UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
BACHARELADO EM ENGENHARIA COMPUTACIONAL

Marcelo Ian Rezende Menezes

Um algoritmo Multi-Start Iterated Greedy para o Problema de Roteamento de Veículos com Drones

Juiz de Fora
2025

Marcelo Ian Rezende Menezes

Um algoritmo Multi-Start Iterated Greedy para o Problema de Roteamento de Veículos com Drones

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Engenharia Computacional.

Orientadora: Profa. DSc. Lorenza Leão Oliveira Moreno

Coorientadora: Profa. DSc. Luciana Brugiolo Gonçalves

Juiz de Fora

2025

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Ian Rezende Menezes, Marcelo.
Um algoritmo Multi-Start Iterated Greedy para o Problema de Roteamento de Veículos com Drones / Marcelo Ian Rezende Menezes. – 2025.
57 f. : il.

Orientadora: Lorenza Leão Oliveira Moreno
Coorientadora: Luciana Brugiolo Gonçalves
Trabalho de Conclusão de Curso (Graduação) – Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Bacharelado em Engenharia Computacional, 2025.

1. Roteamento de Veículos com Drones. 2. Janelas de Tempo. 3. Q-Learning. 4. Otimização. 5. Logística. I. Moreno, Lorenza Leão Oliveira, orient. II. Gonçalves, Luciana Brugiolo, coorient. III. Título.

Marcelo Ian Rezende Menezes

Um algoritmo Multi-Start Iterated Greedy para o Problema de Roteamento de Veículos com Drones

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Engenharia Computacional.

Aprovada em 17 de março de 2025

BANCA EXAMINADORA

Profa. DSc. Lorenza Leão Oliveira Moreno - Orientadora
Universidade Federal de Juiz de Fora

Profa. DSc. Luciana Brugiolo Gonçalves - Coorientadora
Universidade Federal de Juiz de Fora

Prof. DSc. Carlos Cristiano Hasenclever Borges
Universidade Federal de Juiz de Fora

Prof. DSc. Stênio Sã Rosário Furtado Soares
Universidade Federal de Juiz de Fora

AGRADECIMENTOS

Agradeço a minha mãe Marilda e meu pai Shailon Ian, à minha família por confiarem, incentivarem e apoiarem essa jornada. Agradeço aos amigos e colegas que fiz na faculdade, a todos do Grupo de Educação Tutorial da Engenharia Computacional pelo apoio, por compartilhar momentos, pelos estudos. Agradeço aos amigos de longa data que sempre estiveram presentes, compartilhando momentos e histórias. Agradeço aos professores que me ensinaram, pela paciência, experiência e dedicação.

"Writing means sharing. It's part of the human condition to want to share things – thoughts, ideas, opinions." (Paulo Coelho (1))

RESUMO

O transporte de produtos desempenha um papel essencial na indústria global, impulsionando a busca por soluções que tornem a logística mais eficiente e sustentável. Dentro desse contexto, o Problema de Roteamento de Veículos com Drones e Janelas de Tempo (VRP-DTW) surge como uma abordagem promissora para otimizar entregas, integrando caminhões e drones para reduzir custos operacionais, minimizar a emissão de poluentes e agilizar a distribuição de mercadorias. Nesse tipo de problema, são considerados um conjunto de caminhões, todos equipados com drones, que devem atender um número definido de clientes em rotas. Enquanto os caminhões realizam essas entregas, os drones podem auxiliá-los atendendo a outros clientes. O drone se mostra uma alternativa mais barata do que o caminhão, porém, devido à baixa autonomia, deve realizar as rotas em conjunto com os veículos terrestres. Neste trabalho, propõe-se um algoritmo *Multi-Start Iterated Greedy*, que utiliza técnicas para construção da solução inicial, passa por etapas de destruição, reconstrução e refinamento, utilizando para isso um conjunto de movimentos que exploram as vizinhanças da solução, de forma a encontrar outras melhores. Assim, foram implementados dois métodos, sendo um método baseado em aprendizado por reforço, Q-learning, para guiar a ordem na qual esses movimentos são aplicados usando uma função valor-ação Q. O outro método, Random Variable Neighborhood Descent, envolve a escolha aleatória dessa ordem de movimentos. O desempenho do algoritmo foi comparado a outras abordagens da literatura que tratam o VRP-DTW. Além disso, a performance do Q-learning foi comparada à do Random Variable Neighborhood Descend (RVND) implementado. Os experimentos foram realizados considerando instâncias com 25, 50 e 100 clientes, e os resultados mostram que a abordagem proposta se mostrou competitiva em relação às da literatura, tanto na qualidade das soluções quanto no tempo computacional. E, na comparação do uso do RVND em relação ao uso do Q-learning, a abordagem RVND apresentou desempenho superior.

Palavras-chave: roteamento de veículos com drones; janelas de tempo; q-learning; otimização; logística.

ABSTRACT

The transportation of goods plays a crucial role in the global industry, driving the search for solutions that make logistics more efficient and sustainable. In this context, the Vehicle Routing Problem with Drones and Time Windows (VRP-DTW) emerges as a promising approach to optimizing deliveries by integrating trucks and drones to reduce operational costs, minimize pollutant emissions, and speed up the distribution of goods. In this type of problem, a set of trucks, all equipped with drones, must serve a defined number of customers along their routes. While trucks perform these deliveries, drones can assist by serving additional customers. The drone proves to be a cheaper alternative compared to the truck; however, due to its low autonomy, it must operate in coordination with land vehicles. In this work, a Multi-Start Iterated Greedy algorithm is proposed, which uses techniques for constructing the initial solution, followed by stages of destruction, reconstruction, and refinement, employing a set of moves that explore the solution's neighborhoods to find better ones. Two methods were implemented: one method is based on reinforcement learning, Q-learning, to guide the order in which these moves are applied using a Q action-value function. The other method, Random Variable Neighborhood Descent (RVND), involves the random selection of the order of these moves. The algorithm's performance was compared to other approaches from the literature addressing the VRP-DTW. Additionally, the performance of Q-learning was compared to the implemented RVND. Experiments were conducted considering instances with 25, 50, and 100 customers. The results show that the proposed approach proved competitive with those in the literature in terms of both solution quality and computational time. However, when comparing the use of RVND with Q-learning, the RVND approach demonstrated superior performance.

Keywords: vehicle routing with drones; time windows; q-learning; optimization; logistics.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANAC	Agência Nacional de Aviação Civil
DCC	Departamento de Ciência da Computação
GPS	Sistema de Posicionamento Global
GRASP	*Greedy Randomized Adaptative Search Procedure*
IG	*Iterated Greedy*
LC	Lista de candidatos
MIG	*Multi-Start Iterated Greedy*
MSTPH	Heurística de Duas Fases *Multi-Start*
MSTPH2	Heurística de Duas Fases *Multi-Start* com Segunda Busca Local
MIP	Programação Linear Inteira
RVND	*Random Variable Neighborhood Descent*
UAV	Veículo Aéreo Não Tripulado
UFJF	Universidade Federal de Juiz de Fora
VND	*Variable Neighborhood Descent*
VNS	*Variable Neighborhood Search*
VRP	Problema de Roteamento de Veículos
VRP-D	Problema de Roteamento de Veículos com Drones
VRP-DTW	Problema de Roteamento de Veículos com Drones e Janelas de Tempo

LISTA DE SÍMBOLOS

$\in$	Pertence
$\subset$	Está contido
$\emptyset$	Conjunto vazio
$A \setminus B$	Conjunto A menos o conjunto B
$\sum$	Somatório
$\mathcal{N}_k(x)$	Vizinhança k da solução x
$G(U, A)$	Grafo ponderado na aresta
U	Conjunto de nós
U^*	Conjunto de clientes
U^c	Conjunto de clientes que só podem ser atendidos por caminhões
A^c	Conjunto de arestas da malha para caminhão
A^d	Conjunto de arestas da malha para drones
u_0	Depósito
u_n	Cliente n, com $n \neq 0$
A_{ij}	Distância entre u_i e u_j
q_i	Demanda de u_i
a_i	Abertura da janela de tempo para u_i
b_i	Fechamento da janela de tempo para u_j
K	Conjunto da frota de caminhões
W	Capacidade do veículo
v	Velocidade do veículo
t_{ij}	Tempo de viagem de u_i a u_j
s_i	Tempo de atendimento a u_i
c	Custo de viagem por unidade de distância para o veículo
T	Tempo de espera para o drone
E	Distância máxima de voo para cada entrega para o drone
S	Solução
S_{best}	Melhor solução encontrada
$Q(t, a)$	Função estimativa ação-valor
t	Estado
a	Ação
$R(t, a)$	Recompensa imediata
η	Fator de aprendizado
γ	Fator de desconto
α	Parâmetro definidor do tamanho da lista de candidatos
k_{max}	Máximo número de iterações sem melhora para o *Iterated Greedy*
i_{max}	Máximo número de iterações para o Algoritmo *multi-start*
Q_{init}	Valor inicial para $Q(t, a)$

SUMÁRIO

1 INTRODUÇÃO

O transporte de produtos compõe uma parte vital da indústria mundial, movimentando bilhões de reais em receitas. Trata-se de um setor altamente competitivo, que busca constantemente otimizar e aprimorar a eficiência logística (2). Nesse contexto, a Pesquisa Operacional tem se concentrado no Problema de Roteamento de Veículos (VRP). Esse problema envolve a definição de rotas para veículos, geralmente caminhões, que partem de um depósito para realizar entregas a clientes localizados em pontos específicos de uma malha viária (3). O principal objetivo é minimizar custos operacionais, como a distância total percorrida, ao mesmo tempo em que a viabilidade das entregas é garantida.

O uso de drones, veículos aéreos não tripulados (UAV), em roteamento vem atraindo a atenção de grandes empresas do setor logístico, como Amazon (4, 5, 6), DHL (7) e FedEx (8). Atualmente, essa tecnologia é sugerida para diversas aplicações, incluindo organização de produtos, manutenção de inventário em grandes depósitos, inspeção de segurança e entregas de última milha. A Amazon, por exemplo, solicitou permissão à Autoridade Civil Aeronáutica do Reino Unido para realizar entregas de pacotes a partir de um de seus depósitos, por meio do serviço denominado *Prime Air*. Vale destacar que a empresa já conduziu testes similares nos Estados Unidos e na Itália, aprimorando continuamente o software e a tecnologia para viabilizar esse modelo de entrega em larga escala. No Brasil, a Agência Nacional de Aviação Civil (ANAC) autorizou, em 2022, a fabricante *Speedbird Aero* a realizar entregas comerciais com drone. Essa permissão se extende para casos "além da linha de visão do piloto" (9).

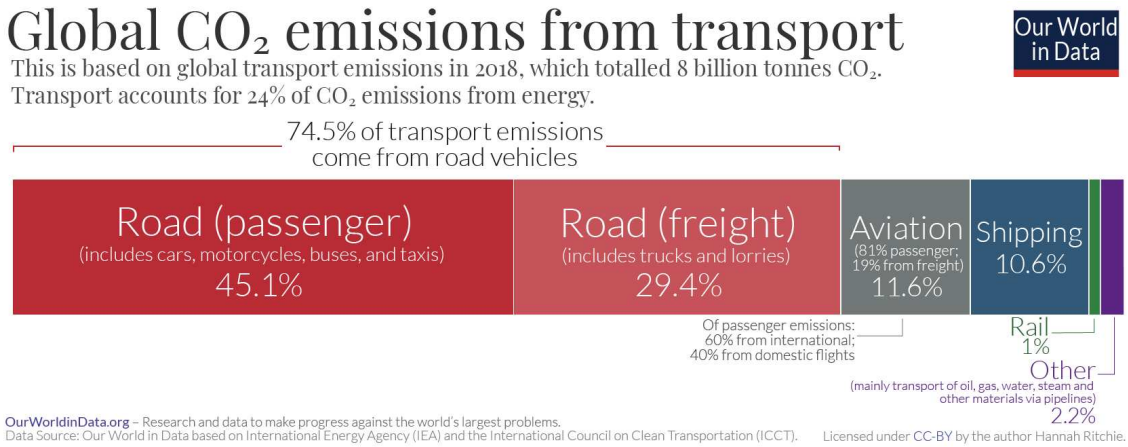
No contexto do VRP, há uma demanda crescente por soluções que reduzam os custos operacionais, ao mesmo tempo em que otimizam o tempo de entrega, minimizam a emissão de gases poluentes e diminuem o tempo de espera dos clientes. Nesse prisma, conforme abordado em (10), o Problema de Roteamento de Veículos com Drones (VRP-D) surge como um recurso promissor para tornar o transporte mais ágil e sustentável, considerando os vários aspectos relacionados à logística verde. A integração entre caminhões e drones tem se destacado como uma alternativa eficiente para otimizar o processo de entrega, especialmente na última etapa da distribuição de mercadorias, ou seja, o transporte do centro de distribuição até o cliente final. Essa estratégia permite que os drones realizem entregas diretas evitando o trânsito urbano, aumentando a velocidade das entregas e diminuindo o custo de transporte por quilômetro percorrido.

No entanto, em (11, 12), os autores destacam as dificuldades na implementação da cooperação entre caminhões e drones, seja por desafios tecnológicos ou por questões legislativas e regulamentares. Os drones são movidos a bateria, o que limita sua autonomia e capacidade de carga. Dessa forma, pesquisas vêm sendo desenvolvidas para aprimorar a eficiência energética. Além disso, sensores e motores tecnologicamente sofisticados são

necessários para garantir segurança e autonomia, o que também impacta diretamente sua viabilidade operacional. Outra limitação refere-se à operação com auxílio do GPS. Em determinadas áreas, como regiões florestais, a conexão com os UAV pode ser comprometida, prejudicando a navegação e a confiabilidade das entregas.

