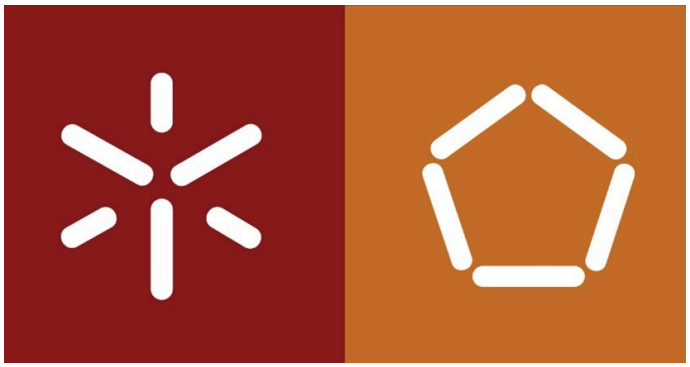

Universidade do Minho
Mestrado em Inteligência Artificial

Relatório do Projeto
Computação Inspirada na Natureza
2025/2026

Grupo 1
Francisco Costa PG60387
João Mendes PG60388
Simão Silva PG60393
Vasco Macedo PG60394

Índice

Índice de Figuras

1. Introdução

No âmbito da cadeira de Computação Inspirada na Natureza, do 1º ano do Mestrado em Inteligência Artificial, foi-nos proposta a elaboração de um projeto com o propósito de ajudar utilizadores a escolher a melhor forma de se deslocarem no Grande Porto.

Atualmente, os cidadãos ponderam múltiplos fatores nas suas deslocações, como a pegada ecológica, o custo financeiro e o benefício físico da caminhada. Este cenário transforma o planeamento de rotas num problema de otimização multiobjectivo, onde frequentemente não existe uma solução única ideal, mas sim um conjunto de compromissos (trade-offs) entre objetivos conflituosos.

O presente projeto visa desenvolver um planeador de rotas multimodal que integra a rede do Metro do Porto, a rede de autocarros da STCP (Sociedade de Transportes Coletivos do Porto) e percursos pedonais. Ao contrário das abordagens clássicas de caminho mais curto, este sistema utiliza algoritmos de computação evolutiva para explorar o espaço de pesquisa e devolver um conjunto de soluções não dominadas (Fronteira de Pareto). O sistema permite ao utilizador personalizar as suas preferências — valorizando, por exemplo, a sustentabilidade ambiental (minimização de CO_2) ou a prática de exercício físico — e respeita restrições operacionais, como o limite máximo de transbordos e o tempo máximo de caminhada permitido.

Para concretizar este objetivo, foram utilizados dados reais no formato GTFS (General Transit Feed Specification), permitindo uma modelação fidedigna das redes de transporte, horários e frequências. A solução final inclui uma interface web interativa que facilita a visualização e comparação das diferentes rotas sugeridas pelo algoritmo.

2. Dados e Pré-Processamento

O desenvolvimento de um planeador multimodal que simule com precisão as redes de transporte público do Grande Porto exige a utilização de dados normalizados e detalhados. A base informacional deste projeto reside nos conjuntos de dados GTFS, fornecidos pelo Metro do Porto e pela STCP, complementados por informações específicas sobre emissões de dióxido de carbono e regras de caminhada. O pré-processamento é a etapa crucial que transforma esta coleção de ficheiros tabulares estáticos num modelo temporal e espacial dinâmico, apto para a construção do Grafo Multimodal e para a avaliação das funções objetivo.

2.1. Dados GTFS

Definição dos ficheiros utilizados, do tipo txt, provenientes da STCP e sua utilização no pré-processamento:

- agency – Agências de transporte público com serviços representados neste conjunto de dados.
- calendar_dates/calendar – Ficheiros do GTFS que descrevem a validade temporal dos serviços (dias da semana e exceções). Nesta versão do projeto, estes ficheiros são carregados para permitir extensões futuras, mas não é aplicada filtragem por dia, assume-se que os horários disponíveis representam um cenário operacional típico.
- routes – Percursos de transporte público. Uma rota é um conjunto de viagens que são apresentadas aos passageiros como um único serviço. Usado para associar viagens aos seus modos de transporte (Metro ou STCP) e determinar o fator de emissão de CO2.

- shapes – Descrevem a geometria do traçado das linhas. Embora estejam disponíveis no GTFS, não foram utilizados para medir distâncias nesta implementação. As distâncias entre paragens (necessárias para emissões e caminhada) são estimadas a partir das coordenadas das paragens usando a fórmula de Haversine.
- stop_times – Horários a que um veículo chega e parte dos pontos de paragem em cada viagem. Fundamental para calcular o tempo de espera e o tempo de viagem.
- stops – Paragens onde os veículos embarcam ou desembarcam passageiros. Também define estações e entradas de estações. Define os nós (pontos de interesse) do grafo multimodal.
- transfers – Regras para efetuar ligações em pontos de transferência entre rotas. Define o custo temporal das arestas de transferência no grafo.
- trips – Deslocamentos para cada percurso. Um deslocamento é uma sequência de duas ou mais paragens que ocorrem durante um período de tempo específico. Liga as linhas (routes.txt) aos horários (stop_times.txt).

A lista dos ficheiros utlizados, provenientes do metro, são do mesmo tipo do que os provenientes da STCP, com a adição de mais dois:

- fare_attributes – Informação sobre tarifas para as linhas de transporte público.
- fare_rules – Regras para a aplicação de tarifas em itinerários.

2.2. Emissões de CO_2

O objetivo é minimizar as emissões, que dependem diretamente da distância percorrida pelo veículo e do fator de emissão associado a cada modo de transporte.

A caminhada é considerada neutra em carbono neste modelo, com uma emissão de 0 gramas por quilómetro. É essencial para ligar os nós de paragens a nós artificiais e garantir que os tempos de acesso ao transporte público são realistas.

