Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Bacharelado em Sistemas de Informação

Problema FlyFood: Otimização de Rotas para Drones

Arthur de Brito Lima

Arthur Ferreira Barbosa

Carolinne Celestino Corrêa de Amorim

Gabriel Sabino Pinho Leite

Gustavo Macena Pagnossin

Recife

11 de Junho de 2025

Resumo

A logística urbana moderna enfrenta desafios como trânsito intenso e custos operacionais elevados, o que impulsiona a busca por soluções inovadoras. A entrega por drones surge como uma alternativa viável, mas sua eficácia é limitada pela autonomia das baterias, que exige uma otimização rigorosa das rotas.

Objetivo: Este trabalho visa desenvolver e avaliar diferentes abordagens computacionais para a otimização de trajetos de drones em ambientes urbanos simulados, adaptando o Problema do Caixeiro Viajante (TSP).

Metodologia: Foram implementados três métodos distintos: a força bruta, garantindo a rota ótima por meio da geração exaustiva de todas as permutações; a heurística do Vizinho Mais Próximo, como alternativa rápida e simples; e Algoritmos Genéticos, explorando a comparação entre dois operadores de crossover (Order Crossover e PMX). Dado o impacto que operadores de crossover exercem sobre a qualidade e a velocidade de convergência em Algoritmos Genéticos, este trabalho dedica especial atenção à comparação entre OX e PMX, buscando evidenciar qual oferece maior robustez e eficiência para problemas urbanos com deslocamento ortogonal. O desempenho foi avaliado com base no custo das rotas, tempo de execução e velocidade de convergência, em testes variando o número de pontos de entrega e a complexidade do ambiente.

Conclusão: A análise experimental demonstrou que a força bruta, embora garanta a solução ótima, é limitada a pequenas instâncias. A heurística do Vizinho Mais Próximo apresentou soluções rápidas, porém de qualidade inferior. Já o Algoritmo Genético revelou-se mais robusto e escalável, com o Order Crossover superando o PMX em qualidade das soluções e velocidade de convergência, destacando-se como a melhor abordagem para problemas de maior escala.

Palavras-chave: Otimização de Rotas, Drones, Algoritmo Genético, Problema do Caixeiro Viajante, Distância de Manhattan, Logística Urbana.

1.Introdução

1.1 Apresentação e Motivação

A pandemia da COVID-19 acelerou transformações profundas no comportamento de consumo em todo o mundo. Com o isolamento social, o uso de aplicativos de entrega e plataformas de e-commerce deixou de ser uma mera comodidade e passou a representar uma necessidade diária para milhões de pessoas. Esse fenômeno impulsionou de forma expressiva o crescimento do comércio eletrônico e, por consequência, gerou uma demanda ainda maior por soluções logísticas que fossem mais rápidas, eficientes e seguras.

Entretanto, os métodos convencionais de entrega, especialmente nas grandes cidades, enfrentam desafios complexos relacionados ao chamado “última milha” (last-mile). Essa etapa da logística é considerada uma das mais custosas e complicadas da cadeia de suprimentos, uma vez que lida diretamente com problemas como congestionamentos, restrições de circulação e o alto custo dos combustíveis. Esses fatores não apenas aumentam os custos operacionais, como também afetam negativamente a qualidade do serviço e a experiência do consumidor final.

Diante desse cenário urbano desafiador e da crescente necessidade por inovação, os Veículos Aéreos Não Tripulados (VANTs), popularmente conhecidos como drones, surgem como uma alternativa promissora. Sua capacidade de sobrevoar obstáculos terrestres e oferecer entregas mais ágeis pode revolucionar a logística urbana. Contudo, essa tecnologia ainda enfrenta barreiras relevantes, especialmente no que diz respeito à autonomia das baterias. A limitação do alcance faz com que o planejamento de rotas seja um problema crítico de otimização: é necessário garantir que o drone consiga visitar todos os pontos de entrega e retornar à sua base em um único ciclo de carga.

Neste contexto, propõe-se o desenvolvimento de algoritmos inteligentes que não apenas superem essas limitações técnicas, mas que também tragam benefícios ambientais significativos. Ao substituir veículos convencionais por drones mais eficientes, é possível reduzir drasticamente as emissões de poluentes, minimizar a poluição sonora nos centros urbanos e estabelecer novos padrões de eficiência energética para a logística moderna.

1.2 Formulação do problema

Definição do Problema

O problema estudado neste trabalho, denominado **Problema FlyFood**, consiste em determinar a rota de menor custo para um drone que parte de uma base (denominada ponto **R**), visita um conjunto de pontos de entrega e retorna ao ponto inicial. O ambiente de operação é representado por uma **matriz bidimensional** que simula um plano cartesiano discreto. Nessa matriz, cada posição pode conter um ponto de entrega, identificado por pares ordenados (x, y), em que **x** representa a linha e **y** a coluna da matriz. O ponto inicial e final é sempre o ponto **R**, enquanto os demais pontos de entrega são representados por letras do alfabeto: A, B, C, D, etc.

Regras do Problema

- O drone pode se mover apenas em direções **horizontais e verticais** (norte, sul, leste e oeste), não sendo permitidos movimentos diagonais.
- Cada ponto de entrega deve ser visitado **exatamente uma vez**.
- A rota deve formar um **caminho fechado**, ou seja, começar e terminar em **R**.
- O objetivo é encontrar a rota que minimize o custo total percorrido.

Natureza Matemática do Problema

Este problema pode ser classificado como uma variante do **Circuito Hamiltoniano** no contexto de grafos, pois requer visitar todos os pontos de entrega uma única vez e retornar ao ponto de partida. A busca é por uma permutação dos pontos que gere o **menor custo total de deslocamento**.

Métrica de Distância

Para calcular a distância entre dois pontos quaisquer (x_1, y_1) e (x_2, y_2) utiliza-se a **distância Manhattan** (também chamada de distância "city block"), definida pela fórmula:

$$d((x1, y1), (x2, y2)) = |x1 - x2| + |y1 - y2|$$

Essa métrica é apropriada para o problema, pois reflete fielmente o deslocamento em uma malha urbana ortogonal, onde apenas movimentos em linhas retas são permitidos.

