

Relatório Final — Modelos de Otimização de Rotas para Apoio ao Uso Profissional de Mota Elétrica

**Aluno:** Rodrigo José Gomes Fernandes  
**Orientador:** Tiago Pinto  
**Coorientadores:** Arsénio Reis, Eduardo Pires

**Prefácio**

O presente relatório foi elaborado no âmbito do projeto final de curso da Licenciatura em Engenharia Informática da Universidade de Trás-os-Montes e Alto Douro (UTAD). Este trabalho insere-se no contexto do projeto A-MoVeR — *Advanced Mobility for e-Delivery Routing* — uma iniciativa colaborativa que visa transformar a mobilidade urbana através da adoção de tecnologias sustentáveis, como o uso profissional de motas elétricas.

Ao longo deste projeto, tive a oportunidade de aplicar conhecimentos adquiridos ao longo do curso em contextos reais, desenvolvendo um sistema de otimização de rotas baseado em algoritmos computacionais, adaptado a cenários com múltiplas restrições, como autonomia limitada, capacidade de carga, janelas de tempo e variabilidade nas condições logísticas.

Este relatório não representa apenas a conclusão de um ciclo académico, mas também o início de uma nova etapa de maturidade técnica e profissional. A componente prática do projeto foi essencial para consolidar competências em desenvolvimento de software, modelação de problemas, análise de dados e trabalho em equipa — uma vez que este sistema foi desenvolvido em articulação com outros projetos complementares, incluindo o frontend da plataforma, a base de dados e a API de integração de serviços.

Agradeço aos meus orientadores, professores Tiago Pinto e Arsénio Reis, pelo acompanhamento constante, bem como às restantes equipas do A-MoVeR pela colaboração e partilha de conhecimento ao longo do percurso. Esta experiência demonstrou, de forma concreta, como a Engenharia Informática pode ser aplicada com impacto direto na inovação tecnológica e na sustentabilidade das cidades.

Bottom of Form

Índice

[1. Introdução 4](#_Toc200562739)

[2. Fundamentação Teórica 6](#_Toc200562740)

[2.1. Problemas de Roteamento de Veículos 6](#_Toc200562741)

[2.2. Desafios Específicos com Veículos Elétricos 7](#_Toc200562742)

[2.3. Técnicas de Resolução 8](#_Toc200562743)

[2.3.1. Algoritmos Exatos 8](#_Toc200562744)

[2.3.2. Algoritmos Heurísticos 9](#_Toc200562745)

[2.3.3. Meta-Heurísticas 10](#_Toc200562746)

[2.4. Métricas de Avaliação 11](#_Toc200562747)

[3. Arquitetura e Funcionamento do Sistema de Otimização 12](#_Toc200562748)

[3.1. Arquitetura Geral 12](#_Toc200562749)

[3.2. Ciclo de Funcionamento 12](#_Toc200562750)

[3.3. Requisitos Técnicos 13](#_Toc200562751)

[3.4. Modularidade e Expansão 14](#_Toc200562752)

[3.5. Considerações sobre Integração 14](#_Toc200562753)

[3.6. Exemplo de Execução Completa 14](#_Toc200562754)

[4. Cenários Reais e Histórias de Aplicação 14](#_Toc200562755)

[Cenário 1 – Rede Local de Distribuição Alimentar (DelivFresh) 15](#_Toc200562756)

[Cenário 2 – Recolha de Análises Clínicas (MobileSaúde) 15](#_Toc200562757)

[Cenário 3 – Assistência Técnica Rápida (TecFast) 16](#_Toc200562758)

[Cenário 4 – Logística com Múltiplos Depósitos (EcoPost) 16](#_Toc200562759)

[Cenário 5 – Visita Técnica a Microempresas (ProLocal) 17](#_Toc200562760)

[Cenário 6 – Entregas Expresso em Zonas Próximas (FastDrop) 17](#_Toc200562761)

[5. Testes e Resultados 19](#_Toc200562762)

[5.1. Resultados por Cenário 19](#_Toc200562763)

[5.2. Análise Geral 20](#_Toc200562764)

[5.3. Integração com a API e Automação da Escolha do Algoritmo 21](#_Toc200562765)

[5.4. Conclusão dos Testes 22](#_Toc200562766)

[6. Discussão 24](#_Toc200562767)

[6.1. Investigação e Fundamentação Científica 24](#_Toc200562768)

[6.2. Escolha e Adequação dos Algoritmos 25](#_Toc200562769)

[6.3. Desafios Técnicos 25](#_Toc200562770)

[6.4. Contributo para a Plataforma A-MoVeR 25](#_Toc200562771)

[6.5. Reflexão Final 26](#_Toc200562772)

[7. Conclusão e trabalhos futuros 26](#_Toc200562773)

[8. Bibliografia 28](#_Toc200562774)

# Introdução

O projeto **A-MoVeR** (*Advanced Mobility for e-Delivery Routing*) integra-se na *Agenda Mobilizadora para o Desenvolvimento de Produtos e Sistemas Inteligentes de Mobilidade Verde*, uma iniciativa nacional focada na inovação em mobilidade elétrica e sistemas urbanos inteligentes. Com base em Vila Real, o A-MoVeR reúne instituições académicas, empresas e centros de investigação em torno de três pilares fundamentais: comunicação veicular (V2X), motociclos elétricos de alta autonomia, e plataformas digitais de gestão logística inteligente.

Neste contexto, este projeto de licenciatura enquadra-se na vertente logística do A-MoVeR, coordenada pelo Prof. Tiago Pinto (UTAD), e tem como objetivo o desenvolvimento de um **sistema de otimização de rotas** especificamente desenhado para frotas de motas elétricas. A proposta passa pela aplicação de algoritmos de otimização a cenários urbanos realistas, considerando restrições relevantes como **autonomia de bateria, capacidade de carga, janelas temporais e pedidos dinâmicos**.

O sistema desenvolvido permite simular diferentes situações operacionais e comparar o desempenho de algoritmos exatos, heurísticos e meta-heurísticos, como **Branch and Bound, Dijkstra, Savings, Nearest Neighbor, GRASP e Tabu Search**. Para além de funcionar de forma autónoma para testes e análise de resultados, o sistema foi planeado para futura **integração com a infraestrutura tecnológica do A-MoVeR**, permitindo interoperabilidade com a base de dados logística, a API de serviços e o frontend de gestão.

