  - Bloqueios: Estradas temporariamente fechadas (`open`: False);
  - Trânsito: Fatores de congestionamento (`traffic_factor`) aplicados às arestas.
- G: Geografia e Navegabilidade - O Grafo que representa a rede viária de Gualtar:
  - Nós: Interseções e locais de interesse com coordenadas projetadas em metros;
  - Arestas: Segmentos de estrada contendo:
    - Distância real (metros);
    - Tempo que a aresta demora ser percorrida (já afetado por fatores de trânsito e clima);
    - Tempo de percurso estimado (influenciado por condições climáticas e trânsito);
    - Direcionalidade (bidirecional ou sentido único).

4.2. Estado Inicial

O estado inicial corresponde à configuração do sistema no início da simulação, na qual todos os veículos se encontram posicionados na estação da empresa UberUM localizada no Oeste da freguesia de Gualtar, com autonomia inicial definida e sem pedidos atribuídos. Os pedidos de transporte iniciais são considerados vazios, sendo que novos pedidos são introduzidos dinamicamente ao longo do tempo.

Logo temos os seguintes valores no estado inicial:

- T = 0;
- V, Veículos:
  - Posição atual : Garagem da empresa UberUM;
  - Energia: Bateria e/ou combustível totalmente abastecidos;
  - Estado : `waiting`.
- R, Pedidos:

  - Ainda nenhum pedido chegou ao sistema.
- E, Estações e Infraestrutura:
  - Definidas antes do inicio da simulação.
- C, Condições e Eventos:
  - Apenas se encontram ativas as condições e eventos cujo inicio foi definido ás 08 : 00h.

4.3. Operadores

Os operadores representam as ações possíveis que podem ser aplicadas a um estado. Entre os principais operadores considerados encontram-se os seguintes.

4.3.1. Atribuição de um pedido de transporte a um veículo

Esta operação é responsável por atribuir um pedido de transporte ainda não atribuído a um táxi disponível na frota.

Operação	Atribuição de um pedido a um táxi com disponibilidade.
Pré-condição	• O instante temporal do pedido tem de ser anterior ou igual ao tempo atual da simulação (não são processados pedidos que ainda não tenham ocorrido). • O pedido tem que estar no estado de `pending` ; • O local de recolha tem de ser alcançável pelos veículos disponíveis.
Efeitos	• Caso exista, pelo menos, um veículo disponível e que cumpra os requisitos do pedido, este inicia a deslocação para o ponto de recolha. • Se o veículo não possuir autonomia suficiente, desloca-se primeiro para um posto de abastecimento ou recarga. • O estado do pedido é atualizado para `assigned`.

4.3.2. Deslocar

Move um veículo da sua posição atual para um destino viável, respeitando as restrições de acessibilidade, autonomia e condições climatéricas.

Operação	Deslocação de um veículo da posição atual para um destino viável.
Pré-condição	• O veículo encontra-se no estado disponível; • O destino é acessível ao veículo; • O nível de combustível do veículo é suficiente para cobrir a distância até ao destino, considerando eventuais penalizações decorrentes das condições climatéricas (chuva, neve, etc.).
Efeitos	• A posição do veículo é atualizada para o destino;

	• O nível de combustível do veículo é reduzido em função da distância percorrida e das condições climatéricas; • Caso o destino corresponda a um ponto de recolha/entrega ou a um posto de abastecimento/recarga, podem ser realizadas, respetivamente, operações de recolha ou entrega de clientes e de reabastecimento do veículo.

4.3.3. Recolher um passageiro na localização de origem

Efetua a recolha de um passageiro no local de origem do respetivo pedido, marcando o início da fase de transporte do cliente até ao seu destino final.

Operação	Recolher um passageiro no local de origem do seu pedido.
Pré-condição	• O veículo possui exatamente um pedido atribuído; • O veículo que efetua a recolha é o mesmo que se encontra alocado ao pedido; • O veículo encontra-se na localização de origem do pedido; • O pedido encontra-se no estado `assigned`.
Efeitos	• O passageiro é recolhido pelo veículo; • O veículo inicia a deslocação para o destino do cliente; • O estado do pedido é atualizado para `picked_up`.

4.3.4. Entregar um passageiro no destino

Efetua a entrega do passageiro no destino final definido no seu pedido, concluindo o serviço de transporte.

Operação	Entregar um passageiro no destino definido no seu pedido.
Pré-condição	• O veículo possui um pedido ativo; • O veículo que efetua a entrega é o mesmo que se encontra alocado ao pedido; • O veículo encontra-se na localização de destino do pedido; • O pedido encontra-se no estado `picked_up`.
Efeitos	• O passageiro é entregue no destino; • O pedido é concluído, passando para o estado `completed` ; • O pedido deixa de estar associado ao veículo.

4.3.5. Reabastecimento/Recarga de veículo

Efetua o reabastecimento de combustível ou a recarga de energia de um veículo num posto apropriado, aumentando a sua autonomia.

Operação	Restabelecer o combustível ou recarregar a energia de um veículo.

Pré-condição	• O veículo encontra-se num posto de reabastecimento ou recarga compatível com o seu tipo de energia; • O veículo não se encontra a transportar passageiros; • O nível de combustível/energia do veículo é inferior à sua capacidade máxima.
Efeitos	• O nível de combustível/energia do veículo é aumentado até ao valor máximo; • A autonomia do veículo é atualizada de acordo com o novo nível de energia; • O veículo mantém-se ou transita para o estado disponível, caso não exista um pedido ativo.