Entretanto, os benefícios do uso de drones não podem ser ignorados. Em (13), evidencia-se a preocupação com a sustentabilidade, considerando a emissão de gases poluentes ao longo das rotas de veículos movidos a combustão. O estudo demonstra que o uso de drones contribui positivamente para a logística verde, uma vez que a emissão de gases poluentes desses veículos são ínfimas. Vale destacar que aproximadamente um quinto das emissões globais de gás carbônico provém do setor de transporte, sendo 29.4% desse total é originado especificamente de caminhões de carga (14), e esse valor tende a aumentar nos próximos anos, conforme gráfico 1.

Figura 1 - Emissão de CO_2 por transporte



Fonte: Hannah Ritchie (2020)
Nota: "Global CO_2 emissions from transport". Publicado online em OurWorldinData.org. Consultado de: 'https://ourworldindata.org/co2-emissions-from-transport'

O uso de métodos computacionais para solucionar problemas de roteamento tem se mostrado vantajoso em termos de custo, eficiência e tempo. Modelos matemáticos permitem representar com precisão as características do VRP e suas variações que surgem a todo momento, mesmo para casos complexos e com múltiplas restrições. Além disso, esses métodos possibilitam a geração de diversas soluções alternativas para diferentes cenários, que podem ser escolhidas com base em um planejamento e análises cuidadosas (3).

Este trabalho aborda uma variação do VRP-D que considera janelas de tempo para os clientes, referenciada na literatura como Problema de Roteamento de Veículos com Drones e Janelas de Tempo (VRP-DTW, do inglês Vehicle Routing Problem With Drones and Time Windows). Em (15, 16), os autores analisaram esse problema considerando restrições de janelas de tempo dos clientes, limitação da bateria dos drones e capacidade

máxima de carga dos veículos. O objetivo principal foi minimizar os custos operacionais monetários para realizar todas as entregas utilizando uma frota de caminhões equipados com drones. Neste trabalho está sendo proposto um algoritmo *Multi-Start Iterated Greedy* (MIG), na qual é utilizado um algoritmo guloso randomizado para geração de soluções iniciais, seguido por uma etapa de destruição, reconstrução e refinamento da solução dentro do método *Iterated Greedy* (IG). No refinamento, foram desenvolvidos dois métodos. O primeiro utiliza-se um método derivado da Aprendizagem por Reforço, chamado de *Q-learning*. O *Q-Learning*, até o presente momento, não foi utilizado para o VRP-DTW, sendo usado na literatura por exemplo para o problema do caixeiro viajante (17), para o *Machine Scheduling Problem* (18), para o VRP com *crowd-shipping* (19), entre outros. O segundo método desenvolvido é o chamado *Random Variable Neighborhood Descent*, que envolve explorar diferentes vizinhanças da solução de forma aleatória. Também procura-se comparar os dois métodos implementados para instâncias com 25, 50 e 100 clientes.

O restante da monografia está assim estruturado: no Capítulo 2 há a fundamentação teórica do problema de roteamento, com heurísticas relevantes ao trabalho, no Capítulo 3 está a revisão bibliográfica do VRP-DTW, bem como alguns algoritmos existentes na literatura e a definição formal do problema, no Capítulo 4 está o algoritmo proposto. No Capítulo 5 estão informações acerca das instâncias e dos experimentos realizados e o Capítulo 6 traz as conclusões e sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo aborda-se a fundamentação teórica acerca do uso de drones no problema de roteamento de veículos, e faz-se uma breve introdução sobre metaheurísticas relevantes para o algoritmo proposto, introduzindo o conceito de vizinhança e busca local, e também destacando o *Iterated Greedy* (IG) e o *Random Variable Neighborhood Descent* (RVND). Em seguida, introduz-se a área de aprendizado por reforço, discutindo-se uma técnica chamada *Q-Learning*.

2.1 USO DE DRONES NO ROTEAMENTO

O avanço da tecnologia de drones e sua integração com caminhões têm impulsionado o estudo de novas abordagens para problemas de roteamento de veículos (10). O uso de drones no transporte de mercadorias permite reduzir tempos de entrega, otimizar custos operacionais e melhorar a eficiência logística em cenários urbanos e rurais. Nesse contexto, diversas variações do problema de roteamento com drones foram propostas na literatura, buscando explorar diferentes configurações e restrições operacionais.

O problema do caixeiro viajante usando drones foi introduzido por (12), no qual os autores sugerem duas variações. A primeira, denominada *Flying Sidekick Traveling Salesman Problem*, incorpora o uso de drones em conjunto com um único caminhão para realizar as entregas. Nesse problema, é proposta uma variação do problema do caixeiro viajante, na qual um único caminhão deve realizar entregas para um número fixo de clientes, com a introdução de um drone. Ambos os tipos de veículos devem partir e retornar a um depósito, e o drone é acoplado ao caminhão, podendo ser lançado a partir do veículo para realizar uma entrega e retornar, desde que o caminhão esteja no depósito ou atendendo um cliente. Já na segunda, chamada *Parallel Drone Scheduling Traveling Salesman Problem*, o depósito está localizado próximo aos clientes, e os drones realizam entregas paralelamente ao caminhão, que, por sua vez, atende clientes muito distantes e/ou com grande demanda. Assim, diferentemente do primeiro problema apresentado, o drone atua de forma independente, saindo e retornando ao depósito para pegar as parcelas que serão entregues aos clientes próximos. Nesse caso, considera-se que o drone possui baixa autonomia, enquanto os clientes estão suficientemente próximos do depósito. Dessa forma, o caminhão atende os clientes distantes, enquanto o drone realiza diversas entregas aos clientes próximos.

Em (11), o problema de roteamento é novamente abordado, dessa vez considerando uma frota com múltiplos caminhões, denominada VRP-D. Nesse trabalho os autores analisam o problema sob a ótica do pior caso, isto é, avaliam o aumento do custo operacional quando as entregas são realizadas exclusivamente por caminhões, sem o auxílio de drones. Para essa análise, é considerado duas frotas, a primeira uma que consiste

apenas de caminhões, e a segunda uma que considera caminhões e drones. Dessa forma, os autores demonstraram o ganho em tempo de entrega ao utilizar uma frota composta por caminhões e drones. Neste trabalho, múltiplos caminhões equipados com múltiplos drones devem realizar entregas aos clientes. Novamente, o drone atua em conjunto com os caminhões, sendo lançado a partir de um caminhão para atender outro cliente. Nota-se que, nesse problema, tanto o drone quanto o caminhão viajam pela rede rodoviária, ou seja, o caminho percorrido pelo drone é o mesmo que seria percorrido por um caminhão.

Em (20), demonstrando o uso de técnicas computacionais, o autor propõe uma abordagem baseada em colônia artificial de abelhas para o VRP-D, na qual se busca minimizar o custo operacional do roteamento. A colônia artificial de abelhas é um modelo que está dentro da categoria de algoritmos de inteligência de enxame, e, demonstra bons resultados para o VRP.

Em (21), os autores formulam um modelo matemático para a resolução do VRP-D, levando em consideração tanto o tempo de entrega quanto o tempo de viagem dos veículos. Nessa variação, foi considerada uma frota de caminhões equipada com um único drone, que deve viajar de forma sincronizada, exceto quando retorna ao depósito. De forma semelhante a outros problemas, o drone é lançado a partir de um caminhão para atender outro cliente. Nesse problema, considera-se um tempo limite para o retorno ao depósito, representando o tempo de trabalho dos funcionários responsáveis pelas entregas. Além disso, desenvolvem um algoritmo *Adaptive Large Neighborhood Search* para resolver o problema. Consiste numa etapa de destruição da solução e reconstrução posterior, na qual é sugerido diferentes métodos para tal. Os autores fizeram uma análise de sensibilidade dos parâmetros dos drones, e mostraram que mesmo para velocidades de drones menores que caminhão, ainda assim há benefícios de usá-los para reduzir tempo de entregas. Mudança na capacidade do drone também afeta significamente a solução, permitindo que drones atendam mais de um cliente por vez. Por outro lado, para drones com alta autonomia, as vantagens se tornam menos visíveis, devido à esparsidade dos clientes, ou seja, para clientes muito próximos entre si, uma maior autonomia não traz vantagens significativas para a redução de custos.

Já em (22), é apresentado um algoritmo genético híbrido para o mesmo VRP-D, no qual são geradas diversas soluções através de um algoritmo guloso pelo método do *nearest neighbor*, ou seja, seleciona o cliente mais próximo a cada passo da construção da solução. O algoritmo proposto conseguiu soluções competitivas com a literatura estudada.

Em (23), propõe-se uma abordagem baseada em *Variable Neighborhood Search* (VNS) para o VRP-D onde os drones e os caminhões seguem métricas diferentes para a distância percorrida. Enquanto os caminhões devem seguir as ruas e avenidas das cidades, os drones podem percorrer as distâncias entre os clientes de forma direta. Além disso, considera-se que o tempo para receber e enviar drones a partir de um veículo é negligível.

Os autores propõem uma decomposição do problema em um VRP, sem drones, que é considerado mais simples de resolver do que a variação com drones. Assim, é utilizado um VNS com vizinhanças para refinar o VRP e, posteriormente, um outro VNS, com outras vizinhanças, é aplicado para montar a solução para o VRP-D, inserindo drones na solução anterior. Essa abordagem mostrou resultados próximos à solução ótima para instâncias pequenas, e mostrou que há ganhos em tempo de entregas com o uso de drones.

Em (24), é apresentada uma variação do problema de roteamento com drones em que não há a restrição de limitar um drone a um único caminhão. Ou seja, um drone pode retornar a qualquer veículo terrestre a qualquer momento durante sua rota. Para isso, foram definidas duas rotas: uma para os caminhões e outra para os drones. Dessa forma, os pacotes para entrega podem ser redistribuídos entre os caminhões.

Devido à complexidade dos problemas de roteamento de veículos, classificados como NP-difíceis (25), métodos exatos para solucioná-los são viáveis apenas para instâncias com um número reduzido de clientes. No entanto, considerando que problemas reais envolvem uma quantidade significativamente maior de clientes, técnicas de Computação Inteligente têm sido amplamente exploradas. Na Seção 2.2 estão apresentados algumas metaheurísticas relevantes para este trabalho. No Capítulo 3 há a variação do problema de roteamento de veículos com drones e janelas de tempo VRP-DTW tratado, destacando algoritmos utilizados na literatura e a definição formal do problema.

2.2 METAHEURÍSTICAS

A complexidade do VRP-DTW cresce exponencialmente com o número de clientes, tornando inviável o uso de algoritmos exatos em instâncias grandes, sendo, portanto, classificado como um problema NP-difícil (25). Diante dessa complexidade, para problemas similares classificados como NP-difíceis, frequentemente são desenvolvidos algoritmos metaheurísticos, que buscam encontrar soluções aproximadas (embora não necessariamente ótimas) dentro de um tempo computacional viável (26).

Métodos exatos encontram a solução ótima para problemas de otimização ao custo de poder computacional. Assim, são desenvolvidos heurísticas para resolução de problemas específicos, e, a nível mais alto, são desenvolvidos metaheurísticas, que resolvem categorias de problemas (27). Uma heurística, no contexto da otimização, pode ser um algoritmo projetado para construir soluções de forma incremental, como nos algoritmos construtivos, ou para melhorar soluções já existentes por meio da modificação de suas partes, como nos algoritmos de busca local. Nesse contexto, há um equilíbrio entre diversificação e intensificação: a diversificação amplia a exploração do espaço de busca para identificar regiões promissoras, enquanto a intensificação foca na busca nessas regiões para refinar as soluções encontradas. Como essa informação não está disponível *a priori*, é necessário estabelecer um balanço entre esses dois processos (28).

2.2.1 Busca Local

A maioria das metaheurísticas utiliza mecanismos para realizar refinamentos nas soluções, visando encontrar outras melhores. Assim, é necessário encontrar, a partir de uma solução construída por um método construtivo, outras soluções vizinhas. Definir a vizinhança $\mathcal{N}(s)$ de uma solução s consiste em especificar os movimentos que podem ser aplicados a s (29). Geralmente, busca-se encontrar soluções vizinhas de s que sejam melhores para substituí-la, realizando pequenas alterações na solução original em um processo de refinamento. Esse processo é chamado de busca local.

No caso do problema de roteamento de veículos, trocar a ordem de visitação entre dois clientes em uma solução pode ser considerado um movimento. A vizinhança dessa solução é definida como o conjunto de todas as soluções que podem ser alcançadas pela aplicação desse movimento.

Na busca local, há duas principais políticas de refinamento da solução: primeiro aprimorante e melhor aprimorante. Na primeira, a solução de origem é alterada assim que uma solução vizinha melhor for encontrada. Na segunda, explora-se todas as soluções vizinhas para identificar a melhor delas, que então passa a ser a nova solução de origem. Dessa forma, a política de melhor aprimorante tende a realizar menos modificações, porém com alterações mais significativas no custo geral da solução. Apesar disso, a política de primeiro aprimorante pode, em média, encontrar soluções de melhor qualidade do que a outra política (30). Dessa forma, estudar o problema a ser tratado é necessário para definir qual a melhor política a ser aplicada.

A respeito da vizinhança explorada, a solução pode atingir um mínimo local, ou seja, um ponto em que não existem soluções vizinhas melhores do que a atual. No entanto, isso não garante que essa seja a melhor solução global para uma determinada instância, ou seja, o ótimo global. Como o mínimo local é definido em relação à vizinhança considerada, ao alterar a vizinhança, o mínimo local também pode mudar. Por esse motivo, muitas heurísticas desenvolvem técnicas para explorar outras regiões do espaço de busca, permitindo que a solução escape de mínimos locais.

2.2.2 Iterated Greedy

O conceito do Iterated Greedy (IG) baseia-se na destruição e reconstrução de soluções geradas por um algoritmo guloso. Inicialmente, uma solução é construída do zero por meio de um algoritmo guloso. Esse algoritmo construtivo opera de forma incremental, construindo uma solução do zero, passo a passo. Em seguida, essa solução passa por uma etapa de destruição, na qual parte de seus componentes é removida, resultando em uma solução parcial. Posteriormente, essa solução parcial é submetida a uma etapa de reconstrução, onde um algoritmo guloso é novamente aplicado para restaurar a solução previamente modificada (31, 32, 33).