**Integração com Outros Projetos do A-MoVeR**

Este projeto está inserido num esforço conjunto que envolve outras três equipas de licenciatura, cada uma com responsabilidades específicas no desenvolvimento da plataforma completa de gestão de motas elétricas:

* **Frontend da Plataforma Web**: desenvolvimento de uma interface de gestão acessível aos operadores logísticos, com funcionalidades como gestão de motas, tarefas e rotas, bem como visualização de alertas e manutenção.
* **Backend e Base de Dados**: modelação e implementação da infraestrutura de dados que suporta a plataforma, garantindo integridade, escalabilidade e segurança na gestão de utilizadores, serviços e histórico de operações.
* **API de Serviços de Otimização de Rotas**: criação de endpoints REST para expor os algoritmos de otimização como serviços consumíveis pelo frontend e pelo backend, possibilitando a integração em tempo real dos cálculos de rotas.

O sistema desenvolvido neste relatório assume, assim, o papel de **núcleo algorítmico** da plataforma logística do A-MoVeR, providenciando a lógica de decisão que será acionada por outros módulos da arquitetura distribuída.

**Objetivos do Projeto**

O trabalho tem como objetivos principais:

* Desenvolver uma aplicação modular para otimização de rotas com motas elétricas;
* Implementar e comparar diferentes algoritmos aplicáveis a contextos logísticos urbanos;
* Criar uma base de testes com cenários realistas;
* Preparar o sistema para futura integração com a plataforma A-MoVeR;
* Contribuir para a eficiência energética e operacional da mobilidade urbana profissional.

**Estrutura do Relatório**

Este relatório está organizado da seguinte forma:

1. **Fundamentação Teórica** – conceitos e algoritmos aplicados à otimização de rotas;
2. **Arquitetura e Funcionamento do Sistema de Otimização** – descrição da arquitetura do sistema, scripts, interface e funcionamento;
3. **Cenários Reais** – apresentação de cenários logísticos urbanos com diferentes restrições;
4. **Testes e Resultados** – análise comparativa do desempenho dos algoritmos aplicados;
5. **Discussão** – interpretação crítica dos resultados, limitações e oportunidades;
6. **Conclusão e Trabalhos Futuros** – encerramento do projeto e propostas de evolução;
7. **Anexos e Referências** – materiais complementares, estruturas de dados e bibliografia.

# Fundamentação Teórica

A otimização de rotas é um ramo da investigação operacional e da ciência da computação que procura resolver problemas de logística e transporte através da definição de percursos ideais para um ou mais veículos. No contexto atual de mobilidade elétrica, a necessidade de adaptação desses algoritmos a veículos com restrições específicas — como a autonomia limitada da bateria, a necessidade de recarga e a capacidade de carga reduzida — torna este tema ainda mais desafiador e relevante.

# 2.1. Problemas de Roteamento de Veículos

O problema de roteamento de veículos (VRP — *Vehicle Routing Problem*) é um problema clássico de otimização combinatória que visa determinar o conjunto de rotas mais eficientes para uma frota de veículos que deve atender a um conjunto de clientes, respeitando restrições como capacidade, tempo de entrega e número de veículos. O VRP é uma generalização do problema do caixeiro-viajante (TSP — *Travelling Salesman Problem*), que busca a menor rota possível que visite todos os pontos e retorne ao ponto de origem.

No contexto deste projeto, foram consideradas várias variantes do VRP, incluindo:

* **Problema de Rotas de Veículos (VRP)**: Neste cenário, há um grupo de veículos designados para entregar mercadorias aos clientes, cada um com as suas exigências únicas. O objetivo é minimizar a distância total percorrida ou os custos incorridos. Métodos de Solução: uma variedade de técnicas pode ser utilizada, incluindo abordagens como branch and bound, métodos práticos como algoritmos vizinhos e genéticos, e estratégias avançadas como recozimento simulado e busca tabu.
* **Problema do Caixeiro Viajante (TSP)**: Este problema envolve encontrar uma rota que visite um conjunto específico de cidades exactamente uma vez antes de retornar à cidade de partida. Aplicações: originalmente projectado para vendedores, também pode ser adaptado para optimizar rotas de entrega onde um único veículo precisa visitar destinos de forma eficiente. Métodos de Solução: semelhante ao VRP, o TSP pode ser abordado usando métodos práticos e estratégias avançadas.
* **Problema de Rotas de Veículos Dinâmico (DVRP)**: Nesta variação, novos pedidos de entrega surgem dinamicamente ao longo do dia. As rotas dos veículos precisam ser ajustadas em tempo real para acomodar esses pedidos, minimizando o tempo de viagem ou os custos. Métodos de Solução: as técnicas utilizadas incluem programação, algoritmos online e métodos de otimização em tempo real.
* **Problema de Rotas de Veículos Dependente do Tempo (TDVRP)**: Foca em tempos de viagem que variam de acordo com a hora do dia, influenciados por fatores como padrões de tráfego. Isso é particularmente útil para planear rotas de entrega que experienciam diferentes durações de viagem em diferentes partes do dia. Para resolver esse problema, são utilizados algoritmos especializados, como os algoritmos Dijkstra.
* **Problema de Rotas de Veículos com Múltiplos Depósitos (MDVRP)**: O MDVRP tem vários depósitos dos quais os veículos iniciam e terminam as suas rotas, e um grupo de clientes associado a cada depósito. É mais eficiente quando as mercadorias são mantidas em vários armazéns em diferentes regiões. Os algoritmos envolvidos na sua solução também são VRP.
* **Problema de Rotas de Veículos com Capacidade (CVRP)**: A necessidade do CVRP surge quando cada veículo numa organização tem uma capacidade para transportar e entregar numa única viagem. Os algoritmos utilizados são semelhantes aos do VRP, mas impõem algumas restrições de linearidade entre o peso da carga a ser transportada e o que os veículos podem manusear.

Estes problemas são considerados NP-difíceis, ou seja, a sua complexidade aumenta exponencialmente com o número de clientes ou restrições envolvidas, o que exige a aplicação de algoritmos específicos para encontrar boas soluções em tempo aceitável.