4.4. Custo da solução

O custo da solução é variável e depende de fatores como a distância percorrida, o tempo de deslocação total, os custos operacionais, o tempo de espera do clientes e o impacto ambiental associado ao tipo de veículo utilizado. A distância percorrida aumenta os gastos proporcionalmente, considerando rotas mais longas, alterações no percurso, e ainda desvios causados por estradas cortadas ou inacessíveis. O tempo que se demora a chegar ao ponto de recolha de um cliente também é importante, e este é afetado por fatores como condições climatéricas adversas (Chuva, Chuva forte, Neve ou Vento forte). Já o custo pode ser influenciado pelo tipo de veículo que estamos a utilizar (elétrico ou combustão), um carro elétrico possui um menor custo de recarregamento , mas este demora 30 minutos até ficar totalmente carregado, já um veículo a combustão possui um maior custo para ser reabastecido, mas fica totalmente abastecido em 6 minutos.

O objetivo principal é minimizar o custo total da solução, promovendo simultaneamente a eficiência operacional e a sustentabilidade ambiental.

Podemos utilizar a seguinte fórmula para determinar o custo da solução:

$$C = \alpha \times F + \beta \times T + \varepsilon \times R + \theta \times D + \delta \times A$$

- C: Custo total a ser minimizado;
- F: Custo de combustível total (em litros ou equivalente energético);
- T: Tempo total;
- R: Pedidos que não puderam ser atendidos;
- D: Distância total percorrida;
- A: Impacto ambiental;
- $\alpha, \beta, \varepsilon, \theta, \delta$: Pesos que representam a importância relativa de cada componente no problema.

5. Estrutura

O projeto é estruturado com várias classes, cada uma desempenha um papel específico e importante na simulação.

5.1. Estrutura do Grafo

Para representar a cidade onde a empresa UberUM opera, utilizou-se um grafo bidirecional limitado, para simplificação, à freguesia de Gualtar, em Braga. A construção do grafo foi realizada com recurso à biblioteca `OSMnx`, que permite carregar e manipular dados geoespaciais reais provenientes do `OpenStreetMap`.

Cada nodo do grafo (classe `Node`) representa um cruzamento e possui:

- Um ID único incremental;
- O tipo de nodo, quando aplicável (por exemplo, posto de abastecimento ou de carregamento);
- A sua posição no mapa, definida pelos dados do `OSMnx`.

Durante a conversão para a estrutura interna `Graph`, é aplicado um algoritmo de simplificação que funde nós geograficamente redundantes. Se dois ou mais nós estiverem a uma distância inferior a um limiar pré-definido (por defeito, 100 metros), estes são aglutinados num único vértice, reduzindo a complexidade do grafo e eliminando micro-interseções irrelevantes para a simulação.

As arestas representam as estradas que ligam os cruzamentos e incluem:

- Um ID único;
- A distância total da estrada;
- O tempo necessário para percorrer a aresta, calculado como a distância dividida pela velocidade da aresta (definida pelo `OSMnx` com base nos limites de velocidade reais, ou assumida como 50 km/h por defeito), multiplicada por um fator de trânsito (onde 1 corresponde a tráfego normal) e um fator de clima;
- O estado da estrada (aberta ou fechada).

5.2. Eventos

Os tempos necessários para percorrer cada aresta são influenciados por dois fatores: trânsito e condições climáticas, ambos estão armazenados na classe `events`:

As condições climáticas estão definidas como:

```
CLEAR = 'clear'          # Tempo limpo (sem impacto)
RAIN = 'rain'            # Chuva (aumenta tempo em ~30%)
HEAVY_RAIN = 'heavy_rain'  # Chuva forte (aumenta tempo em ~50%)
FOG = 'fog'              # Nevoeiro (aumenta tempo em ~20%)
SNOW = 'snow'            # Neve (aumenta tempo em ~60%)
```

Estes parâmetros podem ser configurados através do ficheiro `dataset.json`, onde se define o estado inicial do sistema, como abaixo se vê.

```
{
  "location": "Gualtar, Braga, Portugal",
  "fuel_stations": [
```

```
    {
      "node_id": 10,
      "_comment": "Bomba de gasolina na zona central"
    }
  ],
  "charging_stations": [
    {
      "node_id": 5,
      "_comment": "Posto de carregamento na zona sul"
    }
  ],
  "events": {
    "weather_zones": [
      {
        "nodes": [5, 6, 7, 8, 9, 10],
        "condition": "rain",
        "_comment": "Zona centro com chuva"
      }
    ],
    "closed_roads": [
      {
        "from": 12,
        "to": 13,
        "_comment": "Estrada fechada por obras"
      }
    ]
  }
}
```

5.3. Pedido

A classe `Request` representa um pedido efetuado por um cliente. Este pedido contém o ponto de recolha e o ponto de entrega, bem como o instante temporal em que foi requisitado. É importante salientar que, no contexto do nosso programa, este tempo corresponde ao momento em que a viagem é solicitada, sendo o objetivo do sistema recolher o cliente o mais rapidamente possível.

O pedido inclui ainda a indicação de se o cliente aceita partilhar a viagem com outros passageiros, o estado atual do pedido (`pending`, `assigned`, `picked_up`, `completed`), o veículo ao qual o pedido se encontra associado (caso já tenha sido atribuído), se o cliente é premium e eventuais preferências ambientais do cliente.