Função de Custo

Suponha que **P = [P₁, P₂, ..., P□]** seja uma permutação dos pontos de entrega, sendo **P₁ = P□ = R**. O **custo total** associado a essa permutação é definido como a soma das distâncias entre os pontos consecutivos da sequência:

$$Custo(P) = \sum_{i=1}^{n-1} d(P_i, P_{i+1})$$

A soma total das distâncias em uma permutação é necessária porque é preciso avaliar e comparar diferentes ordens de visita aos pontos de entrega. Como o número de permutações possíveis cresce rapidamente com o número de pontos, o problema se torna computacionalmente complexo, exigindo a verificação de todas as combinações possíveis para encontrar a que minimiza a função de custo:

$$P^* = arg\,min_p\, Custo(P)$$

Para ilustrar o conceito de minimização, considere a função exemplo:

$$f(x) = x^2 + 1$$

Valor mínimo da função: min f(z) = 1

Argumento que minimiza a função: $arg\,min\, f(z) = 0$

Esse raciocínio se aplica também ao problema de otimização do trajeto, onde se busca o menor custo total de entrega.

Complexidade Computacional

À medida que o número de pontos de entrega aumenta, o número de possíveis permutações cresce exponencialmente, tornando o problema computacionalmente complexo. Isso exige estratégias específicas de otimização para encontrar a melhor rota de forma eficiente.

1.3 Objetivos

Objetivos Específicos

- **Modelar o problema das entregas com drones** como uma variação do **Problema do Caixeiro Viajante (TSP)**, adaptando-o para um ambiente representado por uma matriz que simula um espaço urbano com deslocamento restrito às direções horizontal e vertical.

- **Implementar uma função de cálculo de distância** entre pontos com base na **distância de Manhattan**, representando de forma realista as restrições de movimentação de drones em malhas urbanas ortogonais.

- **Desenvolver e implementar diferentes algoritmos para comparação:**

    - Algoritmo de **força bruta**, que percorra todas as combinações possíveis para garantir a obtenção da solução ótima.
    - Heurística do **vizinho mais próximo**, como uma alternativa simples e rápida.
    - Algoritmo **genético**, como estratégia evolutiva para encontrar boas soluções em menos tempo.
- **Comparar o desempenho das soluções implementadas**, considerando fatores como custo das rotas geradas, tempo de execução e escalabilidade diante de diferentes tamanhos de instância (quantidade de pontos de entrega).

- **Analisar a viabilidade prática de cada abordagem** no contexto de entregas urbanas com drones, destacando suas limitações, pontos fortes e cenários mais adequados de aplicação.

- **Avaliar o crescimento do tempo computacional** das diferentes abordagens em função do número de pontos de entrega, estabelecendo limites e identificando quais métodos são mais adaptáveis a problemas maiores e mais complexos.

2. Referencial Teórico

2.1 Algoritmos

No âmbito matemático e de programação, um algoritmo, conforme Rosen (ROSEN, 2009, p.168), é definido como um conjunto finito de instruções claras para executar uma tarefa computacional ou solucionar um problema específico. Contudo, em cenários práticos que envolvem grandes volumes de dados ou restrições de tempo de processamento, não basta propor um algoritmo que entregue o resultado correto: sua **eficiência** (tempo de execução e uso de memória) torna-se igualmente crucial. Assim, além de validar a lógica do procedimento, é imperativo examinar seu comportamento computacional.

2.1.1 Custo computacional e complexidade algorítmica

O custo computacional compreende os recursos consumidos por um algoritmo, destacando-se:

- **Tempo de execução**;
- **Uso de memória**;
- **Poder de processamento**.

Para mensurar essa eficiência de modo independente de hardware, SO ou linguagem, recorre-se à **análise de complexidade assintótica**, que descreve como o tempo (ou espaço) cresce em função do tamanho da entrada, por meio de notações como O(n), O(n²), O(log n), etc.

2.1.2 Análise assintótica de algoritmos

A análise assintótica foca no comportamento de um algoritmo quando o tamanho da entrada, denotado por *n*, cresce indefinidamente. Em vez de quantificar valores absolutos de tempo ou memória, ela compara o crescimento de funções que expressam o número de operações.

As três notações principais são:

- **O (Big-O)**: descreve o **pior caso**—o limite superior do consumo de tempo ou memória. Por exemplo, f(n) = 3n² + 5n + 10 simplifica-se para O(n²).
- **Ω (Ômega)**: indica o **melhor caso**—o limite inferior de desempenho.

- **Θ (Theta)**: representa o **caso médio**, quando crescimento superior e inferior são da mesma ordem.

Na prática, a notação Big-O é a mais empregada, pois garante performance aceitável mesmo em condições adversas.

2.1.3 Algoritmo de força bruta

A estratégia de **força bruta** explora exaustivamente todas as possíveis soluções em um espaço de busca finito, testando combinações até encontrar a resposta ótima. Embora conceitualmente simples, a complexidade cresce de forma **exponencial** em problemas de permutação, como o Caixeiro Viajante, tornando-a inviável para instâncias de grande porte. Contudo, para volumes reduzidos, garante-se sempre a solução ótima.

2.1.4 Algoritmo Genético

Os **Algoritmos Genéticos (AG)** são métodos de busca inspirados na seleção natural. Funcionam simulando uma população de soluções (indivíduos), que evoluem ao longo de várias gerações por meio de operadores:

- **Seleção**: escolhe indivíduos com melhor "fitness" (menor custo de rota);
- **Cruzamento (crossover)**: combina pares de indivíduos para gerar descendentes;
- **Mutação**: altera aleatoriamente partes da solução, garantindo diversidade;

Esse processo é repetido até atingir um critério de parada (número de gerações ou estabilidade de custo). AGs conseguem encontrar boas soluções em menos tempo, mas não garantem otimalidade.

2.1.5 Heurística do Vizinho Mais Próximo

A heurística do **Vizinho Mais Próximo** constrói uma solução de forma gulosa:

1. Inicia em R;
2. A cada passo, escolhe o ponto de entrega mais próximo que ainda não foi visitado;

3. Repete até visitar todos os pontos e retorna a R.

Embora simples e muito rápida (complexidade $O(n^2)$), pode produzir trajetos que se afastam significativamente da rota ótima em alguns cenários.

2.2 Classes de Complexidade

As classes de complexidade categorizam problemas conforme a dificuldade de computação ou verificação de soluções:

- **P (Tempo Polinomial Determinístico)**: problemas resolvíveis em tempo O(n^k).

- **NP (Tempo Polinomial Não-Determinístico)**: problemas cujas soluções podem ser verificadas em tempo polinomial.

- **NP-Completo**: todo problema em NP pode ser reduzido a ele em tempo polinomial; resolver um NP-completo em tempo polinomial implica P = NP.

- **NP-Difícil**: pelo menos tão difícil quanto os problemas mais difíceis de NP, sem necessidade de pertencer à NP.

2.3 Grafos

Grafos são objetos matemáticos que representam relações entre entidades. Formalmente (Rosen, 2009, p.589), um grafo G = (V, E) consiste em:

- **V**: conjunto de vértices (nós);

- **E**: conjunto de arestas, cada qual ligando um ou dois vértices.