Os fatores de emissão utilizados são baseados em estudos de sustentabilidade do transporte e a velocidade pedonal é a velocidade média a que uma pessoa se desloca enquanto caminha. Estes valores estão definidos no ficheiro src/constants.py, o que permite uma gestão centralizada e fácil ajuste dos parâmetros de emissão para toda a aplicação.

- Metro: 40 gramas CO_2 / passageiro·km
- STCP (autocarro): 109.9 gramas CO_2 / passageiro·km
- Velocidade pedonal constante: 1.4 m/s ≈ 5 km/h

As emissões de CO2 para um percurso completo são calculadas somando as emissões de cada segmento de transporte motorizado. A distância é estimada por haversine (fórmula usada para determinar a distância entre dois pontos que pertençam a uma esfera dadas as suas longitudes e latitudes) entre coordenadas. A forma de cálculo é dada por:

$$E_{\text{total}} = \sum_{i \in \text{ segmentos}} D_i * F_{\text{emissão, modo(i)}}$$

- E_{total} é a emissão total em gramas de CO2.
- D_i é a distância do segmento i em quilómetros.
- $F_{\text{emissão, modo(i)}}$ é o fator de emissão do modo de transporte utilizado no segmento i (gCO2/p.km).

2.3. Travessias das Pontes

A modelação precisa da componente pedonal entre o Porto e Vila Nova de Gaia. Isto exigiu uma atenção especial às regras de acessibilidade das pontes sobre o Rio Douro.

O sistema aplica uma política de restrição no módulo de construção do grafo (graph_builder.py) para as travessias entre as duas margens do rio. Estas regras estão centralizadas em ficheiros de configuração (bridges_pedestrian_rules.txt e bridges_geometry.json) e definem dois tipos de estruturas. As pontes em que é permitido andar a pé, sendo estas a ponte Luís I e a ponte Infante Dom Henrique, e as pontes onde é proibido transitar a pé, nomeadamente a ponte da Arrábida, do Freixo, de São João e a Maria Pia.

3. Construção do Grafo Multimodal

A Construção do Grafo Multimodal é a etapa central do projeto, onde os dados tabulares e temporais (GTFS) são transformados numa estrutura de grafo dinâmico. O grafo, modelado através da biblioteca NetworkX, representa o espaço de pesquisa para o algoritmo evolutivo. O processo, gerido no módulo graph_builder.py, resulta num grafo G = (V, E), onde V é o conjunto de nós (vértices) e E é o conjunto de arestas (ligações).

3.1. Tipos de nós

Os nós do grafo (V) representam tanto paragens físicas como pontos auxiliares que permitem ligar qualquer origem/destino à rede de transporte e suportar percursos multimodais.:

- Paragens STCP: nós associados às paragens de autocarro (STCP).
- Estações do Metro: nós associados às estações/paragens do Metro do Porto.
- Nós artificiais (origem e destino): nós criados dinamicamente para representar a origem e o destino do utilizador. Estes nós ligam-se à rede principal através de arestas pedonais para paragens próximas, garantindo que é sempre possível iniciar e terminar a viagem a pé, a partir de qualquer local.

3.2. Tipos de arestas

As arestas do grafo (E) representam ligações possíveis entre nós e incluem custos associados (tempo, emissões), que são posteriormente agregados para avaliar um percurso completo:

- Arestas de transporte (Metro/STCP): Ligam duas paragens consecutivas de uma linha/viagem, com base na sequência de paragens definida pelos dados GTFS (por exemplo, stop_times.txt). Estas arestas contribuem para:
  - Tempo no veículo (tempo estimado do segmento de viagem);
  - Emissões de CO_2, calculadas a partir da distância do segmento e do fator de emissão do modo (Metro/STCP).
- Arestas pedonais (caminhada): Ligam nós que estejam dentro de um raio máximo de proximidade pedonal (definido como parâmetro do sistema, por exemplo algumas centenas de metros) e também suportam o acesso inicial/final a partir de coordenadas arbitrárias (origem/destino). O custo é o tempo de caminhada (velocidade pedonal constante) e as emissões são nulas. A criação destas arestas respeita as regras de travessia das pontes sobre o Douro.

- Transferências / transbordos (mudança de linha ou modo): As mudanças entre serviços são modeladas através de:
  - ligações curtas (muitas vezes pedonais) entre paragens próximas, permitindo transbordos (incluindo Metro ↔ STCP);
  - e, quando disponível, informação GTFS de transferências (por exemplo, transfers.txt) para representar ligações recomendadas entre paragens/estações.

O tempo de espera não é representado como "aresta de espera" num grafo temporal; em vez disso, é estimado durante a avaliação do percurso, com base em headways/frequências, sendo tipicamente contabilizado quando o utilizador entra num serviço/linha (ou quando há mudança relevante no percurso).

4. Formulação do Problema

O problema de planeamento de rotas multimodal é formulado como um Problema de Otimização Multi-Objetivo (MOP). O objetivo não é encontrar uma única solução ótima global, mas sim obter um conjunto de soluções de compromisso (não-dominadas), conhecido como Fronteira de Pareto, para que o utilizador possa escolher a rota que melhor se adequa às suas preferências.

Cada solução corresponde a um percurso P (sequência de nós/arestas) e é avaliada por um vetor de funções objetivo:

$$\min F(P) = \left[f_{\text{tempo}}(P), f_{\text{CO}_2}(P), f_{\text{caminhada}}(P) \right]$$

onde:

- $f_{\text{tempo}}(P)$ é o tempo total do percurso (caminhada + tempo em transporte + espera estimada, quando aplicável);
- $f_{\text{CO}_2}(P)$ é a emissão total de CO_2 associada aos segmentos em transporte (Metro/STCP);
- $f_{\text{caminhada}}(P)$ representa o objetivo ligado ao exercício físico. Como o algoritmo trabalha em regime de minimização, este objetivo pode ser tratado, por exemplo, como a minimização do negativo do tempo/distância a pé (equivalente a maximizar caminhada), ou através de uma formulação equivalente definida na implementação.