Os pedidos chegam dinamicamente ao longo do dia. Sempre que um novo pedido é registado, é executada uma procura — baseada no algoritmo de procura previamente escolhido antes do início da simulação — que tem como objetivo atribuir ao pedido o táxi disponível que cumpra os requisitos do cliente e que consiga realizar todo o pedido no menor tempo possível.

5.4. Reabastecimento

Na simulação desenvolvida existem nodos definidos como pontos de abastecimento ou recarga. A classe `refuel_helper` é uma classe auxiliar responsável por realizar diversas operações relacionadas com este processo.

Quando um pedido é alocado a um veículo, o sistema calcula o percurso até ao ponto de recolha e, posteriormente, até ao destino, utilizando o algoritmo de procura previamente selecionado. Durante este processo, é verificado se o veículo consegue realizar todo o pedido sem necessidade de reabastecimento. Considerámos importante evitar situações em que o veículo tenha de reabastecer com clientes no interior, uma vez que tal representa uma má imagem para a empresa.

Caso o veículo não consiga completar o pedido com a autonomia disponível, o sistema determina o ponto de abastecimento mais próximo e adiciona o tempo necessário para esse desvio ao tempo total da viagem.

É importante destacar que este tempo adicional é tido em conta no cálculo do veículo mais rápido a satisfazer um pedido. Por exemplo, considerando dois veículos disponíveis, A e B:

- O veículo A consegue chegar ao cliente em 5 minutos, mas necessita de reabastecer antes de concluir o pedido. No caso de ser um veículo elétrico, isso implica, no mínimo, mais 30 minutos adicionais.
- O veículo B encontra-se a 20 minutos do ponto de recolha, mas não necessita de reabastecer.

Apesar de o veículo A chegar mais rapidamente ao cliente, o tempo total do pedido é significativamente maior. Assim, o sistema escolhe o veículo B para satisfazer o pedido, por garantir uma execução mais eficiente e rápida no seu todo.

Esta abordagem contribui para uma maior fluidez do sistema e para decisões mais coerentes na alocação de veículos aos pedidos.

5.5. Veículos

A classe `vehicle` representa um veículo pertencente à frota da empresa. Os veículos são definidos no ficheiro `dataset.json`, anteriormente referido, e contêm um conjunto abrangente de informações relevantes para a simulação.

Cada veículo possui dados relativos ao condutor (nome), ao tipo de veículo (elétrico, combustão ou híbrido) e ao próprio veículo, incluindo o seu nome, autonomia atual, consumo, autonomia máxima e capacidade total de passageiros (não incluindo o condutor). São ainda armazenadas informações sobre a sua posição atual, o identificador do nodo onde se encontra, o número de passageiros atualmente a bordo e o pedido que o veículo está a atender no momento, caso exista.

Relativamente ao seu funcionamento, cada veículo mantém um estado atual, que pode assumir os seguintes valores: `idle`, `to_pickup`, `to_dropoff`, `to_refuel` ou `refueling`. Adicionalmente, são guardadas diversas informações relacionadas

com o abastecimento, nomeadamente se o veículo necessita de reabastecer, o tipo de reabastecimento associado ao seu tipo de veículo e o tempo restante até à conclusão desse processo.

Por fim, o veículo mantém os percursos que tem planeados em cada instante, incluindo os caminhos a percorrer para chegar ao ponto de recolha de um cliente e para efetuar a respetiva entrega no destino.

No início da simulação, todos os veículos encontram-se na garagem da empresa. À medida que novos pedidos vão surgindo ao longo do dia, os veículos vão sendo alocados aos pedidos de acordo com a sua disponibilidade e com os critérios definidos pelo sistema.

6. Simulação do Ambiente

Para representar o ambiente, optou-se por realizar uma simulação baseada em passos (*steps*). Cada *step* corresponde a um intervalo de tempo configurável pelo utilizador antes do início da simulação, podendo ser de 1, 2 ou 5 minutos. Este tempo representa o tempo que passaria na vida real, a cada *tick* da simulação.

6.1. Funcionamento da Simulação

Durante cada *step*, são executadas sequencialmente as seguintes operações:

1. **Processamento de novos pedidos**: Identificam-se os pedidos (*requests*) que chegaram no intervalo de tempo atual e atribui-se cada um a um veículo disponível da frota. Caso não haja nenhum veículo disponível no momento, o pedido permanece em espera e será reavaliado no *step* seguinte;
2. **Atualização do estado dos veículos**: Todos os veículos têm o seu estado atualizado, incluindo a sua posição no grafo, o nível de combustível ou bateria, e o estado atual (idle, em viagem, a recarregar, etc.);
3. **Atualização das estatísticas**: São recalculadas as métricas do sistema, como número de pedidos atendidos, tempos de espera, distâncias percorridas e consumo de recursos.

6.2. Implementação

A simulação é implementada através do método `step()`, que coordena todas as operações de cada iteração:

```
def step(self):
    """
    Executa um passo (tick) da simulação.
    Returns:
        dict: Informação sobre o passo executado
    """
    if self.is_finished():
        return {'finished': True}
    # Processa novos requests
```

```
    new_requests = self.process_new_requests()
    # Atualiza veículos
    self.update_vehicles()
    # Atualiza estatísticas
    self.update_statistics()
    # Avança tempo usando time_step
    self.current_time += self.time_step
    return {
        'finished': False,
        'time': self.current_time,
        'new_requests': new_requests,
        'stats': self.stats.copy()
    }
```

Este método retorna um dicionário contendo informações sobre o estado atual da simulação, incluindo se esta foi concluída, o tempo atual, os novos pedidos processados e as estatísticas atualizadas.