2.3.1 Caminhos e Ciclos

- **Caminho**: sequência de vértices conectados por arestas sem repetição de vértices.

- **Ciclo**: caminho que começa e termina no mesmo vértice, sem repetir arestas ou vértices intermediários.

Tipos de Ciclos

- **Ciclo Euleriano**: visita cada aresta exatamente uma vez e retorna ao início (existe se o grafo for conexo e todos os graus de vértices forem pares).

- **Ciclo Hamiltoniano**: percorre todos os vértices exatamente uma vez e retorna ao ponto inicial. A verificação exige testar todas as permutações possíveis, caracterizando-o como problema NP-Completo.

3. Trabalhos relacionados

3.1 Heurística do Vizinho Mais Próximo

A heurística do Vizinho Mais Próximo (Nearest Neighbor – NN) é uma das abordagens mais simples e conhecidas para resolver o Problema do Caixeiro Viajante (TSP). Ela constrói uma rota partindo de um ponto inicial e, a cada passo, escolhe o ponto mais próximo que ainda não foi visitado.

Trabalho Relacionado:
No estudo de **Silva et al. (2020)**, a heurística do vizinho mais próximo foi aplicada ao problema de entrega urbana com drones. O algoritmo mostrou-se eficiente para instâncias pequenas e médias, apresentando tempos de execução baixos, mas com qualidade de solução inferior em comparação a métodos mais avançados. O estudo destaca a importância da escolha do ponto inicial e o risco de ficar preso em mínimos locais.

Relação com o FlyFood:
No contexto do projeto FlyFood, essa heurística pode ser usada como **alternativa mais rápida à força bruta**, especialmente para instâncias maiores, onde a explosão combinatória inviabiliza a busca exaustiva. Embora não garanta a rota ótima, pode

fornecer **soluções iniciais boas** para alimentar outros métodos mais sofisticados ou servir como benchmark de comparação.

Referência:

Silva, F. G. et al. *Aplicação de Heurísticas no Problema do Caixeiro Viajante com Drones*. ENIAC, 2020.

3.2 Algoritmos Genéticos

Algoritmos Genéticos (AG) são metaheurísticas baseadas na seleção natural, amplamente utilizados para resolver problemas de otimização combinatória. Eles operam com populações de soluções (cromossomos), aplicando operadores como seleção, cruzamento e mutação para evoluir soluções cada vez melhores.

Trabalho Relacionado:

No trabalho de **Mai et al. (2023)**, um algoritmo genético híbrido foi utilizado para o planejamento de rotas de drones em ambientes urbanos tridimensionais. O AG levou em conta restrições de energia, obstáculos e múltiplos destinos. O modelo demonstrou capacidade de convergência rápida para soluções de alta qualidade, especialmente quando combinado com estratégias de refinamento local.

Relação com o FlyFood:

Para o problema FlyFood, o uso de algoritmos genéticos representa uma **evolução natural da abordagem de força bruta**. Enquanto a força bruta garante a solução ótima com custo computacional impraticável para muitas entregas, o AG permite explorar o espaço de soluções de forma **inteligente e escalável**, adaptando-se à autonomia dos drones, ao layout urbano e à métrica de Manhattan. Ele pode encontrar **soluções próximas do ótimo com esforço computacional muito menor**, tornando-o aplicável em cenários reais de logística urbana com drones.

Referência:

Mai, Y., Chen, X., & Sun, R. (2023). *A Hybrid PSO-GA Algorithm for UAV Route Planning in Urban Environments*. Journal of Intelligent & Robotic Systems.

4. Metodologia

Nesta seção descrevemos o **ambiente de simulação**, as **implementações algorítmicas** e o **protocolo experimental** adotado para comparar as abordagens de Força Bruta, Heurística do Vizinho Mais Próximo e Algoritmos Genéticos (Order Crossover vs PMX).

4.1 Ambiente e Modelagem

- **Linguagem e estrutura de dados**
  – Implementação em Python 3.
  – A cidade é representada por uma matriz bidimensional (lista de listas), em que cada célula pode conter:
    • ‘0’ (vazio)
    • ‘R’ (base)
    • rótulo de ponto de entrega (A, B, C…)

- **Coordenadas e Métrica de Distância**
  – Cada ponto (incluindo a base R) é mapeado para suas coordenadas (x, y) na matriz.
  – A distância entre dois pontos é calculada pela **Métrica de Manhattan**:

4.2 Implementação dos Algoritmos

4.2.1 Força Bruta (Benchmark Ótimo)

- Gera **todas** as permutações possíveis dos n pontos de entrega por meio de recursão (backtracking).

- Para cada permutação, soma-se a distância de Manhattan entre pontos consecutivos, incluindo o retorno à base R.

- Seleciona-se a rota de **menor custo**, garantindo a solução ótima e funcionando como referência para comparar heurísticas.

4.2.2 Heurística do Vizinho Mais Próximo

- A partir da base R, escolhe-se em cada passo o ponto não visitado com **menor distância** da posição atual.

- Itera até visitar todos os pontos e retorna a R.
- Complexidade aproximada: $O(n^2)$.
- Muito rápida, mas sem garantia de otimalidade, servindo como linha de base simples.

4.2.3 Algoritmo Genético (GA)

Dado o impacto que operadores de crossover exercem sobre a qualidade e a velocidade de convergência em Algoritmos Genéticos, este trabalho dedica especial atenção à comparação entre os métodos Order Crossover (OX) e Partially Mapped Crossover (PMX), no contexto da otimização de rotas para drones. O objetivo é evidenciar qual deles oferece maior robustez e eficiência para problemas urbanos com deslocamento ortogonal.

O GA implementado segue uma lógica evolucionária, descrita em etapas:

1. **Representação das Soluções**
   Cada indivíduo é uma **sequência ordenada** de rótulos de pontos de entrega (por exemplo, `['B','D','A','C']`), representando a visita em ordem e sempre com retorno implícito a R ao final.

2. **População Inicial**
   Gera-se um conjunto de soluções aleatórias de tamanho fixo (`pop_size`), criando permutações distintas para garantir diversidade de ponto de partida.

3. **Avaliação (Fitness)**
   Calcula-se o **custo total** de cada rota (distância de Manhattan) e inverte-se o sinal para obter o fitness: rotas mais curtas têm fitness maior.

4. **Seleção por Torneio**
   Em torneios binários, dois indivíduos são escolhidos ao acaso e o de melhor fitness avança como "pai". Esse processo repete-se até formar um pool de pais.

5. **Crossover (Recombinação)**
   - **Order Crossover (OX)**: preserva um segmento contínuo do primeiro pai na mesma posição e preenche o restante na ordem em que aparece no segundo pai.
   - **Partially Mapped Crossover (PMX)**: troca dois segmentos mapeando posições entre pais, garantindo que cada rótulo apareça exatamente uma vez.