Além dos objetivos, são consideradas restrições, como:

- limite máximo de tempo a pé;
- limite máximo de transbordos

Soluções que violem restrições são penalizadas durante a avaliação, tornando-as menos competitivas no processo evolutivo.

4.1. Funções Objetivo

O sistema considera três objetivos principais, que frequentemente entram em conflito entre si (a rota mais rápida pode não ser a mais ecológica):

1. Minimizar tempo total de viagem.
2. Minimizar emissões de CO_2 (dióxido de carbono).
3. Maximizar ou minimizar caminhada (exercício).

4.2. Restrições

Para garantir que as soluções geradas são viáveis e aceitáveis para o utilizador, o espaço de pesquisa é limitado por restrições rígidas ("hard constraints"). Se uma rota violar qualquer uma destas restrições, é aplicada uma penalização severa (PENALTY) à sua aptidão (fitness), excluindo-a efetivamente da seleção.

- Limite máximo de caminhada (Wmax): impede rotas com tempo total a pé excessivo.
- Máximo de transbordos (Tmax): limita o número de trocas de veículo/linha.
- Regras de travessia das pontes: impede ligações pedonais em pontes onde a travessia não é permitida.

5. Algoritmo de Otimização

Para resolver o problema de otimização multi-objetivo formulado, foi implementado um Algoritmo Evolutivo (EA), especificamente o NSGA-II (Non-dominated Sorting Genetic Algorithm II), utilizando a biblioteca Python deap. Ao contrário dos métodos exatos, o NSGA-II trabalha com uma população de soluções, evoluindo-as ao longo de gerações para aproximar a Fronteira de Pareto ótima.

5.1. Representação da solução

Dado que o problema consiste em encontrar rotas num grafo direcionado G = (V,E), adotou-se uma representação baseada em caminhos (Path Representation) com tamanho variável. Neste modelo, cada indivíduo (solução candidata) não é representado por um vetor de características fixas, mas sim por uma sequência ordenada de identificadores de nós que descreve o trajeto completo desde a origem até ao destino. Formalmente, um indivíduo (rota) P é definido como um vetor de nós:

$$P = [v_1, v_2, ..., v_k]$$

- v_1 corresponde ao nó de origem;
- v_k corresponde ao nó de destino;
- k é o comprimento do caminho, número de nós, que é variável entre indivíduos, uma vez que diferentes rotas podem ter números distintos de paragens ou transferências;
- Cada v_i representa um ID único no grafo multimodal.

5.2. População inicial

A geração de rotas puramente aleatórias num grafo de grande dimensão tende a produzir caminhos inválidos, redundantes ou excessivamente longos. Para melhorar a qualidade inicial e acelerar a convergência (evitando também convergência prematura), foi adotada uma estratégia de Inicialização Inteligente (Seeding), inspirada nas sugestões do enunciado.

Em vez de iniciar a população totalmente ao acaso, uma parte significativa das soluções iniciais é gerada por uma heurística baseada em caminhos mais curtos, usando Dijkstra (ou abordagem equivalente) com uma função de custo combinada entre objetivos contraditórios (tempo e emissões), no espírito de MOEA/D:

1. Seeding por combinação ponderada (grelha λ): São geradas várias soluções iniciais resolvendo caminhos mínimos com uma função do tipo:

$$\text{custo}_\lambda = \lambda * \hat{f}_{\text{tempo}} + (1 - \lambda) * \hat{f}_{\text{CO}_2}$$

Onde $\hat{f}$ indica valores normalizados (para tornar as escalas comparáveis) e λ varia num conjunto discreto (ex.: de 0 a 1).

Isto produz, logo no arranque, soluções distribuídas ao longo de diferentes compromissos entre rapidez e sustentabilidade.

1. Diversificação da população inicial: O restante da população é preenchido com soluções adicionais geradas por variações heurísticas (por exemplo, introduzindo ruído no custo, pequenas perturbações em segmentos do caminho) e/ou por geração exploratória (ex.: random walk controlado), garantindo diversidade genética.

Esta abordagem permite começar a evolução com soluções já "boas" (não apenas aleatórias), e direciona o esforço do algoritmo para descobrir trade-offs intermédios e melhorar a cobertura da Fronteira de Pareto, incluindo soluções com perfis distintos também ao nível da caminhada (exercício) e dos transbordos, respeitando as restrições definidas.

5.3. Algoritmo Evolutivo

O ciclo evolutivo segue a estrutura padrão do NSGA-II, selecionando os sobreviventes com base em dois critérios hierárquicos:

1. Non-Dominated Sorting (Ordenação Não-Dominada): A população é dividida em "fronteiras" (F1, F2, ...). A fronteira F1 contém as soluções que não são dominadas por nenhuma outra. F2 contém as soluções dominadas apenas por membros de F1, e assim sucessivamente. O objetivo é maximizar a pertença às primeiras fronteiras.
2. Crowding Distance (Distância de Aglomeração): Dentro da mesma fronteira, o algoritmo prefere soluções que estejam mais "isoladas" no espaço dos objetivos. Isto força o algoritmo a manter a diversidade, evitando que todas as soluções convirjam para um único ponto (ex: apenas soluções rápidas) e ignorando outras regiões do Pareto (ex: soluções ecológicas).

5.4. Operadores genéticos

A aplicação de operadores genéticos em caminhos de grafos exige cuidados especiais para manter a validade das rotas.