6.3. Visualização

Para a interface visual e monitorização em tempo real da simulação, o projeto recorre à biblioteca *Matplotlib*, fundamental de `Python` em várias áreas, e a dois dos seus módulos:

Matplotlib.pyplot foi utilizada para a renderização estática e estrutural do grafo. Esta biblioteca é responsável por desenhar a topologia da rede viária (nodos e arestas) e os marcadores dos pontos de interesse (como estações de abastecimento e carregamento) no plano cartesiano. A representação gráfica base inclui:

- Os nodos do grafo como pontos;
- As arestas como linhas que conectam os cruzamentos;
- Identificação visual das estações de combustível e carregamento elétrico;
- Legendas e informações contextuais sobre o estado do sistema.

Matplotlib.animation é encarregue de conferir dinamismo à visualização. Esta ferramenta atualiza ciclicamente a figura gerada pelo `pyplot`, redesenhando as posições dos veículos e o estado dos pedidos a cada *frame*. Isto permite observar em tempo real:

- O movimento dos veículos ao longo das rotas;
- A transição de estados dos pedidos (pendente, em curso, concluído);
- O fluxo de tráfego e a operação da frota de forma animada;
- A evolução temporal das métricas do sistema.

A combinação destas ferramentas possibilita acompanhar visualmente o comportamento do sistema, facilitando a identificação de padrões, gargalos e oportunidades de otimização na gestão da frota.

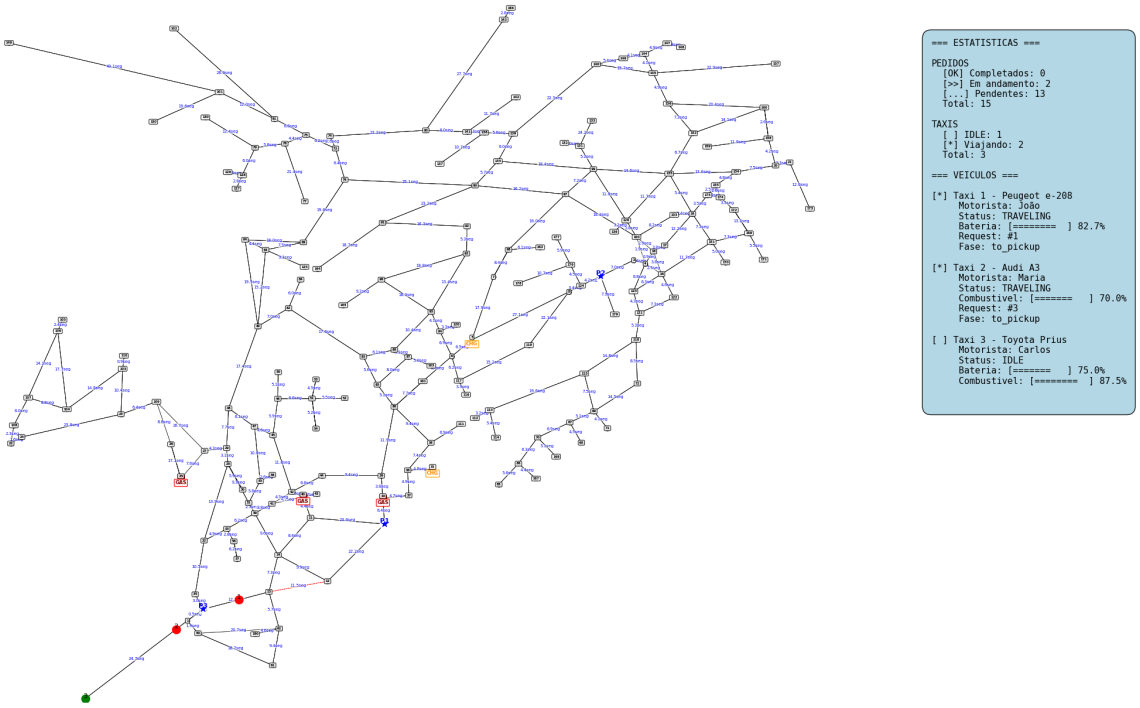

Figura 1: Simulação em curso.

6.4. Atribuição de pedidos a veículos

Para decidir a que veículo deve ser atribuído um pedido, percorrem-se todos os veículos que se encontram disponíveis no momento. Para cada veículo, é calculado o caminho e o tempo necessário para satisfazer o pedido, recorrendo ao algoritmo de procura previamente escolhido. Caso seja necessário, é ainda acrescentado o tempo de recarga ou reabastecimento da viatura, que inclui o desvio até ao posto, o tempo de abastecimento e o percurso do posto até ao ponto de recolha do pedido.