Na fase de destruição, nota-se a importância da seleção dos elementos a serem removidos da solução. No contexto de roteamento, por exemplo, geralmente são escolhidos U' clientes aleatórios para remoção. No caso extremo em que U' é igual ao número total de clientes, ocorre uma destruição completa da solução, exigindo que o algoritmo guloso reconstrua inteiramente a rota. Por outro lado, se $U' = 1$, apenas um cliente é removido, o que pode limitar a exploração do espaço de busca. Assim, o valor de U' deve ser ajustado de forma a permitir um bom equilíbrio entre intensificação e diversificação da busca.

Na etapa de reconstrução, o algoritmo guloso utilizado para gerar a solução inicial costuma ser reaplicado. O objetivo dessa etapa é fornecer um novo ponto de partida para que o algoritmo construtivo continue seu processo após a perturbação causada pela destruição. Como muitos algoritmos construtivos são míopes, ou seja, consideram apenas decisões de curto prazo sem avaliar impactos futuros, modificar a solução parcial pode levar a resultados diferentes em cada iteração.

Algorithm 1: Iterated Greedy

```
   Input: S*
   Output: S*
1  while critério de parada não satisfeito do
2  |  S_p ←destruição(S*)
3  |  S ←reconstrução(S_p)
4  |  if custo(S) <custo(S*) then
5  |  |  S* ← S
6  |  end
7  end
8  return S*
```

No algoritmo 1 há a implementação mais simples do IG. A partir de uma solução construída S^*, há uma etapa de destruição, na linha 2, onde remove-se elementos de S^* para gerar uma solução parcial S_p. Essa solução parcial é então reconstruída na linha 3, gerando uma potencial nova solução S. Nas linhas 4 e 5 há o critério de aceitação, que no caso é feito pela comparação do custo da nova solução S com a antiga melhor solução S^*. Se passar no teste de aceitação, a nova solução S^* passa a ser S. O processo de repete até o critério de parada ser satisfeito (34).

2.2.3 Random Variable Neighborhood Descent

A metaheurística *Variable Neighborhood Search* (VNS) envolve a combinação sistemática de buscas locais para explorar diferentes regiões do espaço de soluções. Em uma busca local, uma solução é iterativamente aprimorada até que um ótimo local seja alcançado, ou seja, até que não seja possível encontrar uma solução melhor dentro da vizinhança da solução atual S. No entanto, como diferentes buscas locais consideram vizinhanças distintas, o VNS explora o fato de que o mínimo local em uma vizinhança

pode não ser o mesmo em outra, permitindo que o algoritmo escape de ótimos locais subótimos e explore outras regiões promissoras do espaço de busca (35, 36).

Para aplicar o VNS, é necessário inicialmente definir um conjunto de vizinhanças $\mathcal{N}_k(x)$, com $k = 1, 2, \cdots, k_{max}$, associadas à solução S. No caso do VRP, uma vizinhança pode ser definida com base na troca de clientes em uma rota, por exemplo. O algoritmo opera percorrendo essas vizinhanças de forma sistemática, onde começa investigando a vizinhança $k = 1$ utilizando uma busca local. Se uma solução melhor for encontrada, o processo continua a partir dessa nova solução. Caso contrário, o algoritmo avança para a próxima vizinhança $k = k + 1$ até atingir $k = k_{max}$.

Uma variação do VNS é o *Variable Neighborhood Descent* (VND), que adota uma estratégia mais determinística. No VND, sempre que uma nova solução melhor é encontrada, o algoritmo reinicia a busca a partir da vizinhança $k = 1$. Esse método explora a propriedade de que ótimos locais podem variar entre diferentes vizinhanças (37).

Muitos algoritmos Variable Neighborhood Descent (VND) acessa as vizinhanças de forma sequencial e pré-determinada. A mudança de vizinhança determina qual será a próxima busca a ser realizada, e a ordem escolhida para percorrer as vizinhanças pode impactar diretamente a qualidade da solução final. Existem diferentes estratégias para definir essa ordem, destacando a forma sequencial e a forma composta.

Na forma sequencial, o algoritmo percorre as vizinhanças em ordem crescente, passando para a próxima apenas quando a busca local na vizinhança atual não encontrar melhorias. Já na forma composta, uma busca local N_{k+1} é aplicada sucessivamente sobre todas as vizinhanças de k, resultando em uma estrutura aninhada do tipo $N_k(N_{k+1}(N_{k+2}(\cdots))$.

Uma das principais decisões no VNS e no VND é a definição da ordem de escolha das vizinhanças. A lista de vizinhanças pode ser embaralhada aleatoriamente ao início da execução do algoritmo. Quando uma busca local falha em encontrar uma solução melhor, a próxima vizinhança a ser explorada é escolhida com base nessa ordem aleatorizada, caracterizando o *Random Variable Neighborhood Descent* (RVND) (38, 39).

2.3 APRENDIZADO POR REFORÇO

O aprendizado por reforço é um ramo do aprendizado de máquina no qual um agente aprende por meio da experimentação de diferentes situações. A cada ação tomada, o agente recebe uma recompensa ou penalidade com base no resultado, permitindo que ele aprenda quais ações trazem maior benefício ao longo do tempo. Destaca-se que nem sempre uma ação influencia apenas o resultado imediato; em muitos casos, todas as ações subsequentes são impactadas, afetando também os resultados futuros. O aprendizado por reforço difere de outros paradigmas do aprendizado de máquina, como o aprendizado supervisionado, que se baseia na generalização das decisões a partir de dados previamente

fornecidos por um agente externo.

O aprendizado por reforço na computação, portanto, surge como uma forma de aprendizado através da interação com o ambiente, avaliando a recompensa de certas ações assim que elas são tomadas. Essa área engloba diversos métodos, inclusive o *Q-Learning*. Todos esses métodos possuem em comum quatro elementos: uma política que define a tomada de decisão do agente, uma recompensa que define o prêmio de se tomar uma ação a cada estado do sistema, uma função de valor que especifica quais decisões são boas ou ruins de se tomar longo prazo, e o modelo do ambiente, que nem sempre está presente, mas busca prever o estado resultante e a recompensa com base num estado e numa ação (40).

2.3.1 Q-Learning

O Q-learning foi introduzido por (41). Enquanto o VNS e o RVND (**2.2.3**) apresentam formas de combinar diferentes buscas locais, essas abordagens operam essencialmente por tentativa e erro, sem uma estratégia adaptativa para selecionar a melhor vizinhança a cada momento. Para contornar essa limitação, técnicas de aprendizado de máquina podem ser integradas às metaheurísticas. Uma dessas abordagens é o Q-learning, uma técnica de Aprendizado por Reforço baseado em diferenças temporais. Nesse caso, as ações ou movimentos a serem realizados são tomados considerando tentativas passadas através de recompensas. Nesse método, o algoritmo interage com o ambiente e aprende a selecionar a busca local mais apropriada em cada momento, com base em um processo de recompensa. Também é importante notar que o método converge para uma função de valor ótima independente da política utilizada, dependendo apenas que cada par ação-estado seja propriamente atualizado um número suficiente de vezes, o modelo seja um Processo de Decisão de Markov e os valores de recompensa imediata sejam limitadas por uma constante.

No método *Q-learning*, é necessário definir uma função que retorna a recompensa esperada para cada ação tomada em um determinado estado. Essa função, denominada $Q(t, a)$, representa a função ação-valor que estima o retorno esperado ao executar a ação a no estado t. O valor de $Q(t, a)$ é atualizado com base na equação de Bellman, em 2.1, que define a relação entre o valor de um estado com os valores de seus sucessores.

$$Q(t,a) = Q(t,a) + \eta \cdot [R(t,a) + \gamma \cdot \max_{a' \in \mathcal{A}} Q(t',a') - Q(t,a)] \tag{2.1}$$

$$Q(t,a) = (1-\eta) \cdot Q(t,a) + \eta \cdot [R(t,a) + \gamma \cdot \max_{a' \in \mathcal{A}} Q(t',a')] \tag{2.2}$$

Onde:

- $\eta \in [0, 1]$ é o fator de aprendizado. Valores próximos de 1 fazem com que $Q(t, a)$ tenha mudanças bruscas, enquanto valores menores próximos a 0 resultam em ajustes

mais suaves;

- $R(t, a)$ é a recompensa imediata recebida ao se realizar a ação a no estado t;
- $\gamma \in [0, 1]$ é o fator de desconto. Valores próximos de 1 o algoritmo valorize recompensas futuras determinados por $\max_{a' \in \mathcal{A}} Q(t', a')$, enquanto valores próximos a 0 priorizam resultados imediatos;
- $\mathcal{A}$ é o conjunto de ações possíveis;
- $\max_{a' \in \mathcal{A}} Q(t', a')$ representa a melhor estimativa de recompensa futura, ou seja, o maior valor de Q no próximo estado t' considerando todas as ações a'.

Para a atualização de $Q(t, a)$, é necessário escolher uma ação a a ser realizada a cada estado t. Considerando os fatores de diversificação e intensificação, as ações possíveis $\mathcal{A}$ para cada estado t devem ser exploradas um número suficientemente grande de vezes para garantir a convergência do algoritmo. No entanto, o *Q-learning* tende a priorizar ações que já demonstraram bons resultados, o que pode levar a um aprendizado enviesado, reduzindo a eficácia da solução ao limitar a exploração do espaço de busca. Nesse sentido, faz-se necessário balancear a intensificação de movimentos com boas recompensas com os movimentos de baixa recompensa. Uma estratégia adotada para contornar esse problema é o chamado *ϵ-greedy*.

Na política *ϵ-greedy*, um parâmetro $\epsilon \in [0, 1]$ é definido para determinar a proporção entre intensificação e exploração:

- Com probabilidade ϵ, a ação a é escolhida aleatoriamente;
- Com probabilidade $1 - \epsilon$, a ação selecionada é aquela que maximiza o valor de $Q(t, a)$.

Em (17), o Q-learning é utilizado como uma forma de aprendizado reativo do VNS, escolhendo a melhor busca local para o contexto do momento. Cada ação é representada como uma busca local, assim como cada estado de forma que o novo estado t_{t+1} é o movimento que acabou de ser aplicado a_t. A recompensa dada é positiva caso seja encontrado solução melhor, ou, caso contrário, remove-se a ação da lista de ações disponíveis. Esse algoritmo foi proposto para solucionar o problema do caixeiro viajante.

Para solucionar uma variação do *Machine Scheduling Problem*, (18) propôs o uso do Q-learning para guiar a escolha das buscas locais no VNS, onde os movimentos atuam tanto como estados quanto ações. A recompensa é dada com base no custo da solução, sendo positiva caso a nova solução seja melhor.

Em (19), foi implementado o Q-learning para solucionar uma variação do VRP, que inclui *crowd-shipping*. Utilizou o Q-learning para escolher o melhor movimento dado o

contexto. Cada ação é representada por um movimento e cada estado pelo movimento que acabou de ser aplicado.

Em (42), o Q-learning foi empregado para resolver o problema do caixeiro viajante, sendo utilizado em conjunto com o método GRASP. O GRASP consiste na combinação de um algoritmo construtivo guloso com uma busca local subsequente, permitindo a geração de soluções iniciais de maior qualidade, além de reter informações de iterações anteriores do algoritmo. A recompensa associada ao Q-learning é inversamente proporcional ao custo da rota entre dois clientes e proporcional ao número de vezes em que essa rota foi escolhida. Como resultado, ao final do processo, é gerada uma tabela Q que representa a melhor solução para a instância do problema do caixeiro-viajante. Adicionalmente, neste estudo, o Q-learning foi aplicado de forma cooperativa com um algoritmo genético, no qual a melhor solução de uma dada população provocava uma alteração na tabela Q. Os resultados indicaram que o uso do Q-learning contribui para a redução do tempo computacional gasto na execução, além de proporcionar uma melhoria na qualidade das soluções em comparação com a aplicação de um GRASP tradicional. Os autores também mostraram que o Q-learning pode ser integrado com outras técnicas e metaheurísticas.

3 PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM DRONES E JANELAS DE TEMPO

Esse problema representa uma variação do problema clássico de Roteamento de Veículos Capacitados com Janelas de Tempo (CVRPTW), dado que há a inclusão dos drones. O objetivo é determinar rotas para múltiplos veículos, considerando as restrições de capacidade de carga e janelas de tempo dos clientes atendidos, de forma a minimizar o custo total das viagens.

Neste Capítulo será apresentada a definição formal do Problema de Roteamento de Veículos com Janelas de Tempo (VRP-DTW). Além disso é apresentado uma revisão bibliográfica sobre o tema de roteamento de veículos com drones e janelas de tempo, com o detalhamento de alguns algoritmos utilizados especificamente para encontrar soluções viáveis para o VRP-DTW.

3.1 DEFINIÇÃO DO PROBLEMA

Conforme apresentado por (15), o VRP-DTW pode ser modelado sobre os grafos ponderados nas arestas $G_c(U, A^c)$ e $G_d(U^d, A^d)$, onde $U = u_0, u_1, u_2, \ldots, u_n$ representa o conjunto de nós, A^c é o conjunto de arestas (u_i, u_j) cujos pesos indicam a distância entre os vértices quando se utiliza um caminhão, e A^d é o conjunto de arestas (u_i, u_j) cujos pesos indicam a distância quando se utiliza um drone. Seja u_0 o depósito, $U^* = U \setminus u_0$ o conjunto de clientes a serem atendidos, e $U^d \subset U^*$ o subconjunto de clientes que, devido às características de suas demandas, podem ser atendidos por drones. Denota-se por A^c_{ij} a distância entre u_i e u_j quando percorrida por caminhão, e por A^d_{ij} a distância entre u_i e u_j quando percorrida por um drone.