5.4.1. Crossover (Cruzamentos)

O operador de cruzamento combina informações de dois progenitores para criar descendentes. No contexto de rotas, utilizou-se uma abordagem baseada em interseções geográficas:

- Identifica-se um nó comum (interseção) que exista em ambos os caminhos dos progenitores.
- Trocam-se os segmentos de rota a partir desse ponto.
- Se não houver nós comuns (além da origem/destino), o cruzamento não é possível e os pais mantêm-se.

5.4.2. Mutação

A mutação introduz variabilidade e previne a estagnação em mínimos locais.

- Seleciona-se aleatoriamente um subsegmento da rota original (entre dois nós intermédios u e v).

- Este segmento é removido e substituído por um novo caminho entre u e v.
- Este novo subcaminho pode ser gerado aleatoriamente ou através de uma heurística local, permitindo explorar variantes de trajeto numa zona específica da cidade

6. Arquitetura da Aplicação

O sistema foi desenvolvido em *Python* (v3.10+) segundo uma arquitetura modular que separa a ingestão de dados, a construção do grafo, a otimização evolutiva e a interface de apresentação. Esta organização facilita a manutenção, a testabilidade de componentes individuais e a evolução futura do projeto.

6.1. Tecnologias e Software Utilizado

A implementação recorre a um conjunto reduzido de bibliotecas, selecionadas por suportarem diretamente as necessidades do projeto: processamento de dados GTFS, modelação em grafo, otimização evolutiva e interface web interativa. As dependências encontram-se explicitadas no ficheiro `requirements.txt` do repositório.

6.1.1. Bibliotecas principais

- *pandas*: carregamento e manipulação dos ficheiros GTFS em estruturas tabulares (DataFrames), suportando operações de pré-processamento.
- *NetworkX*: construção e manipulação do grafo multimodal, bem como suporte a heurísticas baseadas em caminhos mínimos (por exemplo, Dijkstra) usadas na inicialização inteligente.
- *DEAP*: implementação do algoritmo evolutivo NSGA-II, incluindo gestão de populações, seleção e operadores genéticos.
- *Streamlit*: implementação da interface web interativa para configuração de origem/destino, preferências (sliders) e visualização/comparação de rotas.

6.1.2. Ambiente de desenvolvimento e controlo de versões

- *Git* e *GitHub*: controlo de versões e alojamento/partilha do repositório do projeto.
- *Visual Studio Code*: IDE utilizada no desenvolvimento, depuração e organização do código.

6.2. Organização modular

A estrutura do projeto encontra-se dividida em módulos com responsabilidades bem definidas:

- Importação GTFS (`loader.py`): carrega os ficheiros GTFS (Metro e STCP) para estruturas em memória (DataFrames do *pandas*). São carregados também `calendar`/`calendar_dates` por completude e para futura extensão, mas nesta versão não é feita filtragem por dia de operação; assume-se o cenário base representado no feed.
- Grafo Multimodal (`graph_builder.py`): converte os dados carregados num grafo direcionado utilizando *NetworkX*. Este módulo cria nós/arestas e aplica regras do domínio, nomeadamente as restrições de travessia pedonal das pontes sobre o Douro.
- Motor Evolutivo (`evolution.py`): implementa o ciclo de vida do NSGA-II com recurso à *DEAP*, gerindo a população, o loop geracional e a aplicação dos operadores genéticos.
- Avaliação (`fitness.py`): calcula as métricas de cada rota (Tempo, CO_2, Caminhada) e verifica as restrições rígidas (Wmax, Tmax), aplicando penalizações quando necessário.
- Heurísticas (`baselines.py`): disponibiliza rotinas de caminho mínimo (por exemplo, Dijkstra) usadas na inicialização inteligente (seeding).

- Camada de Apresentação (`app/streamlit_app.py`): aplicação web em *Streamlit* responsável pela interação com o utilizador e pela apresentação dos resultados.

6.3. Fluxo de Execução

1. *Input* do Utilizador: o utilizador define origem, destino e preferências na interface web.
2. Validação e geo-codificação: o sistema converte os nomes das paragens (ou opções selecionadas) em nós válidos do grafo.
3. Inicialização: o grafo é carregado (ou recuperado de cache) e a população inicial é gerada com base em heurísticas.
4. Ciclo evolutivo: o NSGA-II é executado durante N gerações, aproximando a Fronteira de Pareto.
5. Pós-processamento: as soluções não dominadas são descodificadas em instruções legíveis (segmentos da rota).
6. Visualização: as rotas são apresentadas numa tabela comparativa e numa descrição passo-a-passo.

6.4. Interface Web

A interface foi desenhada para permitir a exploração dos **trade-offs** multiobjetivo de forma intuitiva. As principais funcionalidades incluem:

- Ajuste, em tempo real, da importância relativa de cada objetivo (Tempo vs. CO_2 vs. Exercício) através de sliders, reordenando/selecionando as rotas sugeridas a partir do conjunto não dominado.
- Apresentação de indicadores por rota (por exemplo, emissões em gramas e distância a pé em metros), suportando uma decisão informada.
- Descrição textual detalhada de cada segmento da viagem, incluindo pontos de transbordo e estimativas de espera quando aplicável.