Em pseudocódigo, o processo pode ser descrito da seguinte forma:

```
for pedido in novosPedidos:
  melhorVeiculo = null
  melhorTempo = infinito

  veiculosDisponiveis = getVeiculosDisponiveis()

  for veiculo in veiculosDisponiveis:

    if pedido.passageiros > veiculo.capacidade:
      continue

    # Caminho: veículo → ponto de recolha
    tempo1, caminho1 = procura(veiculo.posicao, pedido.startPoint)

    # Caminho: ponto de recolha → destino
```

```
    tempo2, caminho2 = procura(pedido.startPoint, pedido.endPoint)

    distanciaTotal = calculaDistancia(caminho1) +
                 calculaDistancia(caminho2)

    if precisaAbastecer(veiculo, distanciaTotal):
        estacao = encontraEstacaoPerto(veiculo, tipo)

        # Caminho: veículo → estação
        tempoEstacao, _ = procura(veiculo.posicao, estacao)

        # Caminho: estação → ponto de recolha
        tempo1Novo, caminho1Novo = procura(estacao, pedido.startPoint)

        tempo1 = tempoEstacao + tempoRefuel + tempo1Novo

    tempoTotal = tempo1 + tempo2

    if tempoTotal < melhorTempo:
        melhorTempo = tempoTotal
        melhorVeiculo = veiculo
```

7. Algoritmos Implementados

De seguida, falaremos dos algoritmos de procura não informada e informada utilizados, bem como as suas respetivas heurísticas no caso dos últimos.

Os algoritmos que consideram os custos das arestas utilizam um custo ponderado baseado em três fatores principais: tempo total, custo total e impacto ambiental. Estes critérios foram escolhidos por serem os mais relevantes para a empresa.

- O tempo total recebe um peso de 0.35, sendo calculado com base na distância entre os nodos, já ajustada por fatores de trânsito e clima.
- O custo total, que é a maior prioridade da empresa devido ao objetivo de maximizar lucros, tem um peso de 0.50.
- O impacto ambiental, refletido pelas emissões de gases poluentes, tem um peso de 0.15 e também é incorporado no cálculo do custo do nodo.

Este cálculo é efetuado em tempo de execução pelos algoritmos no ficheiro `cost_function.py`.

7.1. Procura não informada

Os algoritmos de procura não informada implementados utilizam como critério de ponderação apenas os custos entre os nodos, não tendo em consideração as condições dinâmicas da simulação, tais como o trânsito ou o clima. Considerou-se pertinente a implementação de três algoritmos deste tipo: **Breadth-First Search**, **Depth-First Search** e **Uniform-Cost Search**, por permitirem comparar abordagens simples e clássicas de exploração do espaço de estados.

7.2. Procura informada

No âmbito da procura informada, foram implementados os algoritmos **Greedy** e **A-Star**, que recorrem a heurísticas para orientar a exploração do espaço de estados de forma mais eficiente. Estes algoritmos revelam-se mais robustos, uma vez que incorporam informação adicional sobre o problema, permitindo uma melhor adaptação às variações dinâmicas da simulação.

Enquanto o **Greedy** baseia as suas decisões exclusivamente no valor da heurística, privilegiando os estados que aparentam estar mais próximos do objetivo, o **A-Star** combina o custo real do percurso já efetuado com a estimativa heurística do custo restante, garantindo, sob determinadas condições, a obtenção de soluções ótimas.

7.2.1. Heurísticas

Nos algoritmos de procura informada, as heurísticas desempenham um papel fundamental, pois fornecem estimativas que permitem direcionar a procura de forma mais eficiente. Neste projeto, foram desenvolvidas várias heurísticas com o objetivo de lidar com diferentes aspetos do problema, procurando equilibrar fatores como a distância, o tempo, as restrições impostas e os eventos dinâmicos da simulação.

A heurística baseada na **distância euclidiana** é uma abordagem simples, mas frequentemente eficaz. No entanto, ignora outros fatores relevantes, como as condições climatéricas ou o trânsito existente.

A heurística baseada em **custos operacionais** tem como objetivo penalizar percursos que impliquem um elevado consumo de combustível, mesmo que sejam mais rápidos. Esta heurística tende a priorizar a utilização de veículos elétricos, dado que o seu custo operacional é consideravelmente inferior ao dos veículos a combustão.

A heurística baseada no **tempo** é uma das mais sofisticadas desenvolvidas. Esta tem em consideração a velocidade média do veículo, o nível de trânsito associado a cada aresta e as condições climatéricas de cada nodo, que afetam todas as arestas a ele associadas. Com base nestes fatores, é calculada uma estimativa do tempo de viagem, permitindo uma boa adaptação ao dinamismo da simulação, sendo especialmente eficaz quando a prioridade é minimizar o tempo total.

A heurística de **traffic-avoidance** (evitar trânsito) introduz penalizações em caminhos que atravessam zonas com trânsito, uma vez que, tipicamente, áreas congestionadas — sobretudo quando o trânsito é severo — tendem a estender-se por mais do que uma única estrada. Desta forma, o algoritmo é incentivado a procurar rotas alternativas, geralmente mais periféricas, reduzindo a probabilidade de enfrentar congestionamentos prolongados.

A heurística de **impacto ambiental** foca-se na redução do impacto ecológico associado à mobilidade, atribuindo prioridade à utilização de veículos elétricos ou híbridos, em detrimento de veículos exclusivamente a combustão.

Por fim, foi desenvolvida uma heurística **combinada**, na qual são calculados os valores de cada uma das heurísticas anteriormente descritas e considerada a sua média. Esta abordagem permite equilibrar os diferentes critérios, tirando partido das vantagens de cada heurística individual.

8. Resultados Obtidos

Simulou-se um dia completo de operação, decorrendo entre as 8h00 e as 20h 00, com um total de 9 veículos operacionais. Destes veículos, 3 são elétricos, 5 a combustão e 1 híbrido, permitindo avaliar o comportamento do sistema face a diferentes tipos de motorização.