Neste trabalho aborda-se a versão capacitada com frota homogênea do VRP-DTW. Assim, considera-se uma frota $K = \{k_0, k_1, k_2, \ldots, k_m\}$ composta de caminhões com igual capacidade W^c e que operam com a mesma velocidade constante v^c. Assim, o tempo de viagem t^c_{ij} de um caminhão que sai do cliente u_i até o cliente u_j é dado por $t^c_{ij} = \frac{A^c_{ij}}{v^c}$, com A^c_{ij} denotando a distância entre os clientes u_i e u_j. Além disso, também é levado em conta o tempo s^c_i de atendimento do caminhão ao cliente u_i e o custo c^c de viagem por unidade de distância percorrida por caminhão.

Cada cliente $u_i \in U^*$ deve ter sua demanda q_i completamente atendida por um único caminhão ou drone dentro de uma janela de tempo $[a_i, b_i]$, ou seja, o cliente não poderá ser atendido após o instante b_i e só poderá ser atendido a partir do instante a_i. Neste caso, o caminhão ou drone que chegar ao cliente u_i antes do instante a_i deverá esperar no cliente até a abertura da janela, sendo que no caso de atendimento por drone, este poderá esperar até no máximo T instantes de tempo, e, devido aos drones possuírem energia limitada, considera-se uma distância máxima de voo $E = \max_{ij \in U} A^d_{ij}$ para cada

entrega.

No VRP-DTW cada caminhão dispõe de uma quantidade indefinida de drones, que decolam e pousam em seus respectivos veículos. Esses drones podem ser utilizados para atender às demandas enquanto o caminhão percorre sua rota. Semelhante ao que foi definido para os caminhões, cada drone possui uma mesma capacidade W^d, mesma velocidade v^d e tempo de viagem do cliente u_i ao cliente u_j é calculado como $t^d_{ij} = \frac{A^d_{ij}}{v^d}$, com tempo de atendimento s^d_i e custo de viagem c^d por unidade de distância percorrida por drone. Além disso, o tempo de recarga ou troca de bateria dos drones é desprezado.

Cada drone atende um único cliente por vez. Assim, uma rota de drone é sempre da forma $< u_i, u_w, u_j >$, onde u_w é o cliente atendido e os clientes u_i e u_j são clientes atendidos por caminhão ao qual o drone está associado. Além disso, é necessário que o instante de chegada do drone ao cliente u_j se dê antes do caminhão já ter partido e também que o instante previsto para a decolagem de u_i seja após a chegada do caminhão neste cliente. No caso em que o drone chega a u_j antes do caminhão, é permitido que o drone espere pelo caminhão, desde que seja respeitado o limite T de tempo de espera estabelecido.

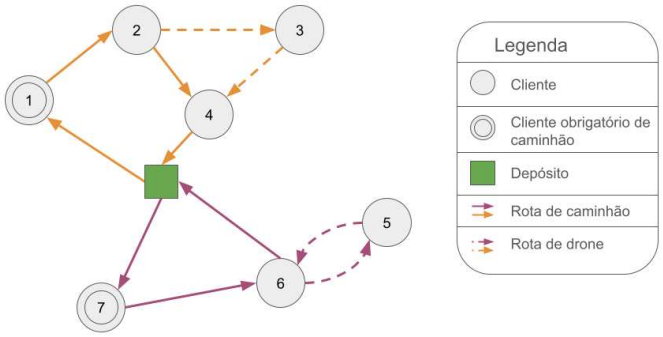
O objetivo do VRP-DTW é encontrar um conjunto de rotas de caminhão e de drones que atendam todos os n clientes, respeitando a capacidade dos veículos e a janela de tempo especificada para cada cliente, de forma a minimizar o custo total de transporte. A função de minimização do custo total da solução é exibida na equação 3.1:

$$min\ Z = c^c \times (\sum_{i \in U} \sum_{j \in U} \sum_{k \in K} A^c_{ij} \times x^k_{ij}) + c^d \times (\sum_{a \in U} \sum_{b \in U} \sum_{c \in U} \sum_{k \in K} (A^d_{ab} + A^d_{bc}) \times y^k_{abc}) \qquad (3.1)$$

onde a variável x^k_{ij} é uma variável binária que assume o valor 1 se a aresta (u_i, u_j) estiver incluída na rota do caminhão k, e 0 caso contrário. De maneira similar, a variável binária y^k_{abc} assume o valor 1 se as arestas (u_a, u_b) e (u_b, u_c) estão sendo utilizadas na rota de um drone associado ao caminhão k, e 0 caso contrário. Desta forma, a primeira parcela se refere ao custo com as entregas feitas por caminhões, enquanto a segunda parcela computa o custo das rotas dos drones. O objetivo é minimizar o custo total.

Na Figura 2, há um exemplo de solução para o VRP-DTW. Em verde tem-se o depósito u_0, de onde as rotas devem partir e retornar. No exemplo há duas rotas, uma na cor amarela e a outra na cor vermelho escuro. Em linha contínua, há o trajeto de caminhão e em linha tracejada o trajeto dos drones. Portanto, a rota amarela atende os clientes $[1 - 2 - 4]$ usando o caminhão, e o cliente 3 usando drone da forma $< 2, 3, 4 >$, ou seja, partindo do cliente 2, atendendo o cliente 3 e retornando ao caminhão no cliente 4. A rota em vermelho escuro atende os clientes $[7 - 6]$ com caminhão e o cliente 5 com drone da forma $< 6, 5, 6 >$, partindo e retornando ao caminhão no nó 6. O círculo destacado com duas bordas representam os clientes que só podem ser atendidos por caminhão, no caso os clientes 1 e 7. Os demais círculos representam clientes que podem ser atendidos

Figura 2 - Exemplo de solução para o VRP-DTW



Fonte: Elaborado pelo autor. (2025).

tanto por caminhão, quanto por drones.

3.2 REVISÃO BIBLIOGRÁFICA

O Problema de Roteamento de Veículos com Drones e Janelas de Tempo (VRP-DTW) foi inicialmente proposto por (43), incorporando todas as características do VRP-D, ou seja, múltiplos caminhões capacitados atendendo clientes em conjunto com drones vinculados aos veículos, sendo também classificado como um problema NP-difícil. Além disso, o problema introduz a restrição de janelas de tempo para os clientes, mantendo como objetivo a minimização dos custos operacionais. Em (16), os autores propõem uma heurística de duas fases com *multi-start* para o VRP-DTW. Em (44), é introduzido um algoritmo *Greedy Randomized Adaptive Search Procedure* (GRASP) Reativo com *Random Variable Neighborhood Descent* (RVND), complementado por uma fase de refinamento baseada em um modelo de Programação Linear Inteira (MIP).

Outra abordagem para a solução do VRP-DTW, porém sob a ótica da minimização do tempo total das rotas, é apresentada por (45), que desenvolveu um algoritmo exato *branch-price-and-cut* capaz de resolver instâncias de até 30 clientes. Os autores mostraram que o tempo médio de entrega pode ser reduzido com o uso de drones nas rotas, enquanto o limite de autonomia de voo dos drones influenciam pouco na solução final. Finalmente, adotar uma escala de atender o mais tarde possível contribui positivamente para a solução final, ao invés de adotar uma política de atender o mais cedo possível. No entanto, o algoritmo exato utilizado não é compatível com a variação do VRP-DTW analisada neste trabalho, pois aqui considera-se que os drones consomem energia durante o tempo de espera. No estudo original, o drone aguarda em solo, sem gasto energético, o que inviabiliza a aplicação direta do modelo à situação abordada.

Por fim, em (46), os autores esturdam uma variante do VRP-DTW com demanda

dinâmica, propondo um modelo de Programação Inteira Mista em conjunto com um algoritmo *Simulated Annealing Chimp Optimization*. O *Chimp Optimization Algorithm* é voltado para problemas de otimização contínua, enquanto o VRP-DTW é um problema de otimização combinatória. Assim, os autores desenvolveram um novo algoritmo com base nesse para solucionar a variação do VRP-DTW estudado. Como conclusão, foi determinado que o número de clientes atendidos por caminhão é positivamente correlacionado ao custo da solução, ao passo que quanto maior a concentração de clientes, menor o custo. O algoritmo proposto mostra estabilidade e consegue solucionar o problema de forma efetiva.

Em (15), foi desenvolvido um GRASP Reativo. Uma solução inicial é construída inserindo, a cada etapa, um candidato na solução, escolhido de forma aleatória a partir de uma lista restrita de candidatos, cujo tamanho é determinado por um parâmetro α, que é escolhido a partir de uma lista que se adapta a cada instância e a cada momento da execução. Em vez de fixar o valor do parâmetro α, propõe-se ajustá-lo dinamicamente a cada iteração, com base na tendência de certos valores conduzirem a soluções mais promissoras. Dessa forma, o conhecimento adquirido nas iterações anteriores é repassado para as subsequentes.

Ainda no trabalho de (15), um candidato é definido como um cliente a ser inserido na solução, seja por drone ou caminhão. Após a seleção do candidato, todas as possíveis inserções na solução parcial são avaliadas e classificadas em ordem crescente de custo. Em seguida, uma inserção aleatória é escolhida entre as $\alpha\%$ mais baratas e incorporada à solução. O critério de seleção dos clientes na lista de candidatos do algoritmo construtivo segue três estratégias distintas:

1. Demanda: Prioriza clientes com maior demanda;
2. Janela de Tempo (aleatória): Prioriza clientes com fechamento mais cedo da janela de tempo;
3. Janela de Tempo (determinística): Semelhante à segunda estratégia, porém a escolha da inserção é feita de forma determinística, sem a aleatorização.

Após a solução ser construída no algoritmo construtivo, ela passa por uma etapa de busca local, na qual o autor utilizou um RVND (ver **2.2.3**). O RVND proposto conta com 10 movimentos de busca local distintos, baseados na troca de clientes dentro da mesma rota ou entre rotas diferentes, conhecidos como *or-opt1*. Nesse tipo de movimento, um cliente atendido por caminhão ou drone é removido e reinserido na solução, e, a depender da busca local, é realocado para caminhão ou para drone. Ao todo, são 8 movimentos envolvendo a remoção e reinserção por drone/caminhão em uma mesma rota ou em rotas distintas. Os dois movimentos restantes são fixos e consistem na remoção e reinserção de u' clientes aleatórios, sendo $u' = 2$ em um caso e $u' = 3$ no outro.

No final, foi adotada uma estratégia de combinação de rotas utilizando Programação Linear Inteira (MIP). Dado que o GRASP Reativo gera um grande número de soluções, a MIP tem o objetivo de recombinar rotas entre essas soluções para encontrar uma de menor custo.

No trabalho de (16), é proposta uma heurística de duas fases do tipo *multi-start* (MSTPH), na qual, em cada fase, é resolvido um subproblema do VRP-DTW. Na primeira fase, considera-se apenas o subconjunto U^c de clientes que não podem ser atendidos por drones, de forma que, ao término dessa fase, obtém-se uma solução parcial onde todos os clientes de U^c foram alocados a um caminhão. Na segunda fase, são considerados os clientes restantes, que podem ser atendidos tanto por drone quanto por caminhão, resultando, ao final, em uma solução completa viável para o VRP-DTW.

Na primeira fase, utiliza-se uma heurística de inserção paralela, cujo objetivo é inserir um cliente em uma das rotas possíveis, podendo ser uma nova rota. Para garantir variabilidade na solução parcial gerada, é introduzido um elemento de aleatoriedade na escolha do cliente a ser inserido. Já na segunda fase, os clientes restantes são atribuídos a drones ou caminhões, verificando-se a viabilidade da solução a cada inserção, de modo que soluções inviáveis não são aceitas. Da mesma forma, um fator de aleatorização é incluído na seleção dos clientes a serem inseridos.

O termo *Multi-Start* refere-se à modificação do conjunto U^c a cada iteração do algoritmo. Inicialmente, U^c contém apenas clientes que obrigatoriamente devem ser atendidos por caminhão. Com o avanço das iterações, um novo cliente é adicionado a U^c de forma guiada, com base no menor custo de inserção.

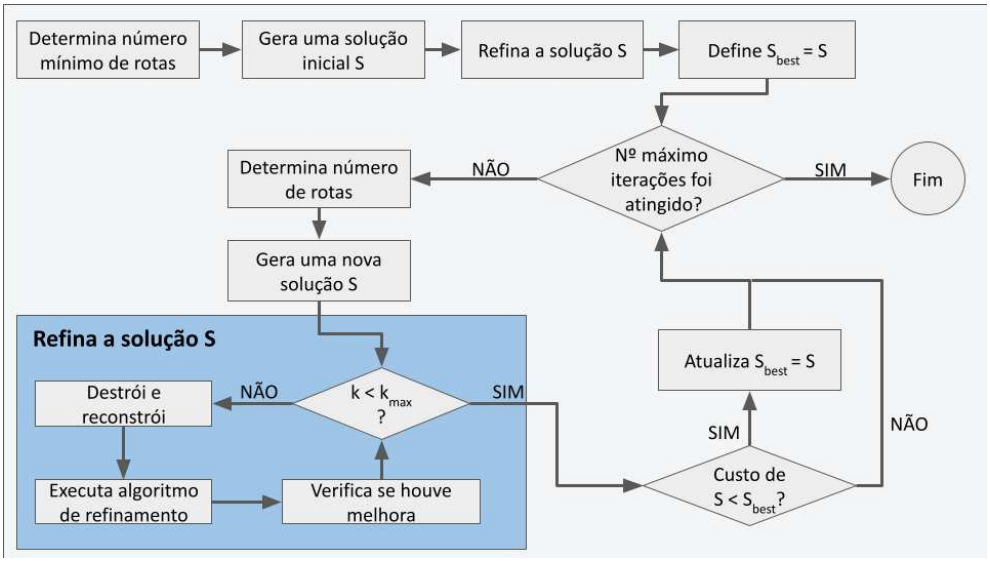
Foram desenvolvidas duas buscas locais. A primeira é aplicada ao final da segunda fase da heurística, com o objetivo de transferir clientes para serem atendidos por drones. A segunda busca local é executada sobre uma solução S' derivada de uma solução viável S. A solução S' é gerada removendo-se todos os clientes atendidos por drones em S, permitindo a reorganização das rotas para fornecer um novo ponto de partida para a segunda fase da heurística.