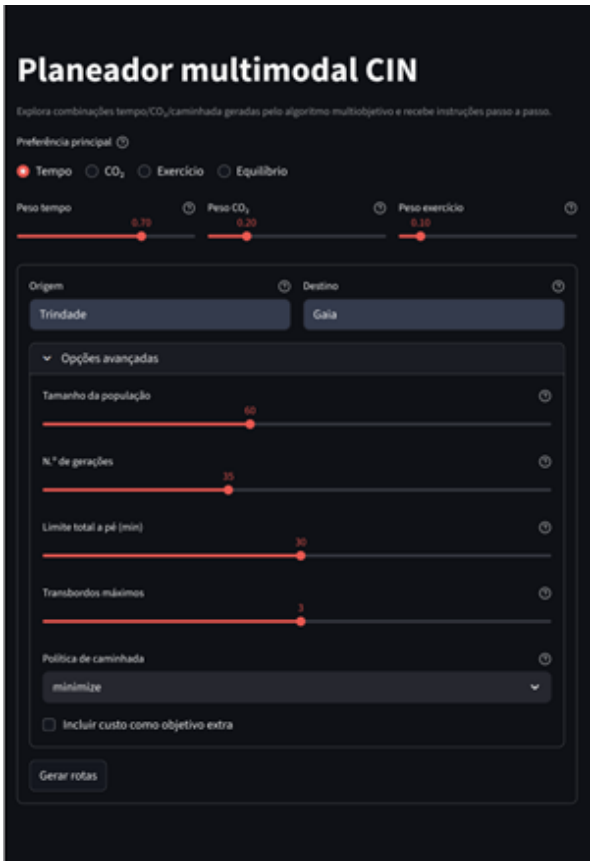

Figura 1: Definição de rota e suas variáveis.

Figura 2: Resultados apresentados.

7. Resultados Obtidos

7.1. Casos de teste (cenários)

Para cumprir o requisito de avaliação com diferentes níveis de dificuldade, foi implementado um gerador de cenários em `src/scenarios.py`. Cada cenário é definido por um par (origem, destino) obtido através de um **random walk** no grafo multimodal: escolhe-se um nó inicial elegível (paragem/estação dos modos Metro ou STCP) e percorrem-se arestas aleatórias, evitando revisitar nós, até atingir um comprimento alvo.

Os cenários são agrupados automaticamente em três classes de dificuldade com base no número de arestas do **walk** gerado:

- **short**: 1–3 arestas;
- **mid**: 6–10 arestas;
- **long**: ≥ 12 arestas.

Para cada cenário, é também calculado `length_edges_shortest`, correspondente ao comprimento (em número de arestas) do caminho mais curto por tempo entre origem e destino (usado apenas como caracterização adicional do problema). A lista completa de cenários e respetivos metadados é exportada para `outputs/experiments/scenarios.json`.

No conjunto de experiências incluído no repositório foram gerados 9 cenários (3 por classe: **short**, **mid** e **long**), permitindo comparar o comportamento dos métodos em trajetos curtos e em trajetos com maior combinatória multimodal.

7.2. Protocolo experimental (baseline vs. NSGA-II)

A avaliação compara duas abordagens, executadas para o mesmo conjunto de cenários:

1. Baseline determinístico (Dijkstra-λ): para cada cenário, executa-se Dijkstra múltiplas vezes com uma soma ponderada dos objetivos tempo e emissões. O custo de cada aresta é definido por:

$$c_\lambda(u,v) = \lambda \cdot \frac{t(u,v)}{T_{\text{norm}}} + (1-\lambda) \cdot \frac{e(u,v)}{E_{\text{norm}}}$$

onde $t(u,v)$ é o tempo da aresta, $e(u,v)$ é a emissão estimada da aresta, e são usados fatores de normalização heurísticos ($T_{\text{norm}} = 3600$ s e $E_{\text{norm}} = 100$ g) para tornar as escalas comparáveis. O parâmetro λ é varrido numa grelha discreta (por omissão, $\lambda \in (0.0, 0.05, ..., 1.0)$), produzindo múltiplos caminhos candidatos. No final aplica-se filtragem Pareto 2D (tempo total, emissões), obtendo a fronteira baseline por cenário.

1. NSGA-II: para cada cenário, executa-se o NSGA-II implementado em `src/evolution.py`, avaliando indivíduos (rotas) por múltiplos objetivos. Nas experiências exportadas em `outputs/experiments/`, o NSGA-II é inicializado com **seeding** informado pelo domínio: parte da população inicial é composta por caminhos gerados pelo mesmo Dijkstra-λ (para vários valores de λ), sendo o remanescente preenchido com soluções geradas de forma exploratória (por exemplo, caminhos obtidos por **random walk** e/ou variantes ruidosas do caminho mais curto). Esta estratégia acelera a obtenção de soluções viáveis e promove cobertura inicial da Fronteira de Pareto.

As experiências são executadas e exportadas pelo **runner** `src/experiments.py`, que guarda, por cenário, a fronteira Pareto do baseline (`baseline_pareto.json`), a fronteira Pareto do NSGA-II (`pareto_solutions.json`) e os artefactos necessários ao cálculo do hipervolume.

7.3. Métrica de comparação: Hipervolume (HV)

A comparação quantitativa entre baseline e NSGA-II usa a métrica de hipervolume em 2D, considerando apenas:

- tempo total (`time_total_s`);
- emissões (`emissions_g`).

Para cada cenário, ambas as frentes são primeiro filtradas para remover pontos dominados em 2D. De seguida, define-se um ponto de referência r_{ref} a partir da união das duas frentes (após filtragem), usando uma margem multiplicativa de 10% sobre os piores valores observados:

$$r_{\text{ref}} = (1.10 \cdot \max(\text{time\_total\_s}), 1.10 \cdot \max(\text{emissions\_g}))$$

O hipervolume é então calculado como a área dominada pela fronteira em relação a r_{ref}. Valores mais elevados indicam melhor compromisso global (maior convergência e/ou diversidade) no plano (tempo, emissões). O resumo global encontra-se em `outputs/experiments/hypervolume_summary.json`.