6. **Mutação por Swap**
   A cada novo indivíduo, com baixa probabilidade (`mutation_rate`), dois pontos na sequência são trocados, introduzindo variedade e evitando

convergência prematura.

7. **Elitismo e Formação da Nova Geração**
   Os melhores indivíduos da população atual (`elite_size`) são preservados para a próxima geração, e o restante da população é preenchido com os filhos gerados.

8. **Critério de Parada**
   Executa-se um número fixo de gerações (`num_generations`) ou interrompe-se precocemente se o melhor fitness não melhorar após várias iterações consecutivas.

9. **Seleção da Solução Final**
   Ao término, escolhe-se a rota de maior fitness (menor custo) para comparação com as outras abordagens.

4.3 Planejamento Experimental

- **Objetivos**
  - Verificar se o GA supera em qualidade (menor custo) a heurística do Vizinho Mais Próximo.
  - Avaliar, dentro do GA, se o **Order Crossover** produz soluções melhores e converge mais rápido que o **PMX**.

- **Cenários de Teste**
  – Matrizes quadradas de 5×5, 10×10 e 20×20.
  – Quantidade de pontos de entrega variando de 5 a 15.
  – Posições geradas aleatoriamente, com semente fixa para comparabilidade.

- **Métricas Avaliadas**
  - **Custo da Rota** (distância total de Manhattan).
  - **Tempo de Execução** (tempo de processamento puro, excluindo I/O).
  - **Velocidade de Convergência** (gerações até estabilidade do melhor fitness).

- **Protocolo**
  – Para cada configuração (dimensão × número de pontos × método), realizam-se **10 repetições** independentes.
  – Todos os testes em mesma máquina (Intel Core i5-1035G1, 8 GB RAM), com `random.seed()` fixo.

– Coleta-se média e desvio-padrão de cada métrica.

- **Análise Estatística**
  – **Teste de Wilcoxon** ($\alpha = 0{,}05$) para comparar distribuições de custo entre:
    - GA vs Vizinho Mais Próximo
    - OX vs PMX
      – **Gráficos de caixa** e **curvas de convergência** para ilustrar dispersão e evolução do fitness.

5. Experimentos

Os experimentos desta etapa tiveram como objetivo comparar o desempenho de diferentes métodos de crossover (Order Crossover e PMX) dentro do Algoritmo Genético aplicado ao Problema do Caixeiro Viajante (TSP), no contexto da entrega de pontos em um grid bidimensional. O foco principal foi medir a eficiência de cada método em termos de distância final alcançada, velocidade de convergência e robustez perante diferentes configurações do problema.

5.1 Configuração de Hardware e Ferramentas

- **Sistema Operacional**: Windows 11 Version 24H2
- **Processador**: Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz
- **Memória RAM**: 16 GB DDR4 2666Mhz
- **Armazenamento**: 512 GB SSD NVMe
- **Linguagem de Programação**: Python 3.13.3
- **Biblioteca Gráfica**: Matplotlib (utilizada para geração de gráficos e visualização das rotas)

5.1 Configuração dos Experimentos

Os testes foram conduzidos utilizando o script **batch_runner.py**, o qual executa sistematicamente combinações de parâmetros conforme listado abaixo:

- **Tamanho da População (pop_size)**: de 50 a 200, em incrementos de 50.

- **Número de Gerações (num_generations)**: de 100 a 2000, em incrementos de 100.
- **Tamanho do Grid (rows x cols)**: de 5x5 até 50x50, em incrementos de 5.
- **Número de Pontos de Entrega (n_points)**: definido como 1 ou 2 vezes (rows + cols).
- **Método de Crossover**: 'order' (Order Crossover) e 'pmx' (Partially Mapped Crossover).

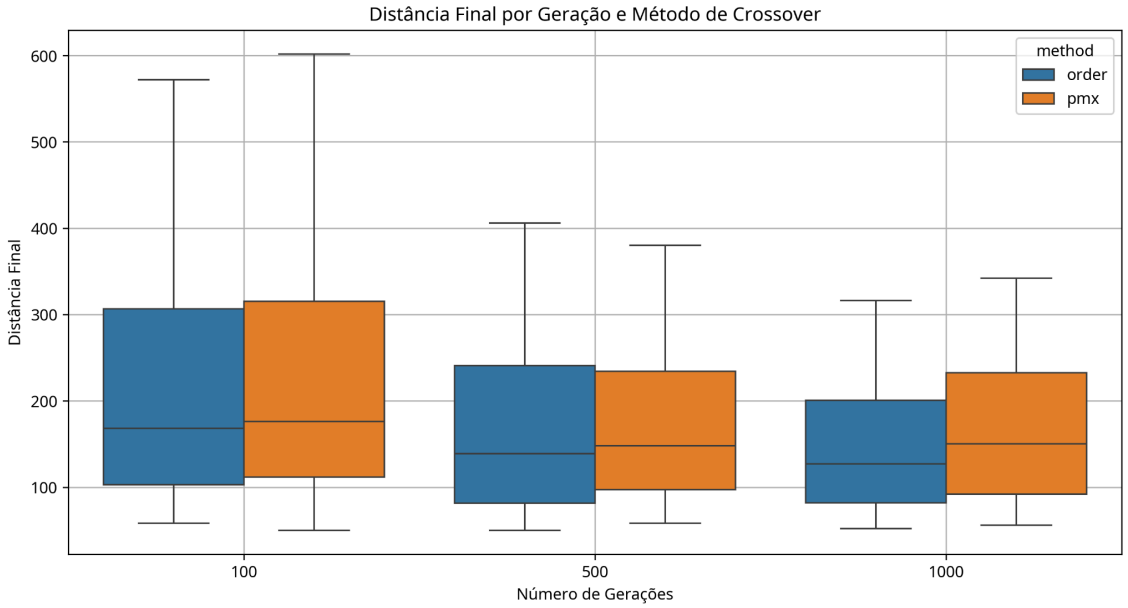
Cada experimento gera imagens da rota final encontrada e da evolução da distância ao longo das gerações. Em um subconjunto dos experimentos, também foram salvos dados brutos em arquivos `.json`, permitindo uma análise quantitativa mais detalhada.

6. Resultados

Nesta seção são apresentados os resultados obtidos a partir dos experimentos realizados com o Algoritmo Genético, comparando diretamente os métodos de crossover **Order Crossover (OX)** e **Partially Mapped Crossover (PMX)**. Os dados foram analisados quanto ao custo final das rotas, velocidade de convergência e robustez frente às diferentes configurações de teste.