# 2.2. Desafios Específicos com Veículos Elétricos

A utilização de veículos elétricos em logística urbana introduz um conjunto adicional de restrições que tornam a otimização das rotas ainda mais complexa:

* **Problema de Rotas de Veículos com Limite de Bateria (BCVRP):** Cada motocicleta elétrica tem uma capacidade de bateria que limita a distância que pode percorrer antes de precisar recarregar. O objetivo é otimizar as rotas para minimizar a distância total percorrida ou os custos, tendo em conta os pontos de recarga disponíveis. Métodos de Solução: Podem ser utilizados algoritmos como branch and bound, algoritmos genéticos, recozimento simulado e busca tabu, com a adição de restrições de capacidade de bateria.
* **Problema de Rotas de Veículos com Janelas de Tempo e Recarga (TWCVRP):** Além da capacidade limitada da bateria, as entregas devem ser feitas dentro de janelas de tempo específicas. É necessário incluir paragens para recarga da bateria nas rotas, que também podem ter horários de funcionamento limitados. Métodos de Solução: Algoritmos especializados como Dijkstra de Tempo, A\*, e técnicas de otimização em tempo real podem ser usados para lidar com as restrições temporais e de recarga.
* **Problema de Rotas de Veículos Dinâmico com Recarga (DVRPC):** Novos pedidos de entrega surgem ao longo do dia, e as rotas dos veículos precisam ser ajustadas em tempo real, considerando a necessidade de recarga das baterias. Métodos de Solução: Algoritmos online e métodos de otimização em tempo real, como recálculo dinâmico de rotas com pontos de recarga, são essenciais para este problema.
* **Problema de Rotas de Veículos com Capacidade e Recarga (CCVRP):** Cada motocicleta elétrica tem uma capacidade de carga limitada e precisa de recarga após percorrer uma certa distância. O objetivo é otimizar as rotas considerando tanto a capacidade de carga como a necessidade de recarga. Métodos de Solução: Técnicas combinadas de VRP com capacidade e restrições de recarga de bateria, como algoritmos genéticos e busca tabu, são úteis para esta abordagem.
* **Problema de Rotas de Veículos com Depósitos Múltiplos e Recarga (MDVRPC):** Existem múltiplos depósitos dos quais as motas iniciam e terminam as suas rotas, e cada depósito possui pontos de recarga. O desafio é otimizar as rotas considerando a distribuição dos depósitos e a necessidade de recarga. Métodos de Solução: Algoritmos especializados para MDVRP, ajustados para incluir paragens de recarga e capacidade de bateria, são necessários para resolver este problema.
* **Problema de Rotas de Veículos Dependente do Tempo com Recarga (TDVRPC):** Os tempos de viagem variam de acordo com a hora do dia, influenciados por fatores como padrões de tráfego, e a necessidade de recarga das baterias deve ser levada em conta. Métodos de Solução: Algoritmos Dijkstra de Tempo ou A\* modificados para incluir restrições de recarga e padrões de tráfego variáveis.
* **Problema do Caixeiro Viajante com Limite de Bateria (BCTSP):** Encontrar uma rota que visite um conjunto específico de cidades exatamente uma vez, antes de retornar à cidade de partida, com a restrição de que a motocicleta deve ser recarregada após percorrer uma certa distância. Métodos de Solução: Algoritmos práticos e estratégias avançadas como recozimento simulado e busca tabu, com restrições adicionais para pontos de recarga.

Estas particularidades fazem com que o VRP adaptado para veículos elétricos (conhecido como EVRP — *Electric Vehicle Routing Problem*) exija modelos de otimização que combinem lógica matemática, técnicas heurísticas e informação contextual (tempo, tráfego, localização de postos de carregamento).

# 2.3. Técnicas de Resolução

Neste projeto foram aplicadas e comparadas várias técnicas de otimização divididas em três grupos principais: algoritmos exatos, heurísticos e meta-heurísticos. A seguir descrevem-se os algoritmos implementados.

# 2.3.1. Algoritmos Exatos

Os métodos exatos são algoritmos de pesquisa exaustiva que analisam todo o conjunto de soluções possíveis de um determinado problema até encontrar a solução ótima. A principal vantagem destes métodos é que garantem a obtenção da solução ótima. No entanto, a sua modelação torna-se mais complexa e, quando aplicados a problemas mais complexos ou de maior dimensão, enfrentam grandes dificuldades para encontrar a solução ótima num intervalo de tempo adequado. Consequentemente, o esforço computacional necessário para a sua resolução cresce exponencialmente.

Alguns exemplos desses algoritmos são:

* Algoritmo Branch and Bound
* Algoritmo de Dijkstra

**Branch and Bound**  
O algoritmo Branch and Bound é um método exato de otimização combinatória utilizado para encontrar soluções ótimas em problemas como o TSP ou o VRP. A sua estratégia baseia-se em dividir o espaço de soluções (ramificação) e eliminar subconjuntos que não podem conter a solução ótima (poda), com base em limites inferiores e superiores previamente calculados.

Este algoritmo garante a solução ótima, mas o seu desempenho decresce drasticamente à medida que o número de clientes aumenta. No projeto A-MoVeR, foi utilizado em cenários com até 6 clientes, onde ainda era computacionalmente viável.

# 2.3.2. Algoritmos Heurísticos

Os algoritmos heurísticos são amplamente utilizados para resolver problemas de otimização de rotas, como o Problema do Caixeiro Viajante (TSP) e o Problema de Roteamento de Veículos (VRP). São projetados para encontrar soluções aproximadas ou subótimas em um tempo razoável, sendo especialmente úteis quando métodos exatos se tornam impraticáveis devido ao tamanho do problema.

**Vantagens dos Algoritmos Heurísticos**

* **Rápida Convergência:** São capazes de encontrar soluções de boa qualidade em um número limitado de iterações.
* **Adaptação:** Podem ser ajustados e combinados para lidar com diferentes variações e complexidades dos problemas de otimização de rotas.
* **Aplicabilidade:** São aplicáveis a uma ampla gama de problemas do mundo real, oferecendo soluções viáveis em tempo hábil.

**Limitações dos Algoritmos Heurísticos**

* **Soluções Subótimas:** Não garantem a solução ótima, pois focam em encontrar soluções aceitáveis dentro de limitações de tempo e recursos.
* **Dependência de Parâmetros:** O desempenho pode depender da escolha adequada de parâmetros como tamanho da vizinhança, taxa de resfriamento, entre outros.

Os algoritmos heurísticos fornecem uma abordagem flexível e eficaz para resolver problemas desafiadores de otimização de rotas, encontrando soluções aceitáveis em um tempo viável e adaptando-se às complexidades e demandas específicas de diferentes problemas de roteamento.