Dessa forma, observa-se a diversidade de técnicas existentes para a resolução do problema de roteamento de veículos com drones e janelas de tempo. Neste contexto, o presente trabalho toma como base os estudos de (16) e (15) para a elaboração e comparação dos algoritmos desenvolvidos.

4 ALGORITMO PROPOSTO

Neste Capítulo, apresenta-se a metodologia empregada para solucionar o problema, incluindo os algoritmos utilizados. Propõe-se um método *multi-start* para a geração de soluções, que se inicia com um algoritmo guloso randomizado, seguido por etapas de destruição, reconstrução e refinamento da solução por meio do método *Iterated Greedy*. Na etapa de refinamento, sugere-se o uso do *Q-learning* para selecionar a melhor sequência de buscas locais a serem aplicadas. Como alternativa, também é proposto o método *Random Variable Neighborhood Descent* (RVND), no qual a sequência de buscas locais é escolhida de forma aleatória.

Figura 3 - Fluxograma para o MIG



Fonte: Elaborado pelo autor. (2025).

O fluxograma do processo completo para o *Multi-Start Iterated Greedy* (MIG) pode ser visto na Figura 3. Inicialmente, determina-se o número mínimo de rotas para gerar uma solução inicial por meio de um algoritmo guloso randomizado. Em seguida, a solução é refinada utilizando o método *Iterated Greedy* (IG). Esse processo será detalhado na Seção 4.1, enquanto a construção da solução será abordada na Seção 4.2.

A etapa de refinamento da solução ocorre após a construção da solução S (tanto após a construção da solução inicial quanto após a contrução de uma nova solução, vide Figura 3). Nessa etapa, S é parcialmente destruída, gerando uma solução incompleta que é reconstruída por meio do algoritmo guloso randomizado utilizado em etapas anteriores. Em seguida, um algoritmo de refinamento é acionado, podendo ser o *Q-Learning* (MIG-QL) ou o *RVND* (MIG-RVND), ambos responsáveis por selecionar a ordem das vizinhanças a serem exploradas. A nova solução refinada é comparada com a solução S original, e o processo se repete até que o critério de parada seja satisfeito, ou seja, até que transcorram k

iterações sem melhora na solução. Essa etapa será detalhada na Seção 4.4. Os movimentos que compõem o refinamento serão apresentados na Seção 4.3, enquanto o *Q-Learning* será abordado na Subseção **4.5.1** e o *RVND* na Subseção **4.5.2**.

4.1 ALGORITMO MULTI-START

Algorithm 2: Algoritmo Multi-start

Input: $\alpha,\ \beta,\ i_{max},\ k_{max}$
Output: S_{best}
1 $R_{min} \leftarrow inicializaRotas(\ )$
2 $S \leftarrow algoritmoGuloso(\alpha = 0,\ R_{min})$
3 $S_{best} \leftarrow iteratedGreedy(S,\ \alpha,\ \beta,\ k_{max})$
4 $i \leftarrow 0$
5 **while** $i < i_{max}$ **do**
6 $\quad R_{min} \leftarrow aleatorizaRotas(\ )$
7 $\quad S \leftarrow algoritmoGuloso(\alpha,\ R_{min})$
8 $\quad S^* \leftarrow iteratedGreedy(S,\ \alpha,\ \beta,\ k_{max})$
9 $\quad$ **if** $custo(S^*) < custo(S_{best})$ **then**
10 $\quad\quad S_{best} \leftarrow S^*$
11 $\quad$ **end**
12 $\quad i \leftarrow i + 1$
13 **end**
14 **return** S_{best}

O termo *multi-start* refere-se à mudança da configuração utilizada na criação da solução inicial. Nesse contexto, o parâmetro que define o número mínimo de rotas para a construção da solução, conforme será descrito na Seção 4.2, é modificado a cada iteração do algoritmo *multi-start*. O processo é apresentado no Algoritmo 2.

Inicialmente, na linha 1, o número mínimo de rotas é definido como $\frac{\sum_{u_i \in U} q_i}{W^c} + 1$, ou seja, representa a menor quantidade de rotas necessárias para realizar todas as entregas, considerando apenas a demanda total dos clientes e a capacidade dos caminhões. Em seguida, na linha 2, a solução S é construída utilizando o algoritmo guloso descrito na Seção 4.2, com $\alpha = 0$, o que implica em uma solução determinística, sem aleatorização na escolha dos candidatos. Na linha 3, a solução gerada passa pelo algoritmo *Iterated Greedy*, conforme será visto na Seção 4.4.

A partir da linha 5, o processo se repete até no máximo i_{max} iterações, sempre aleatorizando as rotas iniciais na linha 6. Esse processo seleciona aleatoriamente uma entre quatro opções:

- Mantém o número de rotas da iteração anterior;
- Incrementa em 1 unidade o número de rotas da iteração anterior;

- Reduz em 1 unidade o número de rotas da iteração anterior;
- Redefine para o valor mínimo possível.

Nas linhas 7 e 8, uma nova solução é criada e refinada, utilizando os parâmetros α e β para definir o tamanho da destruição e da lista de candidatos (LC). Ao final, a melhor solução viável S_{best} encontrada é retornada.

4.2 ALGORITMO CONSTRUTIVO GULOSO RANDOMIZADO

Algorithm 3: Algoritmo Construtivo Guloso Randomizado

Input: $\alpha,\ R_{min}$
Output: S
1 $S \leftarrow \emptyset$;
2 $\text{CNA}^c \leftarrow U^c$
3 $\text{CNA} \leftarrow U^* - U^c$
4 $S \leftarrow inicializaRotas(R_{min}, \text{CNA}^c, \text{CNA})$
5 $S \leftarrow insercao(\text{CNA}^c,\ \alpha)$
6 $S \leftarrow insercao(\text{CNA},\ \alpha)$
7 **return** S

Para construir uma solução inicial, foi desenvolvido um algoritmo guloso randomizado que consiste na inserção sistemática de clientes nas rotas de caminhões ou drones. O algoritmo proposto possui duas etapas: a primeira de atendimento dos clientes que devem obrigatoriamente ser atendidos por caminhões; na segunda, os demais clientes que ainda não foram atendidos são inseridos na solução, podendo ser atendidos por drones ou caminhões.

Seja U o conjunto de nós, $U^* = U \setminus \{u_0\}$ o conjunto de clientes (sem o depósito u_0), U^c o conjunto de clientes que só podem ser atendidos por caminhões, e o conjunto resultante $U^* - U^c$, o conjunto de clientes que podem ser atendidos tanto por drones quanto por caminhões. O Algoritmo 3 descreve o processo de construção de uma solução.

Inicialmente, são criadas duas listas de clientes: CNA^c, que corresponde aos clientes que só podem ser atendidos por caminhão e ainda não estão na solução, e CNA, que corresponde aos demais clientes. Com essas listas, a solução passa por uma etapa de inicialização de rotas, onde é criado um número de rotas equivalente a R_{min}, passado como parâmetro. Cada uma dessas rotas iniciais é composta apenas por um par depósito-cliente. Há uma prioridade para inicializar esse par sempre com clientes da lista CNA^c. Caso isso não seja possível devido ao fato de R_{min} ser maior que o tamanho da lista CNA^c, são utilizados os clientes da lista CNA. Nota-se que a escolha do cliente a ser inserido na rota impacta na construção da solução final. Portanto, foi desenvolvido uma abordagem na

qual o cliente é selecionado com base na sua posição na malha, priorizando aqueles mais distantes de uma rota pré-existente.

Na linha 5, realiza-se a etapa de inserção dos clientes CNA^c, detalhada no Algoritmo 4, denominado "Algoritmo de Inserção de Clientes". Observa-se que esses clientes só podem ser atendidos por caminhões, portanto, ao final dessa etapa, obtém-se uma solução parcial na qual uma ou mais rotas de caminhão atendem a todos esses clientes.

Algorithm 4: Algoritmo de Inserção de Clientes

```
   Input: C, α
   Output: S
 1 LC ← candidatos(S, C);
 2 while C ≠ ∅ do
 3 |   if LC = ∅ then
 4 |   |   S ← insereCliente(u_0, C_0, R_{n+1})
 5 |   else
 6 |   |   C* ← selecionaCandidato(LC, α)
 7 |   |   S ← insereCliente(C*)
 8 |   |   C ← remove(C*)
 9 |   |   LC ← atualiza(C*)
10 |   end
11 end
12 return S
```

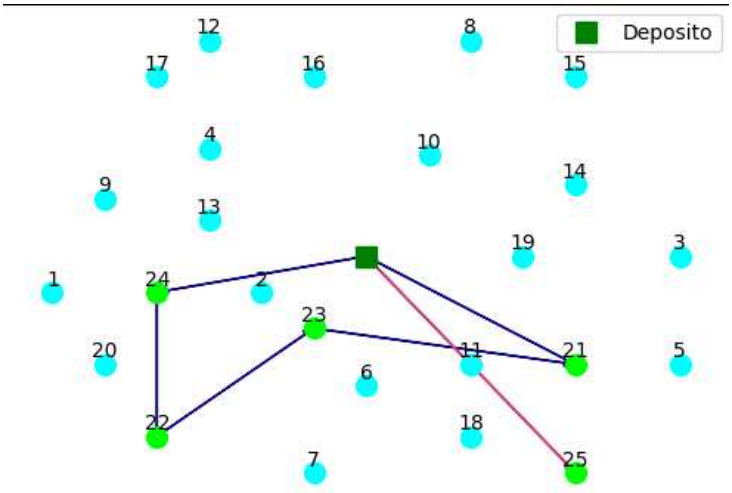
A cada iteração, gera-se a lista de candidatos LC, que engloba todas as inserções possíveis de clientes em todas as posições de cada rota, formando uma tupla *(cliente, rota, posição)*. Também é armazenada a informação sobre se o cliente será atendido por drone ou por caminhão. Essa lista é ordenada por custo, ou seja, as inserções de menor custo aparecem primeiro. O custo de ir de um cliente u_i a um cliente u_j é calculado por $custo_{ij} = A_{ij} \cdot c$, onde A_{ij} indica a distância entre os dois clientes e c o custo por unidade de distância. Nota-se que tanto o custo quanto a distância percorrida variam conforme o veículo responsável pela entrega.

Em seguida, caso existam inserções possíveis em LC, seleciona-se aleatoriamente um candidato C^* dentre os $\alpha \cdot |LC|$ primeiros candidatos de LC (linha 6), com $\alpha \in]0, 1]$. A inserção é realizada na linha seguinte, e o conjunto de clientes não atendidos, C, é atualizado removendo-se o cliente recém atendido. Da mesma forma, atualiza-se LC, removendo-se os candidatos inválidos, ou seja, aqueles que se tornaram inviáveis, e inserindo-se novas possibilidades de inserção.

Se não houver inserção possível, ou seja, $LC = \emptyset$, na linha 4 cria-se uma nova rota R_{n+1}, que é inicializada com o primeiro cliente do conjunto C, chamado C_0, partindo do depósito u_0. O processo se repete até que C esteja vazio. A Figura 4 ilustra o fim desta primeira etapa de inserção, em que os círculos verdes representam os clientes atendidos obrigatoriamente por caminhão, enquanto os azuis indicam os clientes que podem ser

atendidos tanto por caminhão quanto por drones. Note na figura que, ao final dessa etapa, o algoritmo criou duas rotas de caminhão, sendo uma delas responsável por atender um único cliente (nó 25).

Figura 4 - Solução parcial após a primeira etapa do algoritmo guloso



Fonte: Elaborado pelo autor. (2023).

Na segunda etapa, descrita na linha 6 do Algoritmo 3, um processo semelhante é realizado até que todos os demais clientes sejam atendidos. Nesta etapa, a lista LC também inclui atendimentos feitos por drones, sendo que o custo é calculado considerando tanto a ida quanto a volta do drone.

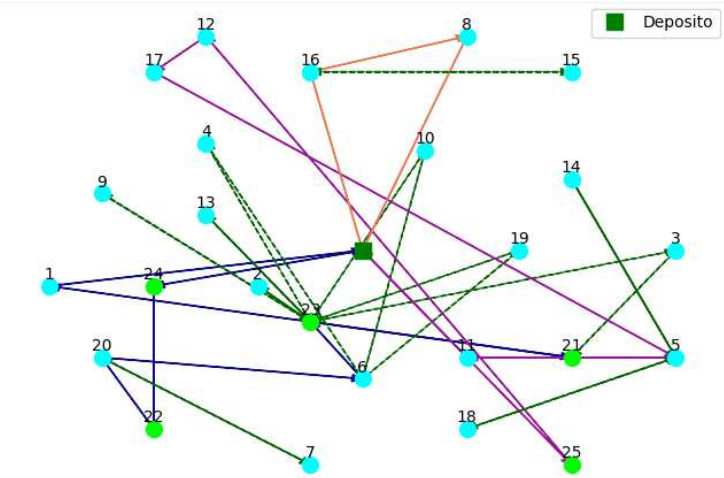
Em ambas as listas, apenas inserções viáveis são consideradas, ou seja, devem respeitar a capacidade máxima do caminhão, as janelas de tempo e, no caso dos drones, a inserção só será permitida se o caminhão correspondente estiver disponível no nó de partida e, no nó de chegada, dentro do tempo máximo de espera T.