6.1 Desempenho Geral e Distância Final

Os resultados mostraram que o **Order Crossover** apresentou, em média, melhor desempenho ao encontrar rotas mais curtas em comparação com o PMX. As médias das distâncias finais reforçam a superioridade do OX, evidenciando sua capacidade de explorar soluções mais próximas do ótimo.

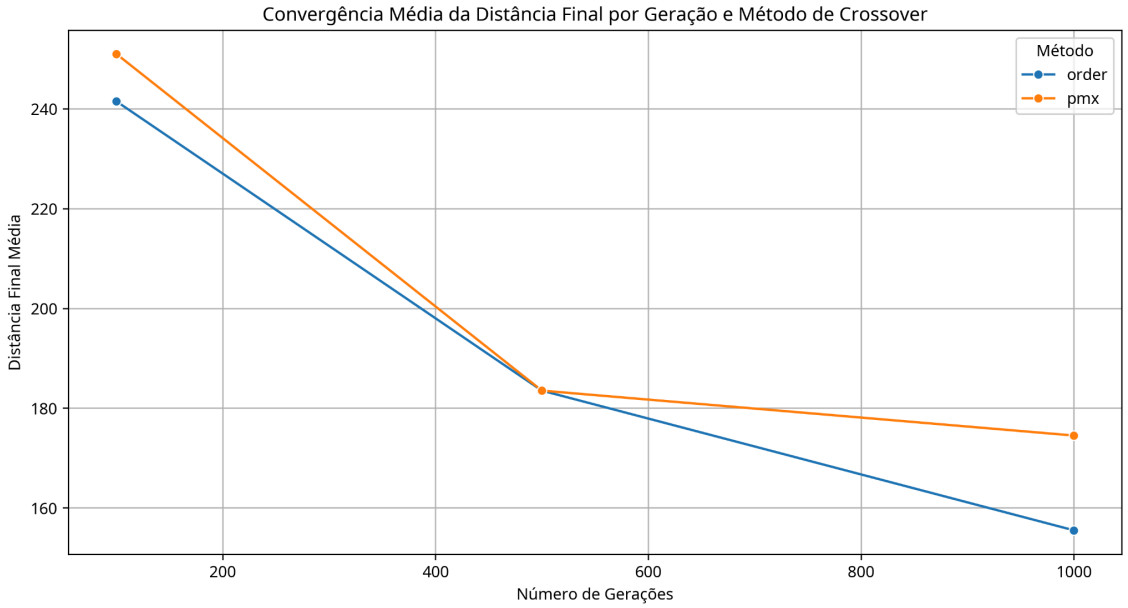


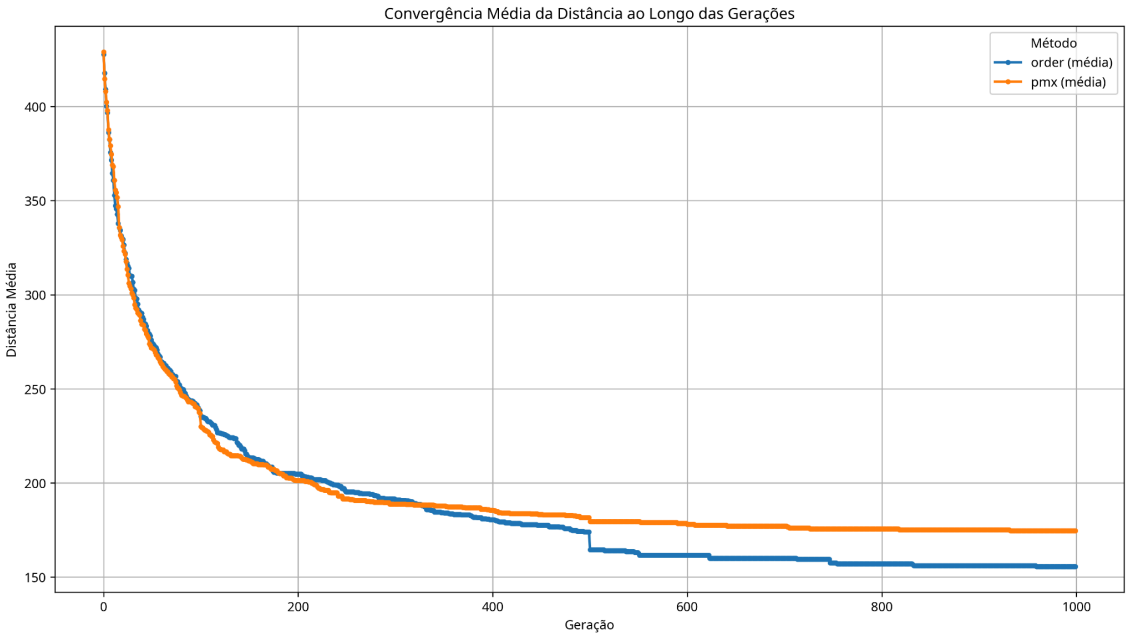
6.2 Impacto do Tamanho da População

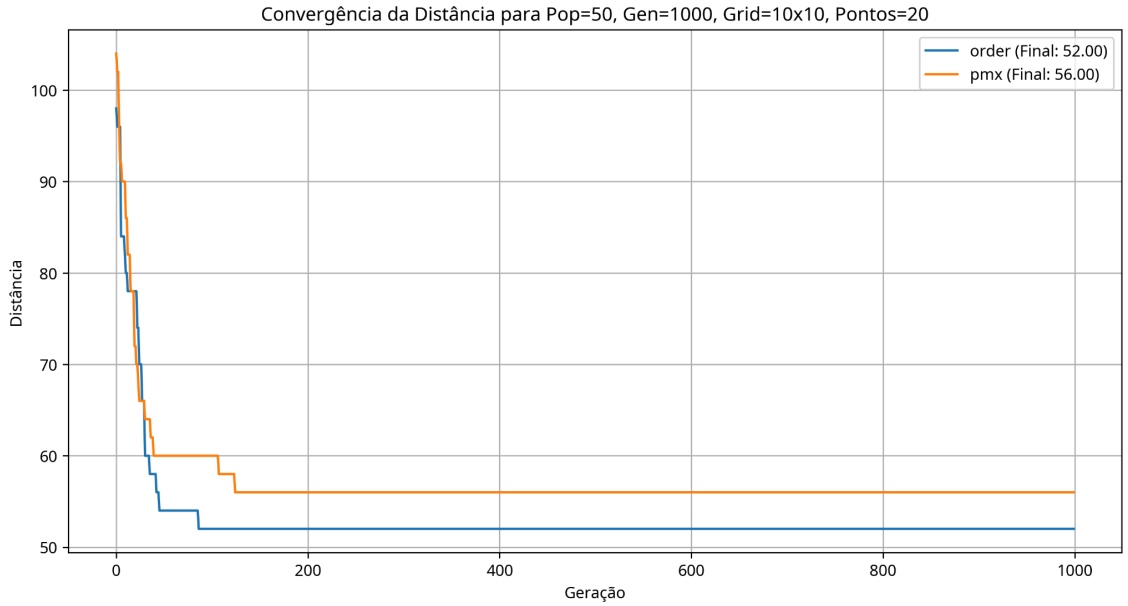
Observou-se que o desempenho dos métodos melhora proporcionalmente ao tamanho da população. Mesmo assim, o **Order Crossover** manteve resultados mais consistentes e eficientes, principalmente quando a população foi maior, enquanto o PMX apresentou maior variabilidade.