Alguns exemplos de algoritmos Heurísticos são:

* Algoritmo Savings
* Algoritmo de Nearest Neighbor

**Dijkstra**  
O algoritmo de Dijkstra é um algoritmo clássico de caminhos mínimos em grafos, utilizado para encontrar a menor distância entre dois nós. Apesar de não resolver o VRP completo, pode ser útil em cenários simples de ponto-a-ponto ou como componente auxiliar de algoritmos mais complexos.

No A-MoVeR, foi utilizado em cenários de assistência técnica rápida, onde o objetivo era chegar ao destino com o menor consumo energético possível.

**Savings Algorithm (Clarke & Wright)**  
O algoritmo de Savings é uma heurística construtiva que começa com uma rota individual para cada cliente e vai combinando rotas vizinhas com base na economia gerada por essa junção. O critério de "saving" calcula o quanto se poupa em termos de distância ao agrupar dois clientes.

É um algoritmo eficiente para problemas com múltiplos depósitos e baixa complexidade de restrições.

**Nearest Neighbor (Vizinho Mais Próximo)**  
Este algoritmo parte do nó inicial e, a cada passo, visita o cliente mais próximo que ainda não foi atendido. É uma heurística muito simples e rápida, mas pode gerar soluções subótimas, principalmente em problemas com mais restrições.

No entanto, a sua simplicidade é vantajosa em contextos onde o tempo de decisão é crítico. No projeto, foi utilizado com a técnica **2-opt** para melhorar localmente a solução inicial.

# 2.3.3. Meta-Heurísticas

Os algoritmos de meta-heurística são técnicas poderosas e flexíveis para resolver problemas complexos de otimização de rotas, como o Problema do Caixeiro Viajante (TSP) e o Problema de Roteamento de Veículos (VRP). Eles são especialmente úteis quando métodos exatos se tornam impraticáveis devido ao tamanho do problema ou à necessidade de encontrar soluções aceitáveis em um tempo razoável.

**Aplicações e Considerações**

* **Complexidade do Problema:** Cada algoritmo de meta-heurística tem suas próprias vantagens e limitações, sendo adequado para diferentes tipos de problemas e requisitos de desempenho.
* **Parâmetros e Ajustes:** A performance dos algoritmos de meta-heurística pode depender da escolha adequada de parâmetros como taxa de mutação, tamanho da lista tabu, entre outros.
* **Adaptação:** Podem ser adaptados e combinados para resolver problemas específicos de otimização de rotas, como VRP com múltiplos depósitos ou restrições temporais.

Alguns exemplos destes algoritmos são:

* Algoritmo de Tabu Shearch
* Algoritmo Greedy Randomized Adaptive Search Procedure (GRASP)

**Tabu Search**  
A Busca Tabu é uma meta-heurística que explora o espaço de soluções através de movimentos iterativos, mantendo uma "memória tabu" de soluções ou transições proibidas temporariamente, de modo a evitar ciclos e zonas de ótimos locais.

Mostrou-se particularmente eficaz em cenários com múltiplas restrições e elevada complexidade combinatória, como rotas com janelas de tempo e limites de bateria.

**GRASP (Greedy Randomized Adaptive Search Procedure)**  
GRASP combina uma fase construtiva (com seleção aleatória guiada) e uma fase de melhoria local, repetindo o processo várias vezes para encontrar boas soluções. A aleatoriedade controlada permite escapar a ótimos locais e obter soluções de qualidade em tempo reduzido.

Foi utilizado em cenários com janelas temporais, onde era necessário considerar ordens específicas de atendimento sem comprometer a eficiência.

# 2.4. Métricas de Avaliação

Para avaliar o desempenho dos algoritmos aplicados, foram definidas as seguintes métricas:

* **Distância total percorrida** (em km): principal métrica de eficiência energética;
* **Tempo total de execução** (em segundos): importante para cenários dinâmicos;
* **Número de visitas ao depósito**: mede a necessidade de recarregamento ou reposição;
* **Respeito às restrições**: capacidade do algoritmo em cumprir janelas de tempo, limites de carga e autonomia;
* **Robustez e escalabilidade**: capacidade de adaptação a alterações e aumento da complexidade.

Estas métricas foram aplicadas a um conjunto de cenários com diferentes características, detalhados no capítulo seguinte.

# Arquitetura e Funcionamento do Sistema de Otimização

O sistema desenvolvido neste projeto tem como objetivo centralizar a gestão de algoritmos de otimização de rotas aplicados ao uso profissional de motas elétricas, permitindo a execução de testes, simulações e comparações de forma modular e extensível. Foi concebido para funcionar de forma independente, mas com uma arquitetura preparada para integração em plataformas maiores, como a solução logística do projeto A-MoVeR.

# 3.1. Arquitetura Geral

O sistema está organizado em pastas e ficheiros com funções bem definidas. A estrutura foi projetada para facilitar a adição de novos algoritmos e o tratamento de diferentes cenários a partir de dados uniformizados.

📁 algoritmos

├── branch\_and\_bound.py

├── dijkstra.py

├── grasp.py

├── nearest\_neighbor.py

├── savings.py

└── tabu\_search.py

📁 JSONS

└── [ficheiros .json gerados automaticamente a partir de Excel]

📄 cenarios.xlsx

📄 exceltoJSON.py

📄 interface\_algoritmos.py

# 3.2. Ciclo de Funcionamento

O sistema foi desenhado para ser flexível e funcional tanto em **modo manual (via terminal)** quanto através de uma **interface interativa**:

**Etapa 1 – Definição de Cenários no Excel**

O utilizador define os parâmetros dos cenários num ficheiro Excel com duas folhas:

* **cenários** – onde se configuram os cenários: nome, algoritmo, capacidade da mota, autonomia, etc.
* **nodes** – onde se definem os pontos de entrega ou recolha: coordenadas, tipo de tarefa, demanda, janelas temporais.

**Etapa 2 – Conversão de Excel para JSON**

O script exceltoJSON.py converte os dados do Excel para o formato JSON, utilizado pelos algoritmos.

* python exceltoJSON.py cenarios.xlsx

Também é possível converter apenas alguns cenários:

* python exceltoJSON.py cenarios.xlsx --cenarios 1 2 3

**Etapa 3 – Execução dos Algoritmos**

Os algoritmos podem ser executados diretamente a partir do terminal, utilizando os ficheiros JSON gerados:

* python branch\_and\_bound.py cenario\_1.json
* python nearest\_neighbor.py cenario\_2.json --two\_opt

**Etapa 4 – Interface Interativa**

O script interface\_algoritmos.py permite ao utilizador correr os algoritmos através de uma interface simples:

* Lista os cenários disponíveis;
* Gera automaticamente o JSON necessário;
* Corre o algoritmo definido para cada cenário;
* Valida a existência do script correspondente.
* python interface\_algoritmos.py

# 3.3. Requisitos Técnicos

O sistema foi desenvolvido em Python 3.8+ e utiliza as seguintes bibliotecas:

* pandas – para manipulação de dados estruturados;
* openpyxl – para leitura e escrita de ficheiros .xlsx;
* argparse e subprocess – bibliotecas nativas para linha de comandos e automação de scripts.