A Figura 5 ilustra a solução final obtida pelo algoritmo, na qual três rotas de caminhão são representadas pelas cores laranja (com dois clientes), rosa (com cinco clientes) e azul (com sete clientes). Cada uma dessas rotas possui suas respectivas rotas de drone, representadas por linhas tracejadas na cor verde. Observa-se que os clientes 2, 3, 4, 7, 9, 10, 13, 14, 15, 18 e 19 são atendidos por drones, enquanto os demais são atendidos por caminhões. A solução final gerada pelo algoritmo é sempre viável.

4.3 MOVIMENTOS

Faz-se necessário definir movimentos para explorar as vizinhanças de S, ou seja, o conjunto de soluções que podem ser alcançados a partir de S quando se aplica um movimento.

Figura 5 - Solução completa após a segunda etapa do algoritmo guloso



Fonte: Elaborado pelo autor. (2023).

Os movimentos escolhidos foram baseados nos de (44). Foram definidas ao todo 9 buscas locais, cada uma explorando uma vizinhança de S, e, ao final, retorna-se um ótimo local pelo método do melhor aprimorante, ou seja, investiga-se todas as vizinhanças e seleciona-se a melhor delas. Os movimentos são os seguintes:

1. 2-opt: consiste na reorganização de dois clientes numa mesma rota a fim de remover cruzamentos entre seus caminhos;

2. or-opt1-truck-truck: consiste na remoção de um cliente atendido por caminhão e reinserção dele também na rota deste mesmo caminhão, *intra* rota;

3. or-opt1-truck-drone: consiste na remoção de um cliente atendido por caminhão e reinserção dele na rota de um drone, *intra* rota;

4. or-opt1-drone-truck: consiste na remoção de um cliente atendido por drone e reinserção dele na rota de um caminhão, *intra* rota;

5. or-opt1-drone-drone: consiste na remoção de um cliente atendido por drone e reinserção dele também na rota de um drone, *intra* rota;

6. shift-opt1-truck-truck: consiste na remoção de um cliente atendido por caminhão e reinserção dele na rota de outro outro caminhão, *inter* rota;

7. shift-opt1-truck-drone: consiste na remoção de um cliente atendido por caminhão e reinserção dele na rota de um drone, *inter* rota;

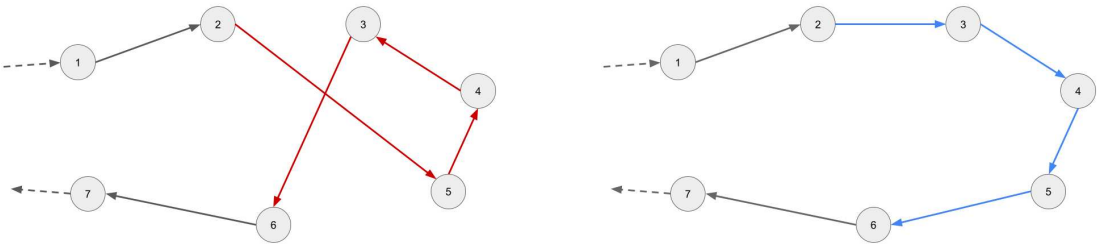
8. shift-opt1-drone-truck: consiste na remoção de um cliente atendido por drone e reinserção dele na rota de um caminhão, *inter* rota;

9. shift-opt1-drone-drone: consiste na remoção de um cliente atendido por drone e reinserção dele na rota de outro drone, *inter* rota.

Conforme a implementação, as rotas de caminhão são representadas por listas, de forma semelhante a (16). Para cada caminhão, a rota é uma lista de nós visitados, iniciando no depósito u_0. Para cada caminhão, há uma lista de tuplas que representam as rotas dos drones que o acompanham: $< u_i, u_w, u_j >$, onde u_i e u_j são os nós de partida e chegada, respectivamente, e estão contidos na rota de caminhão correspondente. O cliente atendido por drone é representado por u_w. Assim, um movimento *intra* rota é aquele que ocorre em uma única rota de caminhão, enquanto os movimentos *inter* rota são aqueles que ocorrem em duas ou mais rotas simultaneamente.

Nas figuras 6, 7, 8, 9 e 10 estão ilustrados os movimentos implementados. Em cada figura, em cinza estão representados as rotas que permaneceram inalteradas após o movimento, em vermelho as rotas que participavam da solução anteriormente ao movimento e em azul as rotas que passaram a participar da solução após o movimento. Em pontilhado são as rotas de drones e em linha contínua de caminhão. Ressalta-se que estão representados apenas trechos de uma solução. Nas figuras 6, 7 e 8 apenas um caminhão realiza a entrega no trecho representado. Nas figuras 9 e 10, dois caminhões realizam entregas simultaneamente.

Figura 6 - Ilustração do movimentos 2-opt



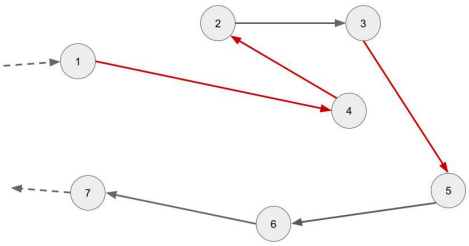
(a) Rota antes 2-opt (b) Rota após 2-opt

Fonte: Elaborado pelo autor. (2025).

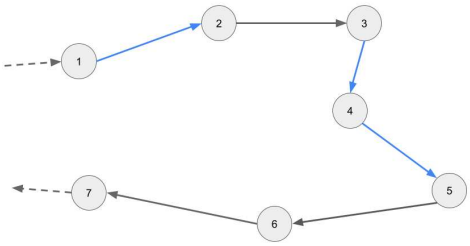
Como observa-se na Figura 6, o movimento busca eliminar cruzamentos entre rotas, potencialmente diminuindo o custo de solução. Nesse caso, a ordem de visita dos clientes é alterada, de $[1-2-5-4-3-6-7]$ passa a ser $[1-2-3-4-5-6-7]$.

Na Figura 7, é representado os movimentos que envolvem a remoção e reinserção de um cliente que anteriormente era atendido por caminhão. No caso, o cliente 4 era atendido após o cliente 1 e passa a ser atendido após o cliente 3, onde no caso da Figura 7b é atendido por caminhão, e na Figura 7d é atendido por drone. Observa-se que o cliente 4 continua a ser atendido na mesma rota, sem ocorrer mudança de caminhão.

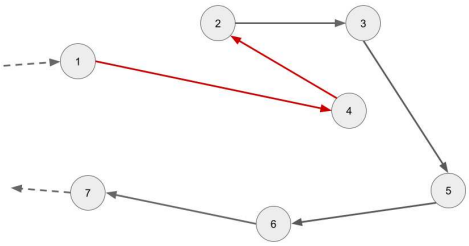
Figura 7 - Ilustração do movimentos or-1opt-truck



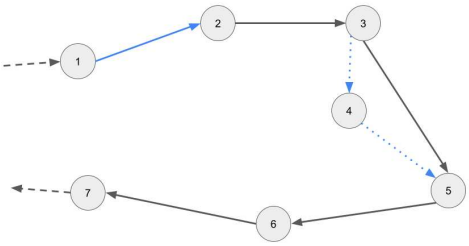
(a) Rota antes or-opt1-truck-truck



(b) Rota após or-opt1-truck-truck



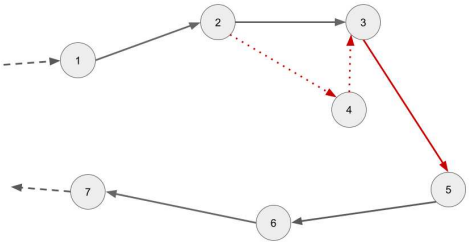
(c) Rota antes or-opt1-truck-drone



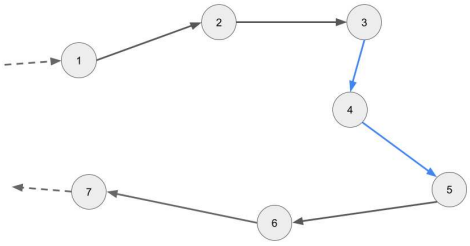
(d) Rota após or-opt1-truck-drone

Fonte: Elaborado pelo autor. (2025).

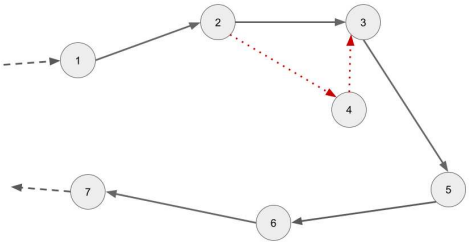
Figura 8 - Ilustração do movimentos or-1opt-drone



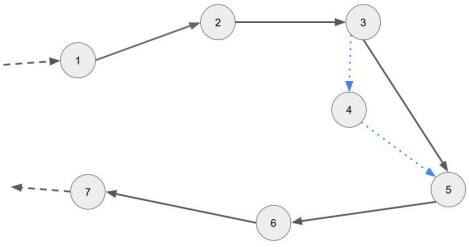
(a) Rota antes or-opt1-drone-truck



(b) Rota após or-opt1-drone-truck



(c) Rota antes or-opt1-drone-drone

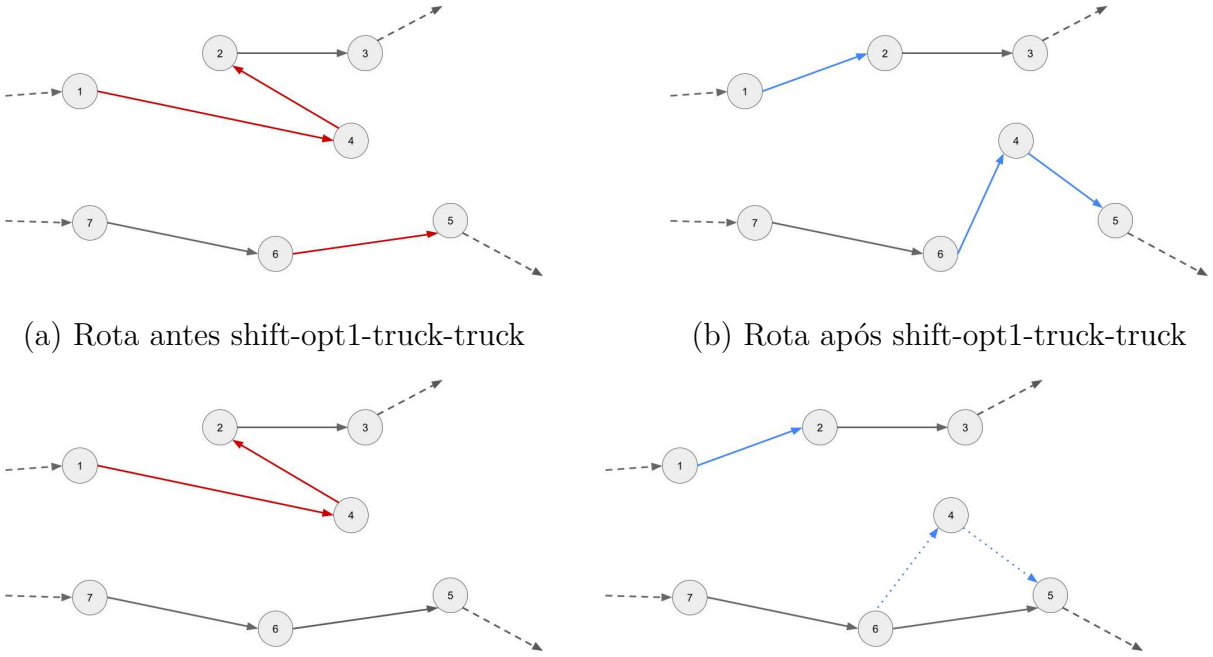


(d) Rota após or-opt1-drone-drone

Fonte: Elaborado pelo autor. (2025).

Na Figura 8, o cliente 4 é atendido por um drone atrelado ao caminhão da rota. No caso da Figura 8b, esse cliente deixa de ser atendido pelo drone e passa a ser atendido diretamente pelo caminhão. Na Figura 8d, o cliente continua a ser atendido por drone, porém é atendido em outro momento, ou seja, o ponto de partida do drone deixa de ser o cliente 2 e passa a ser o cliente 3, e o mesmo ocorre com o nó de chegada, do cliente 3 passa a ser o cliente 5.

Figura 9 - Ilustração do movimentos shift-1opt-truck



(a) Rota antes shift-opt1-truck-truck

(b) Rota após shift-opt1-truck-truck

(c) Rota antes shift-opt1-truck-drone

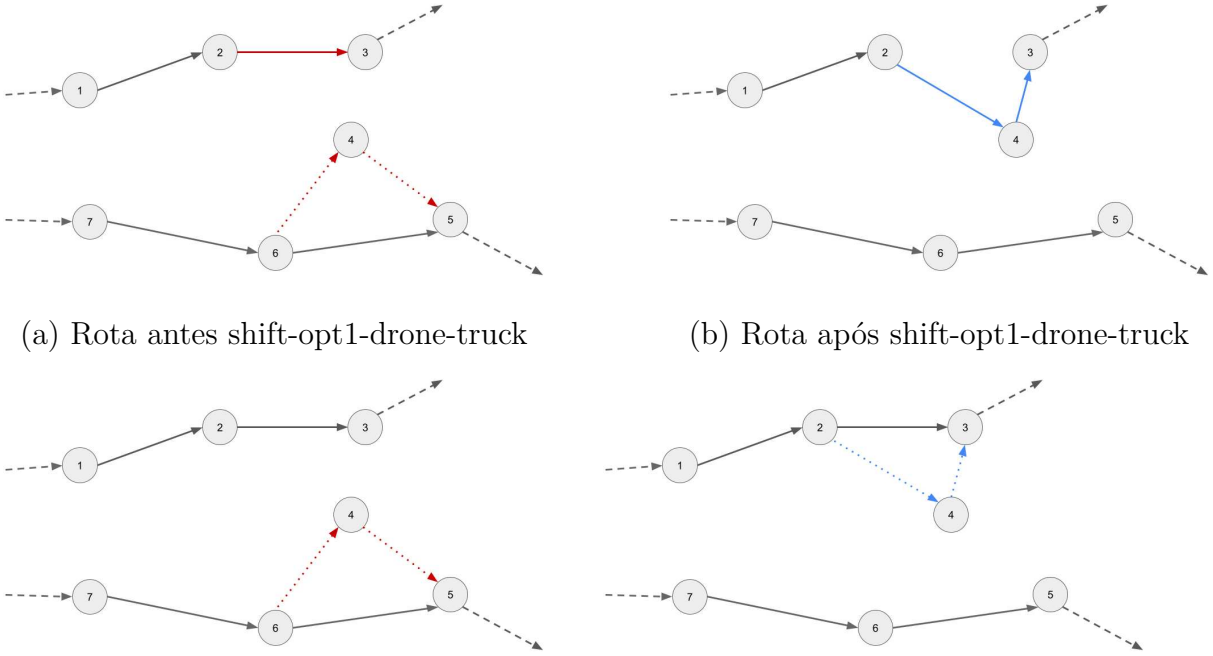
(d) Rota após shift-opt1-truck-drone

Fonte: Elaborado pelo autor. (2025).