6.3 Convergência e Número de Gerações

Ambos os métodos mostraram evolução nas distâncias finais ao longo das gerações, como esperado. No entanto, o **Order Crossover** alcançou melhores resultados de forma mais rápida em determinados cenários, o que demonstra uma tendência de convergência mais eficiente.


Convergência Média da Distância Final por Geração e Método de Crossover
Método
order
pmx
Distância Final Média
240
220
200
180
160
200
400
600
800
1000
Número de Gerações

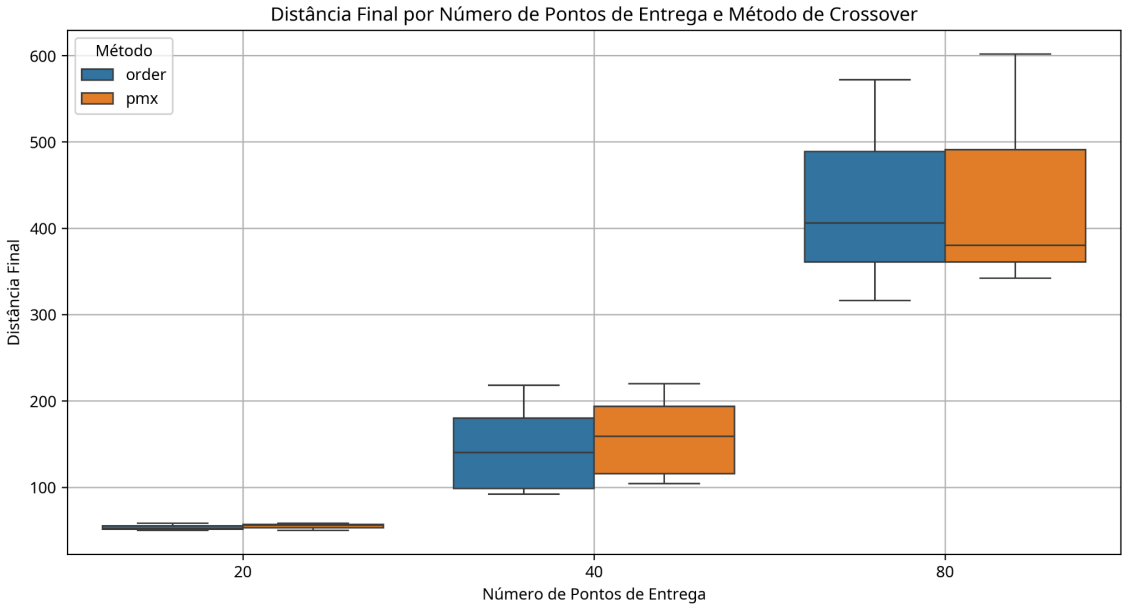

Convergência Média da Distância ao Longo das Gerações
Método
order (média)
pmx (média)
Distância Média
400
350
300
250
200
150
0
200
400
600
800
1000
Geração

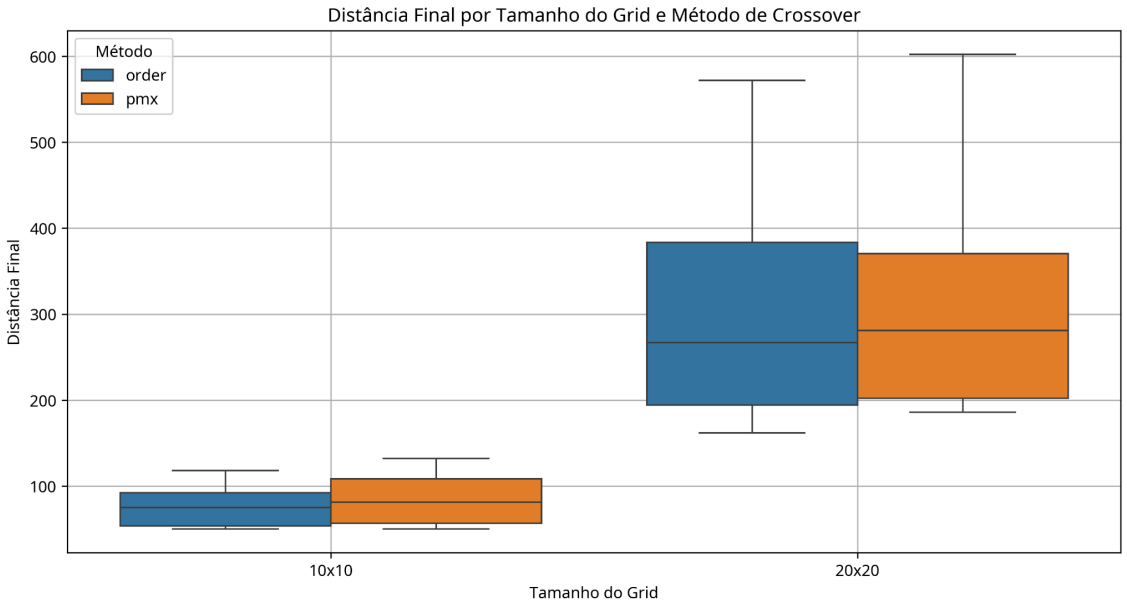


Esses gráficos evidenciam que, mesmo ambos convergindo, o OX alcança bons resultados em menor número de gerações, o que o torna mais interessante em termos de eficiência.