7.4. Justificação da métrica de comparação (HV em 2D)

Embora o sistema modele três objetivos (tempo, emissões e caminhada), a comparação quantitativa principal entre métodos utiliza hipervolume (HV) em 2D, considerando apenas (tempo total, emissões). Esta escolha é intencional e prende-se com:

1. Comparabilidade direta com o baseline: o método baseline (Dijkstra-λ) é construído a partir de uma combinação ponderada apenas de tempo e emissões, produzindo naturalmente uma fronteira no plano (tempo, CO_2). Usar HV 2D garante que ambos os métodos são comparados no mesmo espaço objetivo, sem introduzir uma dimensão adicional que o baseline não otimiza explicitamente.
2. Robustez e interpretabilidade da métrica: o cálculo de HV em 3D exigiria decisões adicionais (normalização e escalas entre objetivos, transformação do objetivo "exercício" para minimização, e escolha do ponto de referência em 3D). Essas escolhas podem influenciar significativamente o valor do hipervolume e dificultar a interpretação dos resultados, especialmente quando se pretende comparar com um baseline 2D.

Assim, o HV 2D é utilizado como métrica quantitativa principal para avaliar convergência e diversidade no compromisso tempo–CO_2. O objetivo ligado ao exercício (caminhada) é avaliado de forma complementar, através da análise qualitativa na interface e de estatísticas descritivas (por exemplo, distribuição de tempo a pé nas soluções não dominadas), mantendo a interpretação do terceiro objetivo alinhada com a perspetiva do utilizador.

7.5. Resultados quantitativos (HV) por cenário

A Tabela 1 sumariza os resultados de hipervolume para os 9 cenários exportados (3 **short**, 3 **mid**, 3 **long**). Observa-se que:

- em cenários **short**, baseline e NSGA-II tendem a coincidir (espaço de soluções pequeno);
- em cenários **mid** e **long**, o NSGA-II consegue, com maior frequência, ampliar a cobertura da fronteira (maior número de pontos não dominados e HV superior).

Cenário	Classe	$\lvert P\rvert_{\text{walk}}$	$\lvert P\rvert_{\text{shortest}}$	HV baseline	HV NSGA-II
short_000	short	1	2	1107.05	1107.05
short_001	short	3	3	7153.25	7153.25
short_002	short	1	1	0.74	0.74
mid_000	mid	7	5	15816.74	33880.73
mid_001	mid	6	2	620.46	620.46
mid_002	mid	8	7	7642.59	10991.55
long_000	long	17	7	5811.66	7163.75
long_001	long	14	5	12271.38	57085.89
long_002	long	20	4	880.59	880.59

[Figura 6 – Tabela 1: Hipervolume (2D) por cenário. $\lvert P\rvert_{\text{walk}}$ indica o nº de arestas do **random walk** do cenário e $\lvert P\rvert_{\text{shortest}}$ o nº de arestas do caminho mais curto por tempo.]

Globalmente, no conjunto de 9 cenários, o NSGA-II supera o baseline em 4 cenários (44.4%), obtendo uma média de hipervolume superior (baseline: 5700.50; NSGA-II: 13209.33). A melhoria ocorre sobretudo em cenários **mid** e **long**, onde existe maior combinatória multimodal e maior diversidade potencial de soluções.

7.6. Resultados qualitativos na interface

A melhor maneira de averiguar o sucesso da solução criada é para o mesmo ponto de origem e de destino, verificar as rotas que são geradas consoante o peso do objetivo pretendido (tempo ou CO2 ou exercício).

Mantendo inalteráveis as opções avançadas (tamanho da população, número de gerações, limite total a pé, transbordos máximos e política de caminhada), executou-se o NSGA-II para gerar um conjunto de soluções não-dominadas para o par origem–destino escolhido.

De seguida, variaram-se as preferências do utilizador na interface (Tempo/CO_2/Exercício e pesos nos sliders). Estas preferências não alteram a fronteira gerada, mas selecionam automaticamente (ou reordenam) uma solução dentro do conjunto obtido, permitindo observar diferentes compromissos entre tempo total, emissões e caminhada para o mesmo problema.

Figura 3: Preferência no tempo total.

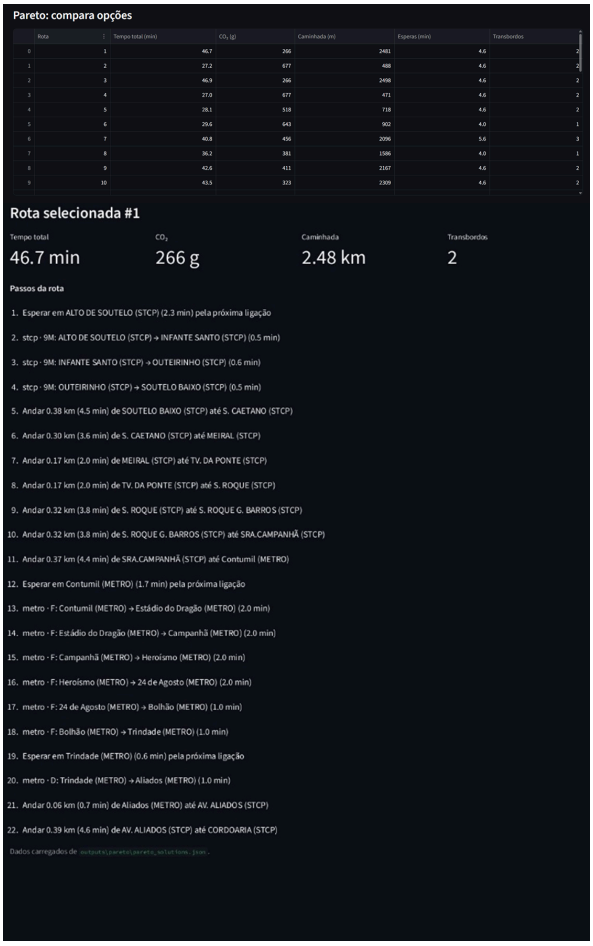

Figura 4: Preferência nas emissões de CO2.