Todos os veículos iniciam a simulação na garagem da empresa. Relativamente à capacidade, 5 veículos possuem capacidade para 7 passageiros, enquanto 2 veículos possuem capacidade para 8 passageiros, possibilitando a análise de pedidos com diferentes dimensões.

Os resultados obtidos pelos algoritmos de procura serão apresentados de seguida com disposição tabelar. Estes resultados permitem analisar a eficiência e o desempenho dos algoritmos nos diferentes cenários, tendo sido todos obtidos para o mesmo conjunto de pedidos, garantindo assim a comparabilidade entre cenários.

Os indicadores analisados foram:

- R – número de pedidos completos;
- D – distância total percorrida;
- T – tempo total de operação;
- C – custo acumulado em gasolina;
- CO_2 – emissões totais.

8.1. Cenários

Procedeu-se à criação de 3 cenários distintos, descritos nesta seção.

Cenário 1: Este cenário simula um dia ideal, sem qualquer tipo de trânsito ou estradas encerradas, e com condições climatéricas favoráveis, tendo como principal objetivo avaliar o comportamento base dos algoritmos em condições ótimas, servindo como referência para comparação com os restantes cenários.

Cenário 2: Tentou-se que este cenário fosse o mais semelhante possível a um dia normal de operação da empresa, num cenário real urbanístico, tendo sido encerradas algumas estradas e introduzido trânsito intenso na zona central entre as 08 : 00 e as 11 : 00 e entre as 17 : 00 e as 19 : 00, juntamente com outros locais com trânsito moderado e diferentes condições climatéricas, com o objetivo de analisar o desempenho dos algoritmos em situações realistas e a sua capacidade de adaptação a congestionamentos previsíveis e restrições parciais da rede viária.

Cenário 3: Este cenário tem como objetivo representar um dia atípico, simulando um cenário de condução caótico, onde existe congestionamento extremo

espalhado por todo o grafo, condições climatéricas muito adversas e inúmeras estradas encerradas, tendo como objetivo principal testar se os algoritmos conseguem fazer decisões de rotas mais eficientes pelas estradas que não estão afetadas por estas adversidades.

8.2. Resultados

Na tabela seguinte, apresentam-se os resultados adquiridos. Para os algoritmos de procura informada, existem as variações com diversas heurísticas, representadas por "Nome Do Algoritmo (Nome Da Heurística)", denominadas como acima apresentadas.

Algoritmo	Cenário 1	Cenário 2	Cenário 3
BFS	R: 27 D: 1170.82 km T: 2098.07 min C: 183.76 € CO_2: 4.466 kg	R: 24 D: 1118.53 km T: 2945.07 min C: 116.59 € CO_2: 4.451 kg	R: 10 D: 701.84 km T: 7794.55 min C: 153.23 € CO_2: 3.116 kg
DFS	R: 13 D: 2591.29 km T: 3651.90 min C: 421.22 € CO_2: 12.190 kg	R: 10 D: 2233.40 km T: 4906.10 min C: 345.55 € CO_2: 11.586 kg	R: 0 D: 1018.42 km T: 8266.66 min C: 202.56 € CO_2: 3.730 kg
Uniform-Cost	R: 25 D: 1040.18 km T: 1870.87 min C: 244.17 € CO_2: 5.063 kg	R: 22 D: 1073.24 km T: 2605.26 min C: 110.28 € CO_2: 4.914 kg	R: 10 D: 690.05 km T: 5884.84 min C: 105.17 € CO_2: 4.036 kg
A-Star (Distância Euclidiana)	R: 25 D: 1040.18 km T: 1870.87 min C: 244.17 € CO_2: 5.063 kg	R: 22 D: 1073.24 km T: 2605.26 min C: 110.28 € CO_2: 4.914 kg	R: 10 D: 690.05 km T: 5884.84 min C: 105.17 € CO_2: 4.036 kg
A-Star (Tempo Estimado)	R: 25 D: 1040.18 km T: 1870.87 min C: 244.17 € CO_2: 5.063 kg	R: 22 D: 1073.24 km T: 2605.26 min C: 110.28 € CO_2: 4.914 kg	R: 11 D: 751.03 km T: 6669.61 min C: 170.80 € CO_2: 3.953 kg
A-Star (Custo Operacional)	R: 25 D: 1040.18 km T: 1870.87 min C: 244.17 € CO_2: 5.063 kg	R: 22 D: 1073.24 km T: 2605.26 min C: 110.28 € CO_2: 4.914 kg	R: 10 D: 690.05 km T: 5884.84 min C: 105.17 € CO_2: 4.036 kg
A-Star (Impacto Ambiental)	R: 25 D: 1040.18 km T: 1870.87 min C: 244.17 € CO_2: 5.063 kg	R: 22 D: 1073.24 km T: 2605.26 min C: 110.28 € CO_2: 4.914 kg	R: 10 D: 690.05 km T: 5884.84 min C: 105.17 € CO_2: 4.036 kg

Tabela 1: Comparação de Algoritmos em Diferentes Cenários.