Instalação rápida:

* pip install pandas openpyxl

# 3.4. Modularidade e Expansão

A estrutura do sistema permite:

* Adicionar novos algoritmos no diretório /algoritmos, bastando nomear o ficheiro de acordo com a coluna do Excel;
* Atualizar ou criar cenários diretamente no Excel, sem necessidade de alterar código;
* Simular testes com dados reais ou fictícios;
* Preparar o sistema para **exposição via API**, conforme a integração com o projeto complementar.

# 3.5. Considerações sobre Integração

Apesar de ser uma aplicação autónoma, o sistema foi desenhado com vista à futura integração nos seguintes componentes do A-MoVeR:

* **Frontend da Plataforma Web** – o sistema poderá ser chamado por comandos no frontend para execução remota dos algoritmos e apresentação de resultados em tempo real;
* **API de Serviços** – os algoritmos poderão ser expostos como serviços REST, recebendo dados JSON e devolvendo a rota otimizada;
* **Base de Dados Logística** – os ficheiros JSON poderão ser substituídos por inserções e leituras diretas da base de dados através de ORM ou endpoints.

# 3.6. Exemplo de Execução Completa

**Cenário: Entrega Alimentar Urbana com Tabu Search**

1. Definido no Excel com coordenadas, demanda e janelas de tempo;
2. Convertido para JSON com exceltoJSON.py;
3. Executado com:
   * python tabu\_search.py cenario\_delivfresh.json
4. Resultado: rota otimizada com menor distância total e respetiva análise energética.

# Cenários Reais e Histórias de Aplicação

Para validar o sistema de otimização de rotas desenvolvido neste projeto, foram definidos e implementados vários **cenários realistas**, baseados em situações logísticas comuns numa cidade de média dimensão como Vila Urbana. Estes cenários foram construídos com dados representativos, incluindo coordenadas geográficas, demandas específicas, janelas temporais e restrições operacionais (capacidade de carga, autonomia energética e limites de tempo).

Cada cenário foi modelado com base numa **história realista** e testado com um algoritmo específico, escolhido pela sua adequação às características do problema.

# Cenário 1 – Rede Local de Distribuição Alimentar (DelivFresh)

**Contexto:**  
A empresa DelivFresh inicia as entregas de produtos frescos logo ao amanhecer, com o objetivo de manter a qualidade e frescura. As encomendas incluem pão quente, bolos e queijos regionais. O serviço tem de ser concluído antes das 08:00, respeitando restrições de tempo e autonomia.

**Configuração:**

* Partida: Armazém DelivFresh (nó 0) às 06:30
* Clientes: 6 estabelecimentos (nós 1 a 6)
* Capacidade da mota: 25 kg
* Autonomia: 10 kWh
* Janela temporal: 06:30 – 08:00

**Algoritmo aplicado:** *Tabu Search*  
**Justificação:** Adapta-se bem a problemas com múltiplas restrições e alta complexidade combinatória.

**Resultado:**

* Rota: [0, 1, 2, 3, 4, 5, 6, 0]
* Distância total: 45.38 km
* Energia estimada: 9.08 kWh

**Análise:**  
A rota cumpre todos os requisitos logísticos, respeita a capacidade da mota e garante a entrega dentro da janela de tempo.

# Cenário 2 – Recolha de Análises Clínicas (MobileSaúde)

**Contexto:**  
A unidade móvel do laboratório MobileSaúde recolhe amostras clínicas em várias instituições. As recolhas devem seguir uma ordem específica, respeitando janelas de tempo e mantendo o material em condições adequadas.

**Configuração:**

* Partida: Laboratório Central (nó 0) às 08:00
* Locais: 5 pontos com janelas de tempo (nós 1 a 5)
* Capacidade: 18 kg
* Autonomia: 10 kWh

**Algoritmo aplicado:** *GRASP*  
**Justificação:** Geração de múltiplas soluções com melhoria local, ideal para janelas de tempo.

**Resultado:**

* Rota: [0, 1, 5, 2, 4, 3, 0]
* Distância total: 28.88 km

**Análise:**  
Cumprimento total das janelas de tempo, capacidade e eficiência energética. Adequado para serviços sensíveis.

# Cenário 3 – Assistência Técnica Rápida (TecFast)

**Contexto:**  
Uma avaria urgente num painel solar ativa uma missão de assistência técnica com partida imediata. O objetivo é chegar ao local no menor tempo e distância possíveis, com bateria limitada.

**Configuração:**

* Partida: Garagem central (nó 0) às 15:12
* Destino: Escola Técnica (nó 3)
* Capacidade: não relevante
* Autonomia: 5 kWh
* Restrições: Rota direta e mínima

**Algoritmo aplicado:** *Dijkstra*  
**Justificação:** Cálculo do caminho mais curto entre dois pontos.

**Resultado:**

* Caminho: [0, 3]
* Distância: 8.6 km
* Energia estimada: 1.72 kWh

**Análise:**  
Solução ótima de ponto-a-ponto, simples e eficaz.

# Cenário 4 – Logística com Múltiplos Depósitos (EcoPost)

**Contexto:**  
A empresa EcoPost distribui materiais reciclados a partir de dois depósitos. Cada mota parte do seu depósito e realiza entregas específicas com foco na eficiência energética.

**Configuração:**

* Mota Norte (nó 0): Escola 1, Papelaria Laranja
* Mota Sul (nó 5): Escola 2, Papelaria Azul
* Capacidade: 30 kg por mota
* Autonomia: 8 kWh
* Objetivo: máxima eficiência e equilíbrio logístico

**Algoritmo aplicado:** *Savings (Clarke & Wright)*  
**Justificação:** Ideal para agrupamento local e economia de percurso em múltiplos depósitos.

**Resultado:**

* Norte: [0, 1, 2, 0] – 25.62 km
* Sul: [5, 3, 4, 5] – 21.04 km

**Análise:**  
Boa divisão de carga e percurso, rotas equilibradas e energeticamente eficientes.

# Cenário 5 – Visita Técnica a Microempresas (ProLocal)

**Contexto:**  
Serviço técnico de manutenção preventiva a pequenos comércios. O número de clientes é reduzido, permitindo uso de algoritmo exato.