Na Figura 9, estão representadas duas rotas de caminhão. O cliente 4 está inicialmente sendo atendido na rota $[1-4-2-3]$ e, como o movimento é *inter* rota, há a mudança de caminhão. No caso da Figura 9b, o cliente passa a ser atendido diretamente pelo outro caminhão, passando a fazer parte da rota $[7-6-4-5]$, enquanto o outro caminhão passa a atender os clientes $[1-2-3]$. No caso da Figura 9d, o cliente 4 passa a ser atendido por um drone atrelado a outro caminhão, ocorrendo então a mudança de rota.

Na Figura 10, estão representados duas rotas de caminhão e a mudança de rota do cliente 4, que inicialmente está sendo atendido por um drone. Na 10b, o cliente 4 é inicialmente atendido por um drone atrelado a um caminhão e passa a ser atendido diretamente por um caminhão diferente, pertencente a outra rota. No caso da 10d, o cliente passa a ser atendido por um drone atrelado a outro caminhão.

Figura 10 - Ilustração do movimentos shift-1opt-drone



(a) Rota antes shift-opt1-drone-truck

(b) Rota após shift-opt1-drone-truck

(c) Rota antes shift-opt1-drone-drone

(d) Rota após shift-opt1-drone-drone

Fonte: Elaborado pelo autor. (2025).

Algorithm 5: Iterated Greedy

Input: $S,\ \alpha,\ \beta,\ k_{max}$
Output: S_{best}

$k \leftarrow 0$;
inicializaRefinamento( )
while $k < k_{max}$ **do**
  $S^* \leftarrow S$
  $S^* \leftarrow destroi(\beta)$
  $S^* \leftarrow reconstroi(\alpha)$
  $S^* \leftarrow refinamento(S)$
  if $custo(S^*) < custo(S)$ **then**
    $S \leftarrow S^*$
    $\beta \leftarrow ajustaBeta(\ )$
    if $custo(S) < custo(S_{best})$ **then**
      $S_{best} \leftarrow S$
      $k \leftarrow 0$
    end
  else
    $k \leftarrow k + 1$
    $\beta \leftarrow ajustaBeta(\ )$
  end
end
return S_{best}

4.4 ITERATED GREEDY

De forma a escapar de bacias de atração no espaço de busca e aumentar a diversidade das soluções geradas, foi desenvolvido um algoritmo *Iterated Greedy* (IG). Ele consiste em destruir parcialmente uma solução S^* e reconstruí-la novamente. Após essa etapa, é realizado um refinamento através do método *Q-learning*. O Algoritmo 5 apresenta esse processo.

Na linha 2 do algoritmo, são inicializados parâmetros para a etapa de refinamento. No caso do RVND essa função não é necessária, sendo utilizada apenas para o Q-Learning, onde a tabela Q é inicializada. A decisão de inicializá-la nessa etapa visa preservar informações sobre os movimentos ao longo das iterações do IG. A partir da linha 3, há um laço cuja condição de parada é definida pelas k_{max} iterações sem melhora da solução.

Nas linhas 5 e 6, são realizadas as etapas de destruição e reconstrução da solução S^*. Na fase de destruição, são selecionados pelo menos $\beta \cdot |U|$ clientes aleatórios para remoção da solução, independentemente de terem sido atendidos por caminhão ou drone. Caso um cliente atendido por caminhão seja removido, verifica-se se há clientes atendidos por drones que possuam esse nó em comum, seja no ponto de chegada ou de partida, e esses também são removidos. Após a remoção parcial dos clientes, a reconstrução é realizada utilizando o mesmo método apresentado na Seção 4.2, considerando tanto drones quanto caminhões.

Na linha 7, é feito um refinamento da solução, podendo ser através do *Q-learning*, que será apresentado na Subseção **4.5.1** ou, no caso do MIG-RVND, nessa etapa é acionado o algoritmo RVND no lugar, que será mostrado na Subseção **4.5.2**.

Se houver melhora na solução S, esta é atualizada e o valor de β é reduzido na linha 10, a fim de limitar a destruição e favorecer a intensificação. Na ausência de melhoria, β é aumentado com o objetivo de diversificar a busca. Esse mecanismo de ajuste é acionado periodicamente ao longo da execução do algoritmo.

Ao final do algoritmo, a melhor solução viável encontrada S_{best} é retornada.

4.5 ALGORITMOS DE REFINAMENTO

Neste trabalho, foram propostas duas estratégias distintas de seleção de movimentos para um algoritmo de refinamento, baseadas no método Q-Learning e no método RVND. O Q-Learning tem como objetivo definir a ordem na qual as buscas locais são executadas através do aprendizado por reforço. O RVND, diferentemente do Q-Learning, embaralha a ordem das buscas locais a serem realizadas. O Q-Learning é detalhado na Subseção **4.5.1** e o RVND na Subseção **4.5.2**

Algorithm 6: Q-Learning

Input: $S,\ Q(t,a),\ \epsilon,\ \eta,\ \gamma$
Output: S

```
1  M ← {m_0, m_1, ..., m_n}
2  t ← Q(t,a)_best
3  while M ≠ ∅ do
4      p ← random(]0,1])
5      if p < ε then
6          a ←selecionaAçãoAleatória(M)
7      else
8          a ←selecionaAçãoComPeso(Q(t,a'))
9      end
10     S* ←buscaLocal(S, a)
11     t ← a
12     if custo(S*) < custo(S) then
13         S ← S*
14         R(t,a) ← (custo(S) − custo(S*)) * Q(t,a)_num
15         M ← {m_0, m_1, ..., m_n}
16     else
17         R(t,a) ← 0
18         M ← M \ {a}
19     end
20     Q(t,a) ←atualizaQ(ε, η, γ, R(t,a))
21 end
22 return S
```

4.5.1 Q-Learning

No Algoritmo 6, está representado o método *Q-learning* implementado. A ação a pode ser definida como um movimento de busca local aplicado à solução S^*. Assim, o estado t nesse problema pode ser definido como a última ação a tomada. Dessa forma, $Q(t,a)$ pode ser modelado como uma matriz, onde as linhas representam os estados e as colunas representam as ações, de modo a medir quais são as melhores ações a' a serem tomadas para cada estado $t = a$.

Na linha 1, é inicializado o conjunto M de movimentos, que definem as vizinhanças da solução. Em seguida, na linha 2, é necessário definir um estado inicial t, que é escolhido como o estado que leva à melhor ação a ser tomada, representado por $Q(t,a)_{best}$.

No algoritmo, é utilizada a estratégia *ϵ-greedy* para aumentar a exploração do espaço de busca, permitindo que novas ações sejam tomadas. Há uma probabilidade $p < \epsilon$ de a ação a ser selecionada aleatoriamente dentro do estado t. Caso contrário, a ação a é escolhida com base em um peso, conforme mostrado na linha 8. Essa abordagem difere da implementação usual, onde é escolhida a ação $\max_{a \in \mathcal{A}} Q(t,a)$, ou seja, a melhor ação a a partir do estado t. A função *selecionaAçãoComPeso* escolhe a ação a com base no

valor de Q para o estado atual t. O movimento é aplicado com base na ação a escolhida. Só são aceitas soluções viáveis, portanto, caso não seja encontrada, retorna-se a solução de entrada. A probabilidade de cada ação a' ser escolhida é definida como mostrado na Equação 4.1:

$$P(a') = \frac{Q(t, a')}{\sum_{a'' \in Q(t,a)} Q(t, a'')} \tag{4.1}$$

Ou seja, as melhores ações a de t possuem maior probabilidade de serem escolhidas. Isso permite que $Q(t, a)$ apresente maior variabilidade nas escolhas das ações. Nas linhas 10 e 11, aplica-se em S o movimento escolhido, gerando uma nova solução S^* potencialmente melhor, e o novo estado t é definido como a ação a.

Se o custo dessa nova solução S^* for menor que o custo da solução anterior S, então S é atualizada. A recompensa $R(t, a)$ da ação realizada é definida como $(custo(S) - custo(S^*)) * Q(t, a)_{num}$, de forma que soluções melhores resultam em recompensas positivas. Com o objetivo de reforçar a consistência de determinadas decisões ao longo da execução, foi incorporado um fator de multiplicação $Q(t, a)_{num}$ à recompensa, proporcional ao número de vezes que uma ação a foi aplicada em um estado t. Essa modificação busca valorizar ações que, mesmo com recompensas decrescentes ao longo das iterações, mantêm bom desempenho acumulado, favorecendo uma intensificação guiada por histórico. Na linha 15, todos os movimentos $m_0, m_1, \cdots, m_n$ retornam ao grupo de movimentos M.

Caso não haja melhora, a recompensa $R(t, a)$ é definida como 0, e a ação a é removida de M.

Por fim, na linha 20, é feita a atualização de $Q(t, a)$ com base na fórmula 4.2, mostrada na Subseção **2.3.1**.

$$Q(t, a) = (1 - \eta) \cdot Q(t, a) + \eta \cdot [R(t, a) + \gamma \cdot \max_{a' \in Q(t', a)} Q(t', a')] \tag{4.2}$$

Conforme explicado anteriormente, η é a taxa de aprendizado, γ é o fator de desconto, $R(t, a)$ é a recompensa imediata e $\max_{a' \in Q(t', a)} Q(t', a')$ representa o maior valor de Q para o próximo estado t'.

4.5.2 Random Variable Neighborhood Descent

O algoritmo 7 ilustra o RVND implementado. Uma lista de movimentos M é definida e embaralhada a cada execução do algoritmo, com o objetivo de explorar diferentes vizinhanças da solução de entrada.

No algoritmo, é definido a lista de movimentos M com todos os $n = 9$ movimentos listados na Seção 4.3. Essa lista é embaralhada na linha 2, dando o elemento aleatório

Algorithm 7: RVND

```
   Input: S
   Output: S
 1 M ← {m_0, m_1, ..., m_n}
 2 M ← embaralha(M)
 3 k ← 0
 4 while k ≤ n do
 5 |   m_b ← movimento(M, k)
 6 |   S' ← buscaLocal(S, m_b)
 7 |   if custo(S') < custo(S) then
 8 |   |   S ← S'
 9 |   |   k ← 0
10 |   else
11 |   |   k ← k + 1
12 |   end
13 end
14 return S
```

do algoritmo. Dessa forma, sempre que o RVND for chamado durante o IG, a ordem das vizinhanças a serem exploradas não será fixa.

Na linha 5, o k-ésimo movimento da lista M é selecionado. Na linha 6, a busca local correspondente ao movimento escolhido é aplicada à solução S, gerando uma nova solução S'. Se o custo de S' for menor que o custo da solução atual S, então S é atualizado para S', e k é reiniciado para 0, permitindo que todos os movimentos voltem a ser considerados. Caso contrário, k é incrementado em 1, removendo temporariamente o movimento recém-executado da lista de candidatos, para que outro seja explorado. O processo continua até que todos os movimentos tenham sido testados sem melhora na solução, ou seja, quando $k > n$. Nesse momento, o algoritmo termina e retorna a melhor solução encontrada.

5 EXPERIMENTOS COMPUTACIONAIS

Neste Capítulo, descrevem-se as instâncias utilizadas nos testes na Seção 5.1, detalhando suas características. Em seguida, na Seção 5.2, são especificados o ambiente computacional e os parâmetros adotados para a execução dos experimentos. Por fim, na Seção 5.3, são apresentados e discutidos os resultados obtidos, analisando a qualidade das soluções geradas e comparando o desempenho dos métodos MIG-RVND e MIG-QL com as abordagens MIP e MSTPH2, da literatura.

5.1 INSTÂNCIAS

O algoritmo foi executado para um conjunto de 81 instâncias, classificadas como "Clusterizadas - C" (24 instâncias), "Randomizadas - R" (33 instâncias) e "Randomizadas Clusterizadas - RC" (24 instâncias), retiradas de (47). As instâncias variam em grupos de 25, 50 e 100 clientes, com 27 instâncias cada. Em (15), é feito um tratamento sobre essas instâncias e, para que seja possível uma comparação entre os algoritmos desenvolvidos, o tratamento realizado neste trabalho é o mesmo.

Dessa forma, assume-se que 80% dos clientes podem ser atendidos por drones, sendo eles os com menor demanda. O custo de transporte por unidade é fixado em 25 para os caminhões e 1 para os drones, sendo a métrica utilizada para a distância calculada como a de Manhattan para os caminhões e Euclidiana para os drones. O tempo de espera máximo de drones é definido como 10 unidades de tempo para instâncias R e RC e 90 unidades de tempo para instâncias C. Em relação à capacidade de carga, o valor da instância original foi mantido.