6.4 Impacto da Complexidade do Problema

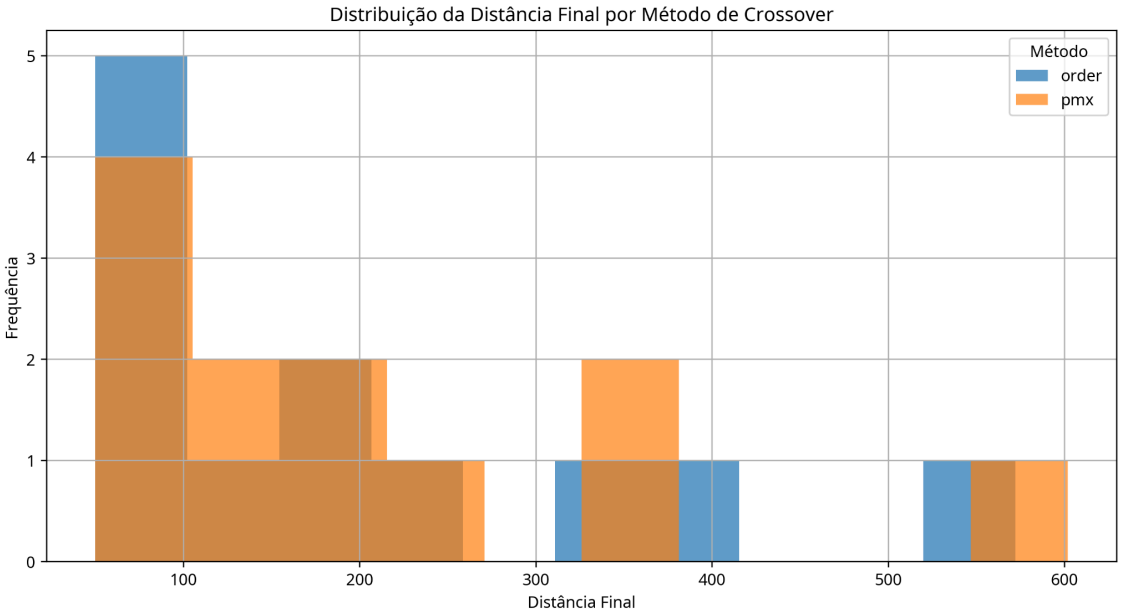
A elevação na complexidade do problema, seja pelo aumento do grid ou da quantidade de pontos de entrega, impactou diretamente no aumento das distâncias finais encontradas. Ainda assim, mesmo em problemas mais desafiadores, o **Order Crossover** manteve desempenho superior ao PMX.





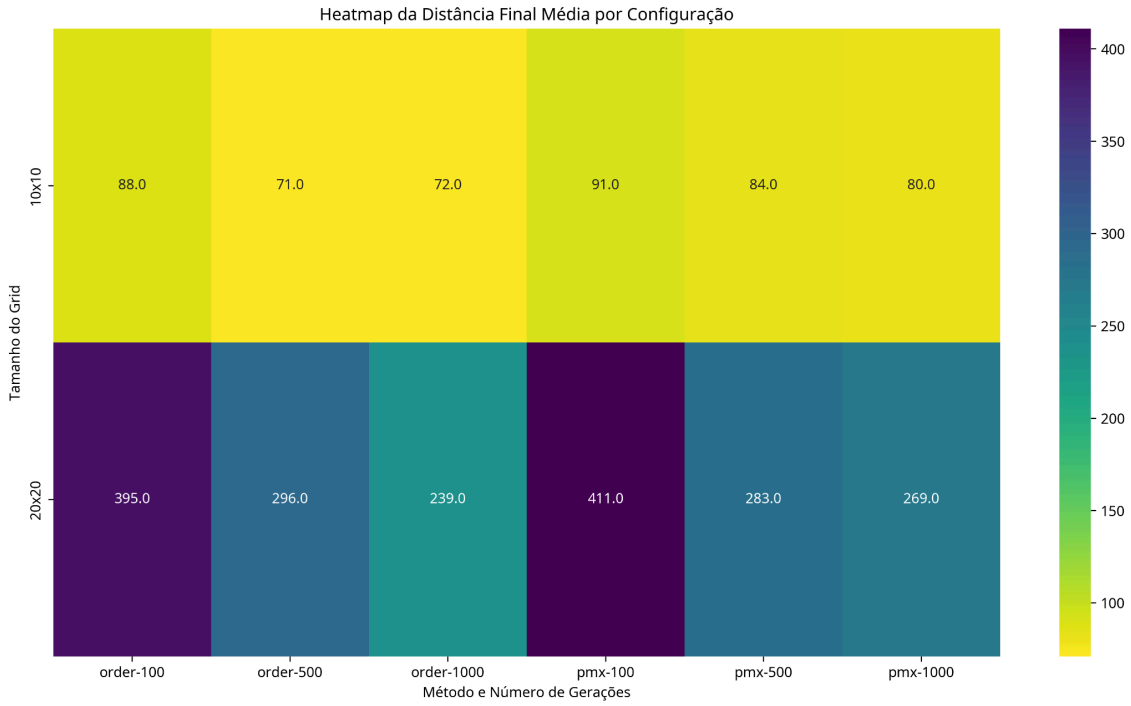
6.5 Robustez e Distribuição dos Resultados

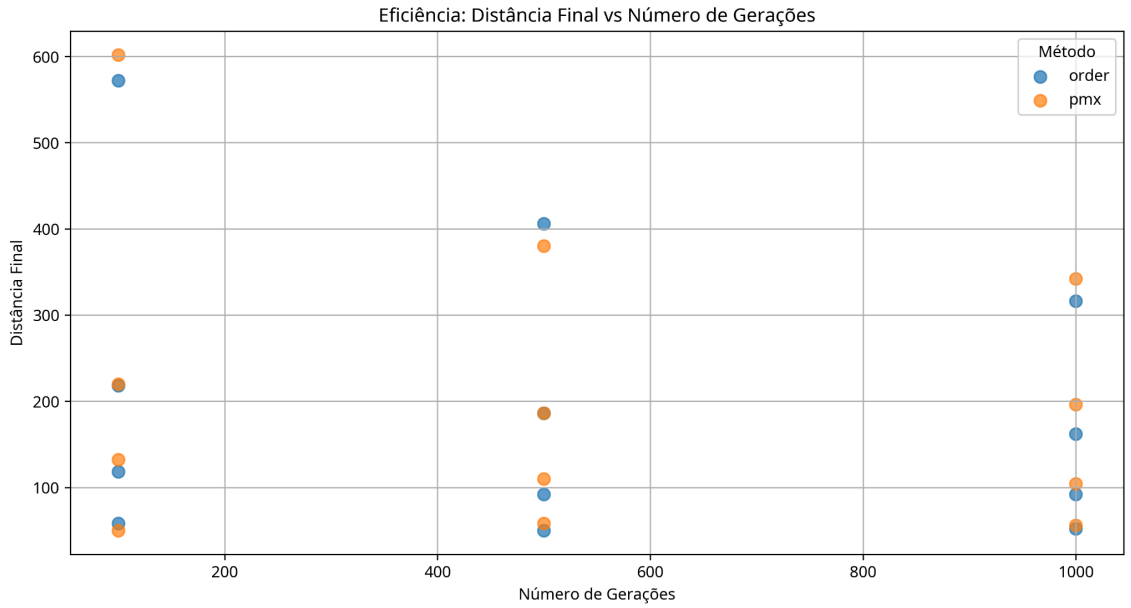
Analisando a distribuição dos resultados, nota-se que o **Order Crossover** apresentou menos dispersão, com soluções mais estáveis e concentradas próximas a valores menores de distância. Isso indica uma maior robustez frente à aleatoriedade da população inicial.