**Configuração:**

* Partida: Oficina (nó 0) às 09:00
* Visitas: 6 locais até às 12:00
* Capacidade: irrelevante
* Autonomia: 10 kWh

**Algoritmo aplicado:** *Branch and Bound*  
**Justificação:** Ideal para instâncias pequenas, permitindo encontrar a solução ótima.

**Resultado:**

* Rota ótima: [0, 1, 3, 5, 4, 6, 2, 0]
* Distância total: 21.4 km

**Análise:**  
Solução ótima garantida com baixo tempo de execução e máxima eficiência.

# Cenário 6 – Entregas Expresso em Zonas Próximas (FastDrop)

**Contexto:**  
Serviço de entregas expresso em zonas densas, onde a decisão rápida é prioritária. O foco é encontrar rotas aceitáveis rapidamente.

**Configuração:**

* Partida: Centro logístico (nó 0) às 10:00
* Visitas: 6 clientes até às 12:00
* Capacidade: 20 kg
* Autonomia: 10 kWh

**Algoritmo aplicado:** *Nearest Neighbor (com 2-opt)*  
**Justificação:** Heurística rápida, melhorada localmente para solução eficiente.

**Resultado:**

* Rota: [0, 1, 6, 4, 5, 2, 3, 0]
* Distância: 17.63 km
* Energia estimada: 3.53 kWh

**Análise:**  
Solução gerada rapidamente, com distância aceitável. Ideal para contextos com baixa complexidade.

# Testes e Resultados

Os testes realizados neste projeto incidiram sobre seis cenários distintos, cada um representando uma situação realista do uso profissional de motas elétricas em contexto urbano. Para cada cenário, foi selecionado **um algoritmo de otimização específico**, de acordo com as características operacionais e restrições envolvidas (como janelas temporais, limitação de autonomia, múltiplos depósitos ou necessidade de resposta imediata).

Esta secção apresenta os **resultados obtidos** em cada cenário, com base em métricas como distância total percorrida, eficiência energética, tempo de execução e cumprimento das restrições logísticas.

# 5.1. Resultados por Cenário

**Cenário 1 – DelivFresh | Tabu Search**

* **Distância Total:** 45.38 km
* **Energia Estimada:** 9.08 kWh
* **Janelas de Tempo:** Cumpridas (06:30–08:00)
* **Capacidade da mota:** 25 kg
* **Observações:**
  + Algoritmo adaptou-se bem às exigências de entrega rápida e múltiplos pontos.
  + Cumprimento integral das restrições.
  + Boa eficiência na redução de distância.

**Cenário 2 – MobileSaúde | GRASP**

* **Distância Total:** 28.88 km
* **Janelas de Tempo:** Cumpridas para todos os locais
* **Capacidade da mota:** 18 kg
* **Observações:**
  + GRASP gerou rota compatível com todas as restrições temporais e de carga.
  + Alta flexibilidade para lidar com múltiplas combinações de visitas.
  + Resultados consistentes com eficiência aceitável.

**Cenário 3 – TecFast | Dijkstra**

* **Distância Total:** 8.6 km
* **Energia Estimada:** 1.72 kWh
* **Autonomia Disponível:** 5 kWh
* **Observações:**
  + Dijkstra foi ideal para encontrar o percurso mais curto até ao destino.
  + Adequado para rotas ponto-a-ponto em situações urgentes.
  + Tempo de execução muito baixo.

**Cenário 4 – EcoPost | Savings Algorithm**

* **Rota Norte:** [0, 1, 2, 0] → 25.62 km
* **Rota Sul:** [5, 3, 4, 5] → 21.04 km
* **Capacidade:** 30 kg por mota
* **Observações:**
  + Savings foi eficaz na divisão das rotas com base nos depósitos.
  + Boa economia de percurso e balanceamento entre as motas.
  + Resultado final com consumo energético equilibrado.

**Cenário 5 – ProLocal | Branch and Bound**

* **Rota:** [0, 1, 3, 5, 4, 6, 2, 0]
* **Distância Total:** 21.4 km
* **Restrições:** Sem janelas de tempo; foco em distância mínima
* **Observações:**
  + Algoritmo exato encontrou a solução ótima.
  + Tempo de execução aceitável, dado o número reduzido de nós.
  + Ideal para casos com pequena escala e elevado rigor.

**Cenário 6 – FastDrop | Nearest Neighbor + 2-opt**

* **Rota:** [0, 1, 6, 4, 5, 2, 3, 0]
* **Distância Total:** 17.63 km
* **Energia Estimada:** 3.53 kWh
* **Observações:**
  + Algoritmo rápido e eficiente para tomada de decisão imediata.
  + Aplicação do 2-opt melhorou a solução inicial.
  + Boa adequação para zonas densas com menor complexidade logística.

# 5.2. Análise Geral

A estratégia adotada no projeto priorizou a **escolha do algoritmo mais adequado para cada cenário**, em vez de aplicar o mesmo algoritmo em todos os casos. Essa abordagem permitiu:

* Ajustar cada técnica às restrições reais do cenário;
* Demonstrar a versatilidade do sistema em lidar com diferentes contextos logísticos;
* Avaliar, qualitativamente, o **comportamento de cada algoritmo em situações práticas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cenário** | **Algoritmo** | **Tipo de Problema** | **Justificação** |
| DelivFresh | Tabu Search | Múltiplas entregas + janela de tempo | Alta complexidade combinatória |
| MobileSaúde | GRASP | Janelas temporais e cargas variadas | Flexibilidade e eficiência rápida |
| TecFast | Dijkstra | Rota direta e urgente | Caminho mínimo ideal |
| EcoPost | Savings Algorithm | Múltiplos depósitos, rotas agrupadas | Economia de percurso com boa segmentação |
| ProLocal | Branch and Bound | Poucos clientes, sem janelas temporais | Solução ótima garantida |
| FastDrop | Nearest Neighbor | Decisão rápida com pontos próximos | Heurística simples e eficaz, com 2-opt |

# 5.3. Integração com a API e Automação da Escolha do Algoritmo

Embora neste relatório cada cenário tenha sido testado manualmente com um único algoritmo escolhido pelo autor com base na lógica do problema, o projeto foi concebido para funcionar **como parte de uma aplicação maior** — uma **plataforma de gestão logística inteligente** desenvolvida no âmbito do projeto A-MoVeR.