Algoritmo	Cenário 1	Cenário 2	Cenário 3
Greedy (Tempo Estimado)	R: 28 D: 1171.60 km T: 2095.88 min C: 166.32 € CO_2: 4.596 kg	R: 24 D: 1264.21 km T: 2737.76 min C: 200.60 € CO_2: 5.372 kg	R: 10 D: 828.15 km T: 6530.90 min C: 126.38 € CO_2: 4.117 kg
Greedy (Custo Operacional)	R: 28 D: 1171.60 km T: 2095.88 min C: 166.32 € CO_2: 4.596 kg	R: 25 D: 1205.11 km T: 3251.73 min C: 197.09 € CO_2: 4.576 kg	R: 10 D: 700.63 km T: 7475.45 min C: 104.50 € CO_2: 2.914 kg
Greedy (Impacto Ambiental)	R: 28 D: 1171.60 km T: 2095.88 min C: 166.32 € CO_2: 4.596 kg	R: 25 D: 1205.11 km T: 3251.73 min C: 197.09 € CO_2: 4.576 kg	R: 10 D: 700.63 km T: 7475.45 min C: 104.50 € CO_2: 2.914 kg
Greedy (Evitar Trânsito)	R: 28 D: 1171.60 km T: 2095.88 min C: 166.32 € CO_2: 4.596 kg	R: 24 D: 1242.42 km T: 2854.91 min C: 185.00 € CO_2: 4.356 kg	R: 10 D: 870.45 km T: 6952.59 min C: 169.62 € CO_2: 4.149 kg
Greedy (Combinada)	R: 28 D: 1171.60 km T: 2095.88 min C: 166.32 € CO_2: 4.596 kg	R: 24 D: 1205.70 km T: 2742.85 min C: 175.48 € CO_2: 4.840 kg	R: 11 D: 795.81 km T: 6839.61 min C: 185.47 € CO_2: 3.950 kg
A-Star (Evitar Trânsito)	R: 25 D: 1040.18 km T: 1870.87 min C: 244.17 € CO_2: 5.063 kg	R: 22 D: 1073.24 km T: 2605.26 min C: 110.28 € CO_2: 4.914 kg	R: 10 D: 690.05 km T: 5884.84 min C: 105.17 € CO_2: 4.036 kg
A-Star (Combinada)	R: 25 D: 1040.18 km T: 1870.87 min C: 244.17 € CO_2: 5.063 kg	R: 22 D: 1073.24 km T: 2605.26 min C: 110.28 € CO_2: 4.914 kg	R: 10 D: 690.05 km T: 5884.84 min C: 105.17 € CO_2: 4.036 kg
Greedy (Distância Euclidiana)	R: 28 D: 1171.60 km T: 2095.88 min C: 166.32 € CO_2: 4.596 kg	R: 25 D: 1205.11 km T: 3251.73 min C: 197.09 € CO_2: 4.576 kg	R: 10 D: 700.63 km T: 7475.45 min C: 104.50 € CO_2: 2.914 kg

Tabela 2: Comparação de Algoritmos em Diferentes Cenários.

8.3. Análise de resultados

A análise dos resultados revela padrões interessantes no comportamento dos diferentes algoritmos de procura aplicados à gestão da frota de táxis. No Cenário 1, os algoritmos Uniform-Cost e as várias versões do A-Star demonstram o melhor desempenho em termos de tempo total, conseguindo completar 25 pedi-

dos em apenas 1870.87 minutos, embora com um custo ligeiramente superior. Por outro lado, os algoritmos Greedy conseguem completar mais pedidos (28) e apresentam os menores custos operacionais e emissões de CO2, tornando-os particularmente atrativos do ponto de vista económico e ambiental. O DFS revela-se claramente inadequado para este tipo de aplicação, percorrendo distâncias excessivas e gerando custos e emissões muito superiores aos restantes algoritmos.

No Cenário 2, mantém-se a tendência de superioridade do Uniform-Cost e A--Star no que toca à eficiência de custos, conseguindo o melhor valor de 110.28€ para 22 pedidos completos. O BFS apresenta um desempenho equilibrado, com custos competitivos e um número razoável de pedidos atendidos.

O Cenário 3 revela-se o mais desafiante para todos os algoritmos, com a maioria a conseguir completar apenas 10 pedidos. Aqui, o DFS falha completamente, não conseguindo completar nenhum pedido, o que evidencia as suas limitações críticas. Os algoritmos Greedy destacam-se neste cenário pela eficiência ambiental, conseguindo as menores emissões de CO2.

De forma geral, os resultados sugerem que o A-Star e o Uniform-Cost oferecem o melhor compromisso entre tempo de operação, custo e número de pedidos completados, sendo as escolhas mais adequadas para uma operação de táxis que procure eficiência global. Os algoritmos Greedy podem ser preferíveis quando o objetivo principal é minimizar custos e impacto ambiental, enquanto o DFS deve ser evitado neste contexto devido ao seu desempenho consistentemente inferior.

8.4. Eficiência dos diferentes algoritmos

Considerou-se pertinente incluir métricas relativas à eficiência de cada um dos algoritmos, embora não seja efetuada uma análise ou discussão das mesmas. Estas métricas são apresentadas com um intuito meramente informativo e complementar, ficando a sua interpretação ao critério do leitor.

O Tempo Total mede o tempo somado que, durante a simulação, o algoritmo esteve em execução nos testes realizados. Já o Tempo Médio informa a média de todas as execuções individuais do algoritmo.