Como podemos constatar, as rotas geradas consoante a importância do fator definido estão de acordo com o objetivo pretendido. É de realçar que a margem entre os valores da distância

da caminhada é pequena se compararmos os resultados entre os cenários onde o objetivo é o exercício e a menor emissão possível de dióxido de carbono. Isto acontece porque o meio de locomoção menos poluente é a caminhada e como resultado disso, quando se pretende minimizar as emissões de CO2, a distância percorrida a pé terá um grande impacto na escolha da rota.

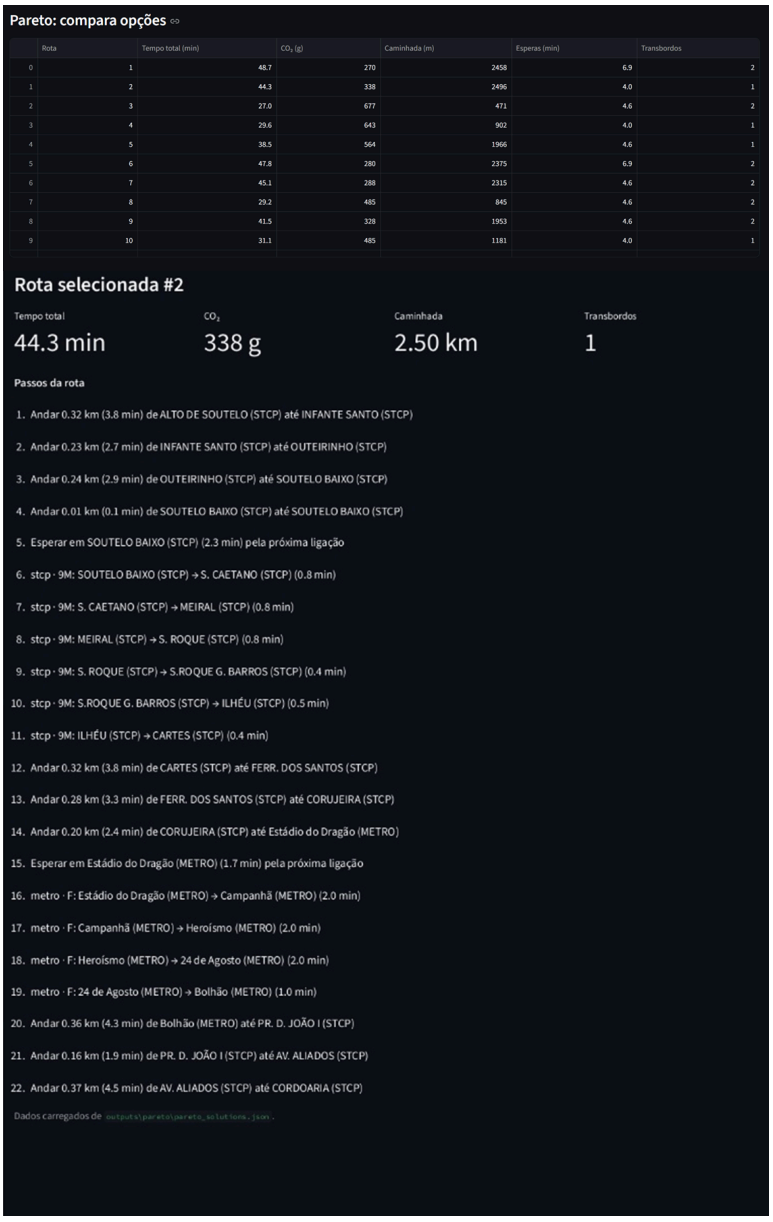

Figura 5: Preferência no exercício.

Para avaliar quantitativamente a qualidade das fronteiras Pareto obtidas, foi utilizada a métrica de hipervolume. O hipervolume mede simultaneamente a convergência e a diversidade das soluções não dominadas, calculando a área dominada pela fronteira em relação a um ponto de referência. Valores mais elevados de hipervolume indicam um melhor compromisso global entre os objetivos considerados. Esta métrica foi utilizada para comparar as soluções obtidas pelo NSGA-II com um baseline baseado em caminhos determinísticos. O hipervolume é calculado sobre a fronteira 2D (tempo total, emissões).

8. Discussão

A implementação de um planeador multimodal baseado em algoritmos evolutivos permitiu explorar soluções de mobilidade que escapam à lógica tradicional dos algoritmos de caminho mais curto uni objetivo. A análise dos resultados obtidos e do processo de desenvolvimento revela tanto as mais-valias da abordagem como as limitações inerentes à modelação.

8.1. O que o sistema faz bem

- diversidade de soluções: Uma das principais vantagens observadas no sistema é a capacidade de apresentar uma Fronteira de Pareto rica e diversificada. Ao contrário dos planeadores comerciais, que tendem a convergir apenas para a rota mais rápida, o nosso sistema identificou consistentemente alternativas onde um pequeno aumento no tempo de viagem resultava numa redução significativa das emissões de CO_2 (por exemplo, preferindo o Metro ao autocarro em trajetos longos) ou num aumento saudável da atividade física.
- Inicialização inteligente vs. Aleatória: A introdução de uma população inicial "semeada" com soluções heurísticas (Dijkstra) provou ser essencial. Nos testes preliminares com inicialização puramente aleatória, o algoritmo despendia as primeiras gerações a tentar encontrar caminhos válidos, resultando numa convergência lenta. A inicialização híbrida permitiu que o algoritmo evolutivo se focasse, desde o início, no refinamento e combinação de rotas já viáveis.