Nessa plataforma, a escolha do algoritmo **não será feita manualmente**. Em vez disso, **uma API dedicada fará essa seleção automaticamente**, com base nos parâmetros enviados pelo utilizador ou sistema gestor.

A lógica de funcionamento será a seguinte:

1. **O frontend da plataforma** (utilizado por operadores logísticos) permitirá a introdução ou consulta de tarefas, pedidos e configurações de entrega.
2. Estes dados serão enviados para a **API de otimização de rotas**, desenvolvida por uma equipa parceira no A-MoVeR.
3. A API analisará os parâmetros recebidos, como:
   * Tipo de operação (entrega, assistência, recolha, etc.);
   * Número de locais a visitar;
   * Existência ou não de janelas de tempo;
   * Capacidade e autonomia da mota;
   * Existência de múltiplos depósitos;
   * Necessidade de resposta imediata.
4. Com base nesses dados, a **API selecionará automaticamente o algoritmo mais adequado** para resolver o problema (por exemplo, Tabu Search para múltiplas restrições, Dijkstra para rotas diretas, etc.).
5. O algoritmo será executado no backend, utilizando os dados convertidos para JSON.
6. A **rota otimizada será devolvida à plataforma** para visualização e execução.

Desta forma, o sistema desenvolvido neste projeto atua como o **núcleo de decisão algorítmica**, enquanto a API e os módulos restantes fazem a mediação com o utilizador e o contexto operacional.

Este design modular permite:

* Separação clara entre lógica algorítmica e lógica de gestão;
* Facilidade de manutenção e substituição de algoritmos;
* Expansão futura para algoritmos com aprendizagem automática;
* Integração em sistemas logísticos de larga escala.

# 5.4. Conclusão dos Testes

Os testes realizados ao longo deste projeto demonstraram que a aplicação de algoritmos de otimização de rotas em cenários urbanos reais é não só viável como altamente eficaz no apoio à operação profissional de motas elétricas. A decisão de **associar cada algoritmo a um cenário específico**, com base nas características e restrições de cada problema, revelou-se acertada e produtiva, permitindo obter soluções sólidas e adaptadas.

Em todos os cenários, os algoritmos implementados foram capazes de:

* **Cumprir as restrições operacionais** (capacidade, janelas de tempo, autonomia);
* **Gerar rotas eficientes**, com boa relação entre distância percorrida e energia consumida;
* **Executar em tempo aceitável**, mesmo em contextos com múltiplos parâmetros a considerar;
* **Demonstrar flexibilidade e escalabilidade**, servindo como base para aplicação prática futura.

A estrutura modular do sistema facilitou a execução independente dos algoritmos, e os resultados obtidos validam a **utilidade do motor algorítmico enquanto componente central da plataforma A-MoVeR**. Para além disso, a conceção orientada à integração — através da API — assegura que este sistema está pronto para funcionar como **serviço inteligente de apoio à decisão logística em tempo real**.

Com base nos testes e resultados obtidos, conclui-se que:

* A escolha criteriosa do algoritmo é **essencial** para garantir qualidade de resultados e respeito às restrições;
* O sistema é tecnicamente robusto e flexível, com capacidade de expansão e adaptação;
* A plataforma A-MoVeR beneficia de uma **camada de otimização sólida**, capaz de responder a diferentes cenários logísticos com eficiência.

Estes resultados fornecem **uma base validada para futura integração e escalabilidade** do sistema, bem como para a adoção de metodologias mais avançadas, como algoritmos híbridos ou técnicas baseadas em aprendizagem de máquina.

# Discussão

O desenvolvimento deste projeto envolveu não apenas a construção de um sistema funcional, mas também uma **investigação aprofundada sobre os fundamentos teóricos** que sustentam a otimização de rotas, bem como os desafios específicos associados ao uso de motas elétricas em contexto urbano. Esta componente investigativa foi essencial para garantir que as decisões técnicas adotadas fossem fundamentadas, justificadas e alinhadas com o estado da arte da área.

# 6.1. Investigação e Fundamentação Científica

Ao longo do projeto, foi realizada uma **pesquisa estruturada** sobre os seguintes temas:

* **Problemas clássicos de roteamento**, como o TSP (*Travelling Salesman Problem*) e o VRP (*Vehicle Routing Problem*), bem como as suas variantes: CVRP (com capacidade), TWVRP (com janelas temporais), MDVRP (com múltiplos depósitos), DVRP (dinâmico), e EVRP (para veículos elétricos);
* **Características operacionais dos veículos elétricos**, como limites de autonomia, recargas, consumo energético variável e pontos de carregamento estratégicos;
* **Modelos matemáticos** e formulações combinatórias para problemas de roteamento, com e sem restrições;
* **Algoritmos de resolução**, classificados como:
  + *Exatos*, como o **Branch and Bound**;
  + *Heurísticos*, como **Nearest Neighbor**, **Savings** e **Dijkstra**;
  + *Meta-heurísticos*, como **Tabu Search** e **GRASP**.

Cada algoritmo estudado foi analisado não apenas quanto à sua lógica interna, mas também quanto ao seu **comportamento esperado em diferentes contextos**, permitindo associar cada método ao tipo de cenário mais adequado.

Esta componente de investigação foi essencial para fundamentar:

* A escolha dos algoritmos;
* A definição das métricas de avaliação;
* A interpretação dos resultados;
* O desenho do sistema como um todo.

O estudo incluiu fontes académicas, manuais de algoritmos, artigos científicos e benchmarks de sistemas similares, garantindo que o trabalho estivesse alinhado com boas práticas e tendências atuais em logística e mobilidade inteligente.

# 6.2. Escolha e Adequação dos Algoritmos

A principal abordagem adotada neste projeto foi a de **alinhamento contextual**: em vez de aplicar todos os algoritmos a todos os cenários, foi feita uma **seleção informada** da técnica mais adequada a cada situação, com base na investigação realizada e na modelação precisa dos requisitos logísticos.

Esta abordagem permitiu:

* Evitar sobrecarga computacional em algoritmos exatos para cenários inadequados;
* Explorar a força das heurísticas em decisões rápidas;
* Maximizar a capacidade adaptativa das meta-heurísticas em cenários complexos.