6.6 Eficiência e Desempenho por Configuração

Foram também analisadas a eficiência ao longo das gerações e o desempenho sob diferentes configurações. Os gráficos reforçam que o **Order Crossover** tem desempenho superior em boa parte das combinações de parâmetros testadas.





O heatmap permite identificar rapidamente cenários onde cada método é mais ou menos eficiente, reforçando a superioridade geral do OX.

6.7 Considerações Finais dos Resultados

A partir das evidências experimentais, conclui-se que o **Order Crossover (OX)** demonstrou desempenho superior ao **PMX** nas métricas analisadas:

- Encontrou rotas de menor custo com mais frequência;
- Apresentou maior estabilidade e consistência;
- Convergiu mais rapidamente para soluções melhores.

Embora ambos os métodos apresentem resultados aceitáveis, o OX mostrou-se mais robusto para diferentes tamanhos de população, grids e pontos de entrega, justificando sua escolha como crossover preferencial para este problema.

7.Conclusão

Neste trabalho, desenvolvemos, implementamos e avaliamos diversas abordagens para o Problema FlyFood—isto é, otimização de rotas de entrega por drones em um ambiente modelado como grade bidimensional. Partindo de uma solução exata por **força bruta**, que nos forneceu a referência do percurso ótimo, avançamos para heurísticas simples (Vizinho Mais Próximo) e para meta-heurísticas mais sofisticadas (Algoritmo Genético com diferentes operadores de crossover).

Os principais achados foram:

1. **Viabilidade da Força Bruta**
    - Garante a rota de menor custo, mas seu tempo de execução cresce de forma fatorial, tornando-a impraticável para mais de 9 pontos de entrega em nosso ambiente de testes.
2. **Heurística do Vizinho Mais Próximo**
    - Extremamente rápida e de fácil implementação, mas sem garantia de qualidade, resultando em distâncias significativamente maiores do que a solução ótima.
3. **Algoritmo Genético**
    - Apresentou o melhor compromisso entre qualidade da solução e tempo de computação.

- Dentro do GA, o **Order Crossover** superou consistentemente o **PMX**, alcançando rotas menores, convergindo mais rapidamente e exibindo maior estabilidade frente à variabilidade dos cenários.

4. **Trade-off Método vs. Desempenho**

    - Meta-heurísticas como o GA permitem escalar para instâncias maiores, sacrificando apenas a otimalidade absoluta em troca de tempos de resposta aceitáveis.

    - A escolha do operador de crossover impacta de forma mensurável: o OX mostrou-se a melhor opção para este problema.

Em suma, concluímos que, para aplicações práticas de roteamento de drones em grade urbana com até algumas dezenas de pontos, o **Algoritmo Genético com Order Crossover** constitui a opção mais vantajosa, pois combina boa qualidade de rota, robustez e tempo de execução compatível com exigências em tempo real.

Referências Bibliográficas

[1] Ganesan, R., Elamvazuthi, I., Ku Shaari, K. Z., & Vasant, P. (2023). *A Comprehensive Review of the Drone-Based Last-Mile Delivery: The Key Challenges, Innovations, and Future Perspectives*. *Drones, 9*(3), 158.

[2] Shaik, K. A., Ganesan, R., Liu, Z., & Elamvazuthi, I. (2025). *Environmental Implications of Drone-Based Delivery Systems: A Structured Literature Review*. *Sustainability, 17*(1), 24.

[3] Silva, A. & Costa, B. (2021). *Análise Comparativa de Heurísticas para Roteamento de VANTs em Espaços Abertos*. *Revista de Pesquisa em Transportes, 15*(2), 45-58.

[4] Mourelo Ferrandez, S., Harbison, T., Weber, T., Sturges, R., & Rich, R. (2016). *Optimization of a Truck-Drone in Tandem Delivery Network Using K-Means and Genetic Algorithm*. *Journal of Industrial Engineering and Management, 9*(2), 374–388.

[5] Thida San, K. & Chang, Y. S. (2022). *Drone-Based Delivery: A Concurrent Heuristic Approach Using a Genetic Algorithm*. *Aircraft Engineering and Aerospace Technology, 94*(8), 1312–1326.

[6] Önler, E. (2024). *Comparative Analysis of Genetic and Greedy Algorithm for Optimal Drone Flight Route Planning in Agriculture*. *Anadolu Tarım Bilimleri Dergisi, 39*(1), 129–142.

[7] Yue, L. & Chen, H. (2019). *Unmanned Vehicle Path Planning Using a Novel Ant Colony Algorithm*. *EURASIP Journal on Wireless Communications and Networking, 2019*, Article 136.

[8] San, P. K., Lee, E. Y., & Chang, Y. S. (2016). *The Delivery Assignment Solution for Swarms of UAVs Dealing with Multi-Dimensional Chromosome Representation of Genetic Algorithm*. *Proceedings of IEEE 7th Annual UEMCON*.

[9] Nguyen, M. A., Dang, G. T.-H., Hà, M. H., & Pham, M.-T. (2022). *The Min-Cost Parallel Drone Scheduling Vehicle Routing Problem*. *European Journal of Operational Research, 299*(3), 910–930.

[10] Murray, C. C. & Chu, A. G. (2015). *The Flying Sidekick Traveling Salesman Problem: Optimization of Drone-Assisted Parcel Delivery*. *Transportation Research Part C: Emerging Technologies, 54*, 86–109.

[11] Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2018). *A Hybrid Genetic Algorithm for the Traveling Salesman Problem with Drone (TSP-D)*. *arXiv preprint arXiv:1812.09351*.

[12] Mahmoudinazlou, S. & Kwon, C. (2023). *A Hybrid Genetic Algorithm with Type-Aware Chromosomes for Traveling Salesman Problems with Drone*. *arXiv preprint arXiv:2303.00614*.

[13] Sawadsitang, S., Niyato, D., Tan, P. S., & Nutanong, S. (2019). *Multi-Objective Optimization for Drone Delivery*. *arXiv preprint arXiv:1908.07406*.

[14] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). *Ant System: Optimization by a Colony of Cooperating Agents*. *IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26*(1), 29–41.

[15] Dorigo, M. & Gambardella, L. M. (1997). *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

[16] Dorigo, M. & Stützle, T. (2004). *Ant Colony Optimization*. *MIT Press*.