Algoritmo	Tempo Total (ms)	Tempo Médio (ms)
BFS	47509.06	124.3693
DFS	35.28	0.1575
Uniform-Cost	93.51	0.2834
A-Star (Dist. Euclidiana)	192.48	0.5833
A-Star (Tempo Estimado)	195.55	0.5926
A-Star (Custo Operacional)	200.22	0.6067
A-Star (Impacto Ambiental)	194.70	0.5900
A-Star (Evitar Trânsito)	192.98	0.5848
A-Star (Combinada)	216.01	0.6546
Greedy (Dist. Euclidiana)	33.38	0.0902
Greedy (Tempo Estimado)	34.93	0.0944
Greedy (Custo Operacional)	38.96	0.1053
Greedy (Impacto Ambiental)	38.12	0.1030
Greedy (Evitar Trânsito)	36.45	0.0985
Greedy (Combinada)	50.58	0.1367

Tabela 3: Desempenho dos Algoritmos - Cenário 1.

Algoritmo	Tempo Total (ms)	Tempo Médio (ms)
BFS	36117.49	109.4469
DFS	28.80	0.1800
Uniform-Cost	88.93	0.2779
A-Star (Dist. Euclidiana)	190.30	0.5947
A-Star (Tempo Estimado)	244.35	0.7636
A-Star (Custo Operacional)	201.87	0.6309
A-Star (Impacto Ambiental)	197.79	0.6181
A-Star (Evitar Trânsito)	240.81	0.7525
A-Star (Combinada)	284.36	0.8886
Greedy (Dist. Euclidiana)	57.22	0.1822
Greedy (Tempo Estimado)	12168.53	39.0017
Greedy (Custo Operacional)	66.07	0.2104
Greedy (Impacto Ambiental)	65.82	0.2096
Greedy (Evitar Trânsito)	1506.60	4.7377
Greedy (Combinada)	1585.57	5.0496

Tabela 4: Desempenho dos Algoritmos - Cenário 2.

Algoritmo	Tempo Total (ms)	Tempo Médio (ms)
BFS	22060.07	134.5126
DFS	24.83	0.1881
Uniform-Cost	42.03	0.2563
A-Star (Dist. Euclidiana)	87.41	0.5330
A-Star (Tempo Estimado)	126.35	0.7019
A-Star (Custo Operacional)	93.10	0.5677
A-Star (Impacto Ambiental)	90.28	0.5505
A-Star (Evitar Trânsito)	94.90	0.5786
A-Star (Combinada)	124.26	0.7577
Greedy (Dist. Euclidiana)	17.25	0.0958
Greedy (Tempo Estimado)	186.07	1.1075
Greedy (Custo Operacional)	19.13	0.1063
Greedy (Impacto Ambiental)	19.08	0.1060
Greedy (Evitar Trânsito)	94.30	0.5750
Greedy (Combinada)	174.76	0.9602

Tabela 5: Desempenho dos Algoritmos - Cenário 3.

9. Conclusão

Do desenvolvimento deste trabalho, retiramos algumas conclusões.

Num sentido mais técnico, adquirimos um entendimento mais profundo e prático dos algoritmos de procura estudados. Conseguimos analisá-los em ação, entender os motivos de ineficiências de algoritmos de procura não informada e entender a importância de uma boa heurística para o o bom desempenho dos algoritmos de procura informada.

Modelamos o problema com base em conceitos importantes, integrando teoria de grafos com algoritmos de procura e aprendendo, na prática, bibliotecas específicas e importantes da linguagem de programação `Python`, como `Matplotlib`que permite criar visualizações interativas e animadas, e `OSM`, que possibilita utilizar dados de mapeamento real de cidades.

Sentimos que o trabalho desenvolvido teve, como pontos positivos, uma modularização com clara separação de responsabilidades, tendo sido trivial alterar os algoritmos de procura sem afetar outros componentes e testar as heurísticas isoladamente de forma simples. O facto de termos utilizado uma *API* externa para incorporar um mapa baseado em dados reais e termos criado uma visualização gráfica também nos orgulha, visto significar que traz um valor de análise de problemas a uma escala real e uma representação visual conseguir ser mais interativa em termos analíticos e fácil de depurar. Utilizamos ainda uma comparação multi--heurística e uma dinâmica de eventos que afeta as rotas em tempo real, o que tem importância.

Também admitimos que houve simplificações de contexto real a um nível exagerado. Assumir possibilidades de reabastecimento de forma estática é problemático, visto este problema ser mais socio-técnico do que matemático. As limitações metodológicas de determinar alterações de clima a certas horas, retirando o fator de aleatoriedade da vida real. A agregação de tempo foi uma decisão também tomada por simplificação, processando a simulação por *ticks* e não sendo *event-driven*. De qualquer maneira, achamos que são simplificações pouco problemáticas não afetando significativamente a comparação de algoritmos e considerando escala e imprevisibilidade suficientes relativamente ao objetivo principal do projeto.

Para trabalho futuro, seria interessante fazer um *benchmark* mais rigoroso, utilizando dados de cidades exageradamente maiores e com um volume de taxis muito superior, testando a escala do nosso programa e fiabilidade da solução por ele apresentada. Alterações à forma do programa, permitindo uma aprendizagem automática com calibração dinâmica de pesos dos custos e uma simulação estocástica, assumindo clima e trânsito probabilístico invés de determinístico são também boas evoluções do programa que permitiriam ter um interesse prático maior, não sendo, no entanto, parâmetro de avaliação nesta unidade curricular.

No final, sentimos que o trabalho foi realizado com ânimo e competência técnica suficiente ao esperado pela equipa docente, tendo atingido os objetivos propostos no enunciado e conseguindo compreender conceitos fundamentais adjacentes à unidade curricular e explorar novas ferramentas úteis em contextos diversos.