8.2. Limitações

- Natureza estática dos dados (GTFS): o modelo baseia-se em horários planeados e não em tempo real. Não considera atrasos operacionais, congestionamento de trânsito (crítico para a STCP) ou supressão de viagens.
- Regras pedonais simplificadas: Assumir uma velocidade de caminhada fixa de 1.4 m/s ignora a variabilidade humana (idade, condição física) e a topografia acidentada do Porto. O esforço de caminhar 500 metros numa subida íngreme é modelado de forma igual a 500 metros em plano, o que pode enviesar a atratividade das rotas a pé.
- Penalização de transferência: o custo de transferência foi modelado como um tempo fixo ou uma penalização no fitness. Embora funcional, este modelo não captura a complexidade real de mudar de cais numa estação grande como a Trindade versus uma paragem de autocarro simples.

8.3. Impacto das escolhas algorítmicas

O desempenho do planeador não depende apenas da formulação do problema, mas também da configuração intrínseca do algoritmo evolutivo.

A utilização de uma população inicial híbrida, contendo soluções extremas geradas por Dijkstra (menor tempo, menor CO_2), teve um impacto determinante na métrica de Hipervolume inicial.

Sem seeding o algoritmo inicia com um Hiper volume baixo e despende de várias gerações apenas para encontrar rotas topologicamente válidas, desperdiçando esforço computacional.

Com seeding o algoritmo inicia a evolução já na proximidade da Fronteira de Pareto real. Isto permitiu reduzir o número necessário de gerações, focando a pesquisa no preenchimento das lacunas (soluções de compromisso) entre os extremos pré-calculados.

Foram realizados testes de sensibilidade com diferentes tamanhos de população {20, 60, 100). Observou-se que populações muito pequenas (N=20) tendem a perder diversidade genética, convergindo prematuramente para mínimos locais (e.g., focando-se apenas em rotas de Metro e ignorando combinações complexas com autocarro).

9. Conclusão

O presente projeto alcançou com sucesso o objetivo de desenvolver um planeador de rotas multimodal para a área do Grande Porto, aplicando técnicas de Computação Inspirada na Natureza para resolver um problema complexo de Otimização Multi-Objetivo.

Ao integrar dados reais GTFS do Metro do Porto e da STCP com um modelo de grafo pedonal detalhado, foi possível construir uma ferramenta que ultrapassa a lógica binária dos planeadores tradicionais. A utilização do algoritmo evolutivo NSGA-II, complementada por uma estratégia de inicialização inteligente baseada em heurísticas, provou ser uma abordagem eficaz para identificar a Fronteira de Pareto num espaço de pesquisa vasto.

Os resultados obtidos demonstram que não existe uma única "melhor rota" absoluta. O sistema revelou consistentemente soluções de compromisso valiosas, onde pequenos sacrifícios no tempo de viagem (e.g., +5 minutos) podem resultar em ganhos significativos na redução da pegada ecológica (e.g., −30% emissões CO_2) ou no aumento da atividade física. A interface desenvolvida permite ao utilizador final navegar nestes trade-offs de forma intuitiva, promovendo uma mobilidade mais consciente e sustentável.

10. Referências

1. K. Deb, A. Pratap, S. Agarwal e T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, vol. 6, n.º 2, pp. 182–197, abr. 2002. doi: 10.1109/4235.996017.
2. Q. Zhang e H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition", *IEEE Transactions on Evolutionary Computation*, vol. 11, n.º 6, pp. 712–731, dez. 2007. doi: 10.1109/TEVC.2007.892759.
3. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca e V. G. da Fonseca, "Performance Assessment of Multiobjective Optimizers: An Analysis and Review", *IEEE Transactions on Evolutionary Computation*, vol. 7, n.º 2, pp. 117–132, abr. 2003. doi: 10.1109/TEVC.2003.810758.
4. E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, vol. 1, pp. 269–271, 1959. doi: 10.1007/BF01386390.
5. R. W. Sinnott, "Virtues of the Haversine", *Sky and Telescope*, vol. 68, n.º 2, pp. 158–159, dez. 1984.
6. F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau e C. Gagné, "DEAP: Evolutionary Algorithms Made Easy", *Journal of Machine Learning Research*, vol. 13, n.º 70, pp. 2171–2175, 2012.
7. A. A. Hagberg, D. A. Schult e P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX", em *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, G. Varoquaux, T. Vaught e J. Millman (eds.), Pasadena, CA, EUA, pp. 11–15, ago. 2008.

8. The pandas development team, “pandas-dev/pandas: Pandas”, Zenodo, 2020. doi: 10.5281/zenodo.3509134.
9. W. McKinney, “Data Structures for Statistical Computing in Python”, em *Proceedings of the 9th Python in Science Conference (SciPy 2010)*, S. van der Walt e J. Millman (eds.), pp. 56–61, 2010. doi: 10.25080/Majora-92bf1922-00a.
10. “Streamlit documentation”. Acedido em: 23 de dezembro de 2025. [Online]. Disponível em: https://docs.streamlit.io/
11. “Reference — General Transit Feed Specification (GTFS)”. Acedido em: 7 de dezembro de 2025. [Online]. Disponível em: https://gtfs.org/documentation/schedule/reference/
12. “Conjunto de Dados — Portal de Dados (GTFS: STCP e Metro do Porto)”. Acedido em: 7 de dezembro de 2025. [Online]. Disponível em: https://opendata.porto.digital/dataset/?q=Infraestruturas+e+Mobilidade&res_format=GTFS
13. “Análise de Sustentabilidade — Metro do Porto”. Acedido em: 23 de dezembro de 2025. [Online]. Disponível em: https://www.metrodoporto.pt/pages/358
14. “STCP — Sustentabilidade / Política Energética”. Acedido em: 23 de dezembro de 2025. [Online]. Disponível em: https://www.stcp.pt/pt/institucional/sustentabilidade/politica-energetica/