# 6.3. Desafios Técnicos

Durante o desenvolvimento, destacaram-se os seguintes desafios:

* **Modelação dos cenários**: transformar narrativas operacionais em dados estruturados exigiu rigor e adaptação;
* **Restrições múltiplas e simultâneas**: como gerir capacidade, janelas de tempo, autonomia e ordem de visita;
* **Conversão de dados entre formatos**: foi necessário criar scripts robustos para extrair, transformar e carregar dados do Excel para JSON;
* **Validação de resultados**: para além do valor da rota, era necessário confirmar o cumprimento total das restrições operacionais;
* **Equilíbrio entre autonomia e eficiência**: sobretudo em motas elétricas, decisões como a ordem dos pontos ou número de viagens podem impactar diretamente o consumo energético.

# 6.4. Contributo para a Plataforma A-MoVeR

O sistema desenvolvido serve como **motor de decisão algorítmica** da plataforma de logística do A-MoVeR. Com a API em desenvolvimento por uma equipa parceira, este sistema passará a funcionar como um **serviço interno**, sendo invocado com dados de entrada e devolvendo rotas otimizadas automaticamente.

Além da componente prática, o projeto demonstra como **conhecimento académico pode ser traduzido em soluções reais**, com impacto direto na eficiência, sustentabilidade e inovação de serviços logísticos urbanos.

# 6.5. Reflexão Final

O projeto permitiu conjugar teoria e prática, desenvolvimento técnico e investigação, colaboração em equipa e aplicação real. A investigação não foi apenas uma etapa inicial, mas **esteve presente ao longo de todo o processo**, orientando decisões, validando resultados e apoiando a escalabilidade da solução.

Mais do que implementar algoritmos, este projeto demonstra a capacidade da Engenharia Informática para **criar ferramentas inteligentes**, integradas e sustentáveis, aplicáveis a desafios concretos da sociedade moderna.

# Conclusão e trabalhos futuros

O presente projeto teve como objetivo o desenvolvimento de um sistema de otimização de rotas aplicado ao uso profissional de motas elétricas no contexto urbano, integrado no ecossistema tecnológico do projeto A-MoVeR. Através da aplicação de algoritmos bem fundamentados, de uma arquitetura modular e da simulação de cenários realistas, foi possível construir uma solução funcional, eficiente e alinhada com os requisitos operacionais da logística elétrica urbana.

# 7.1. Conclusão Geral

Ao longo do trabalho, foram alcançados os seguintes resultados:

* Implementação de um **sistema autónomo** para execução de algoritmos de roteamento, com interface própria e suporte a ficheiros de entrada em formato estruturado;
* Aplicação de **seis algoritmos distintos** em cenários reais, escolhidos de acordo com as restrições e exigências logísticas de cada caso;
* **Validação prática** das soluções geradas, com resultados consistentes e aplicáveis;
* Integração conceptual com os **outros módulos da plataforma A-MoVeR**, nomeadamente o frontend, backend e a API de serviços;
* Produção de um **conjunto de ferramentas reutilizáveis** para simulação, teste e comparação de soluções logísticas com veículos elétricos.

Mais do que um exercício académico, este trabalho demonstrou a capacidade da Engenharia Informática para **responder a problemas reais com soluções inteligentes e tecnicamente sólidas**, contribuindo ativamente para a inovação na área da mobilidade sustentável.

# 7.2. Trabalhos Futuros

Com base na estrutura e resultados obtidos, é possível delinear várias linhas de continuidade para evolução do sistema:

* **Integração total com a API REST**, permitindo o uso em produção dentro da plataforma A-MoVeR;
* **Substituição dos ficheiros JSON por base de dados dinâmica**, permitindo persistência, rastreamento de histórico e análise em tempo real;
* **Criação de um módulo de visualização gráfica das rotas**, integrado com mapas reais (ex: OpenStreetMap, Google Maps);
* **Otimização energética avançada**, incorporando variáveis como topografia, velocidade média e carga da mota;
* **Adição de algoritmos baseados em inteligência artificial**, como reforço ou aprendizagem supervisionada;
* **Simulação em ambiente real**, com testes em motas elétricas reais e recolha de métricas de operação no terreno;
* **Interface web para simulação interativa**, tornando o sistema acessível a operadores e gestores logísticos sem conhecimentos técnicos.

Estes trabalhos futuros não só reforçarão a utilidade prática do sistema, como também permitirão explorar novas áreas de investigação científica e tecnológica no domínio da logística urbana e da mobilidade inteligente.

# Bibliografia

**[1]** Dijkstra, E. W. (1959). *A note on two problems in connexion with graphs*. *Numerische Mathematik*, 1(1), 269–271.  
Referência original do algoritmo de Dijkstra, utilizado para cálculo do caminho mais curto.

**[2]** Clarke, G., & Wright, J. W. (1964). *Scheduling of vehicles from a central depot to a number of delivery points*. *Journal of the Operational Research Society*, 15(4), 568–581.  
Algoritmo Savings clássico, aplicado em cenários com múltiplos depósitos.

**[3]** Glover, F., & Laguna, M. (1997). *Tabu Search*. Springer.  
Referência fundamental para a meta-heurística Tabu Search.

**[4]** Feo, T. A., & Resende, M. G. C. (1995). *Greedy Randomized Adaptive Search Procedures*. *Journal of Global Optimization*, 6(2), 109–133.  
Base teórica do algoritmo GRASP.

**[5]** Toth, P., & Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM.  
Obra seminal sobre o VRP e suas variantes.

**[6]** Rego, C. (1991). *A tabu search algorithm with ejection chains for the capacitated vehicle routing problem*. *European Journal of Operational Research*, 50(3), 420–428.  
Extensão da Tabu Search para VRP com capacidade.

**[7]** Solomon, M. M., & Golden, B. L. (1991). *A computational study of vehicle routing with time windows*. *Transportation Science*, 25(1), 22–35.  
Fonte para modelagem de problemas VRP com janelas temporais.

**[8]** Muñoz, J. C., & Salinas, M. E. (2008). *Vehicle routing in urban logistics: State of the art and future trends*. *European Journal of Operational Research*, 188(1), 1–17.

**[9]** Wang, Q., & Wang, B. (2011). *Route optimization for electric vehicles using metaheuristics*. *Transportation Research Part C: Emerging Technologies*, 19(6), 173–183.

**[10]** Cunha, J. (2000). *Modelos e Técnicas para a Otimização de Rotas*. Dissertação de Mestrado, IST.  
Referência nacional para fundamentos de otimização de rotas e aplicação prática.

**[11]** Python Software Foundation. (2023). *Python Language Reference, version 3.8*. Disponível em: <https://www.python.org/>