Para todas as instâncias, considera-se a velocidade de caminhões como sendo de 1 unidade de distância por unidade de tempo, enquanto a velocidade de drones é o dobro da de caminhões. Além disso, o tempo de atendimento do cliente quando atendido por drone é a metade do tempo de atendimento quando este é atendido por caminhão. Por último, cada drone pode atender um único cliente após cada decolagem. A métrica utilizada para o deslocamento é a de Manhattan para os caminhões, simulando quarteirões e Euclidiana para os drones, seguindo caminhos por linhas diretas.

5.2 AMBIENTE DE TESTE E CONFIGURAÇÕES

O algoritmo proposto neste trabalho foi desenvolvidos na linguagem C++, com o compilador g++ (Ubuntu 11.3.0-1ubuntu1 22.04) 11.3.0. Os testes foram executados em uma máquina com processador AMD Ryzen 5 5500U, 8 GB de memória RAM com 3200 MHz, sistema operacional Ubuntu 22.04.1 LTS x86_64 com kernel Linux 5.15.0-58-generic. Cada uma das 81 instâncias foi executada 30 vezes e foram aferidos o tempo de execução

e o custo da solução encontrada.

Os parâmetros utilizados foram ajustados conforme testes realizados empiricamente junto com o *framework Iterated Race* (IRACE) de (48) tomando como entrada um conjunto de 18 instâncias representativas das instâncias de entrada. Os valores de α, β, i_{max}, k_{max}, Q_{init}, ϵ, η e γ calibrados pelo $framework$ considerando: $\alpha = \{0.1, 0.2, 0.3, \ldots, 0.9\}$, $\beta = \{0.1, 0.2, 0.3, \ldots, 0.9\}$, $Q_{init} = \{500, 600, 700, 900, 1000\}$, $\epsilon = \{0, 0.1, 0.2\}$, $\eta = \{0.1, 0.2, 0.3, \ldots, 0.9\}$, $\gamma = \{0.1, 0.2, 0.3, \ldots, 0.9\}$. Os resultados escolhidos pelo IRACE podem ser vistos na Tabela 1.

Tabela 1 – Parâmetros utilizados no experimento

i_{max}	k_{max}	α	β	Q_{init}	ϵ	η	γ
20	60	0.10	0.20	900	0.0	0.60	0.20

Fonte: Elaborada pelo autor. (2025).

5.3 RESULTADOS E ANÁLISE

Os resultados obtidos pelas abordagens propostas MIG-RVND e MIG-QL são apresentados nas tabelas e comparados quanto ao custo e tempo de processamento com a abordagem MSPTH2, apresentada em (16) e reimplementada por (44), e com o algoritmo MIP de (44). Em seguida, verifica-se o impacto do uso do multiplicador $Q(t, a)_{num}$ no cálculo da recompensa imediata Por fim, compara-se o MIG-QL com o MIG-RVND, ou seja, a eficiência do uso do Q-learning em relação ao uso do RVND no mesmo algoritmo.

Os parâmetros utilizados no MIP e no MSTPH2 foram os mesmos utilizados pelo autores, com o tempo normalizado em relação à máquina utilizada nos testes. Para o MIG-RVND e o MIG-QL foram utilizados os mesmos parâmetros α, β, i_{max} e k_{max} otimizados para o MIG-QL e executados na mesma máquina a efeito de comparação.

Nas Tabelas 2, 3 e 4, o MIG-RVND é o algoritmo MIG utilizando o RVND como algoritmo de refinamento no IG, o algoritmo MIG-QL é o mesmo MIG, utilizando o Q-Learning como técnica de tomada de decisão durante a etapa de refinamento da solução. As instâncias foram agrupadas por tamanho (25, 50 e 100 clientes) e ordenadas por tipo (C, R e RC). Foram realizadas 30 execuções para cada instância, onde todas as soluções são viáveis. A coluna “Melhor” mostra a melhor solução dentre todos os algoritmos comparados, em valor absoluto. A coluna “Melhor (%)” mostra o *gap* percentual, ou seja, a diferença percentual entre a melhor solução conhecida e a melhor solução encontrada pelo algoritmo. A coluna “Média (%)” mostra o *gap* percentual entre a melhor solução conhecida e o valor médio das soluções encontradas pelo algoritmo após 30 execuções. O “Tempo (s)” está o tempo médio em segundos para a execução do algoritmo. A linha “Média” mostra a média de cada coluna apresentada.

Tabela 2 – Comparação entre Algoritmos em instâncias de 25 clientes

		MIP			MSPTH2			MIG-RVND			MIG-QL		
Instância	Melhor Solução	Melhor (%)	Média (%)	Tempo (s)	Melhor (%)	Média (%)	Tempo (s)	Melhor (%)	Média (%)	Tempo (s)	Melhor (%)	Média (%)	Tempo (s)
C201_025	4.974,3	0,122	0,122	0,666	0,154	0,161	**0,093**	**0,000**	0,060	0,333	**0,000**	**0,052**	0,831
C202_025	4.783,0	0,158	0,176	0,791	0,090	0,143	**0,080**	**0,000**	**0,051**	0,505	**0,000**	0,066	1,056
C203_025	4.638,0	0,018	0,120	0,943	**0,000**	**0,020**	**0,198**	0,030	0,045	0,570	0,029	0,038	1,193
C204_025	3.966,0	0,007	0,044	1,025	**0,000**	**0,020**	**0,133**	0,123	0,135	0,637	0,123	0,136	1,277
C205_025	4.606,0	**0,000**	**0,042**	0,730	0,043	0,058	**0,109**	0,068	0,143	0,407	0,068	0,143	0,982
C206_025	4.388,0	**0,000**	**0,004**	0,756	0,005	0,038	**0,090**	0,127	0,190	0,386	0,123	0,212	0,875
C207_025	4.348,0	0,004	0,040	0,929	**0,000**	**0,031**	**0,091**	0,130	0,130	0,468	0,125	0,130	1,018
C208_025	4.361,0	0,001	**0,016**	0,829	**0,000**	**0,016**	**0,096**	0,121	0,196	0,380	0,121	0,218	0,892
R201_025	7.049,5	0,295	0,383	0,871	0,341	0,368	**0,172**	**0,000**	**0,025**	0,311	**0,000**	0,029	0,745
R202_025	7.308,0	**0,000**	0,055	1,017	0,015	**0,043**	**0,230**	0,111	0,111	0,463	0,111	0,111	1,057
R203_025	7.020,0	0,024	0,047	1,122	0,022	0,123	**0,223**	**0,000**	**0,009**	0,549	**0,000**	**0,009**	1,246
R204_025	6.281,0	**0,000**	**0,042**	1,275	0,063	0,159	**0,242**	0,173	0,269	0,682	0,172	0,271	1,405
R205_025	6.070,0	0,001	0,092	0,935	**0,000**	**0,053**	**0,161**	0,089	0,163	0,413	0,089	0,149	0,974
R206_025	5.948,0	**0,000**	**0,042**	1,017	0,019	0,104	**0,220**	0,217	0,235	0,550	0,152	0,234	1,255
R207_025	5.888,0	**0,000**	**0,060**	1,125	0,033	0,182	**0,182**	0,081	0,103	0,548	0,078	0,109	1,222
R208_025	5.331,0	**0,000**	**0,074**	1,230	0,168	0,309	**0,186**	0,327	0,373	0,686	0,311	0,374	1,438
R209_025	5.921,6	0,027	0,030	0,799	0,029	0,035	**0,100**	**0,000**	**0,017**	0,377	**0,000**	**0,017**	0,840
R210_025	6.545,8	0,138	0,184	0,987	0,075	0,175	**0,279**	**0,000**	**0,000**	0,479	**0,000**	**0,000**	1,101
R211_025	5.079,0	**0,000**	**0,035**	0,900	0,072	0,199	**0,133**	0,045	0,070	0,489	0,045	0,083	1,151
RC201_025	8.169,0	0,009	0,042	0,804	**0,000**	**0,023**	**0,192**	0,055	0,055	0,270	0,055	0,055	0,669
RC202_025	6.664,0	0,044	0,124	0,962	**0,000**	**0,074**	**0,157**	0,253	0,254	0,354	0,253	0,254	0,822
RC203_025	6.682,0	0,080	**0,148**	0,977	**0,000**	0,152	**0,222**	0,208	0,208	0,470	0,208	0,208	1,044
RC204_025	6.250,0	**0,000**	**0,010**	1,025	0,001	0,052	**0,200**	0,034	0,034	0,491	0,034	0,034	1,023
RC205_025	7.963,0	**0,000**	**0,009**	0,809	0,014	0,041	**0,125**	0,052	0,052	0,368	0,052	0,052	0,873
RC206_025	6.539,0	**0,000**	**0,050**	0,860	0,011	0,118	**0,194**	0,222	0,239	0,344	0,222	0,233	0,849
RC207_025	6.411,0	**0,000**	**0,030**	0,877	0,032	0,091	**0,166**	0,040	0,041	0,366	0,040	0,044	0,826
RC208_025	5.603,0	0,026	**0,038**	0,834	**0,000**	0,040	**0,115**	0,040	0,104	0,419	0,042	0,103	0,909
Média	5.881,0	**0,035**	**0,076**	0,929	0,044	0,105	**0,163**	0,094	0,123	0,456	0,091	0,125	1,021

Fonte: Elaborado pelo autor. (2025).

Na Tabela 2, nota-se uma dificuldade do algoritmo em resolver as instâncias com 25 clientes. Tanto o MIG-QL quanto o MIG-RVND possuem uma performance pior relacionado aos algoritmos MIP e MSPTH2 da literatura, apesar do algoritmo MIG conseguir resultados melhores em custo em 6 instâncias, sendo 2 do tipo C e 4 do tipo R.

Na Tabela 3, observa-se que o tempo médio de execução do MIG-RVND foi o menor da literatura em 11 instâncias, sendo que 7 foram nas instâncias de tipo C. Além disso, os algoritmos MIG conseguiram os melhores resultados em 9 instâncias para a variação com RVND e em 8 instâncias para a variação MIG-QL. Em relação ao custo médio, ambas as variações do MIG apresentaram a menor diferença percentual geral.

Na Tabela 4, os algoritmos MIG encontraram soluções melhores do que o MIP e o MSPTH2 em média, ao mesmo tempo que aumentou a eficiência média em tempo, mostrando a escalabilidade do algoritmo. Tanto o MIG-RVND quanto o MIG-QL se mostram eficientes à medida que se aumenta o número de clientes. Ao se calcular a razão entre o tempo médio para instâncias de 50 clientes e de 100 clientes (dobrando-se o número de clientes), tem-se que tempo de execução para o MIP aumentou cerca de 10.272 vezes, o MSTPH2 aumentou cerca de 12.118, o MIG-RVND aumentou cerca de 3.238 e o MIG-QL cerca de 2.850. Assim, sugere-se que os algoritmos MIG sejam os que possuem melhor escalabilidade dentre os apresentados para as instâncias consideradas. Em relação aos resultados, o MIG mostrou uma média de resultados melhor do que o MIP e o MSTPH2. Percebe-se a eficiência dos algoritmos para instâncias do tipo RC, onde encontraram as

Tabela 3 – Comparação entre Algoritmos em instâncias de 50 clientes

		MIP			MSPTH2			MIG-RVND			MIG-QL		
	Melhor	Melhor	Média	Tempo	Melhor	Média	Tempo	Melhor	Média	Tempo	Melhor	Média	Tempo
Instância	**Solução**	(%)	(%)	(s)	(%)	(%)	(s)	(%)	(%)	(s)	(%)	(%)	(s)
C201_050	7.952,3	0,159	0,200	2,811	0,148	0,178	2,090	0,004	0,022	**1,794**	**0,000**	**0,020**	3,609
C202_050	7.560,2	0,118	0,179	3,559	0,149	0,187	2,531	**0,000**	0,025	**2,116**	0,012	**0,024**	3,835
C203_050	7.700,3	0,042	0,110	4,754	0,102	0,145	2,856	**0,000**	0,027	**2,235**	0,007	**0,026**	3,976
C204_050	7.379,0	**0,000**	**0,085**	7,292	0,017	0,101	**2,513**	0,082	0,087	2,849	0,072	0,086	4,384
C205_050	7.742,5	0,088	0,149	3,081	0,059	0,063	2,146	0,005	0,038	**1,889**	**0,000**	**0,028**	3,615
C206_050	7.722,5	0,011	0,062	3,673	0,037	0,043	2,321	**0,000**	**0,041**	**1,783**	0,006	**0,041**	3,411
C207_050	7.641,0	**0,000**	0,080	4,544	0,057	0,086	2,125	0,004	0,054	**2,066**	0,011	**0,051**	3,697
C208_050	7.755,5	0,010	0,053	4,111	0,032	0,042	3,036	**0,000**	**0,036**	**1,883**	**0,000**	0,039	3,520
R201_050	10.786,5	0,242	0,298	6,064	0,186	0,262	**1,177**	**0,000**	**0,009**	1,691	**0,000**	**0,009**	3,954
R202_050	10.428,0	0,117	0,158	4,624	0,016	0,145	**1,609**	**0,000**	**0,031**	2,161	**0,000**	0,033	4,491
R203_050	9.364,0	0,134	0,213	5,918	**0,000**	0,110	2,012	0,009	**0,081**	2,682	0,014	0,107	5,069
R204_050	8.149,7	0,037	0,092	6,260	0,054	0,141	**1,580**	0,010	0,100	2,988	**0,000**	**0,077**	5,579
R205_050	10.302,0	**0,000**	0,045	3,790	0,007	0,035	**0,910**	0,007	**0,023**	2,164	0,008	0,029	4,401
R206_050	9.222,0	0,020	0,109	4,850	**0,000**	**0,078**	**1,394**	0,077	0,102	2,207	0,073	0,093	4,702
R207_050	8.634,0	0,070	0,140	5,574	**0,000**	**0,101**	**1,721**	0,053	0,128	2,739	0,052	0,127	5,259
R208_050	8.067,0	0,053	0,089	6,054	**0,000**	**0,049**	**1,605**	0,008	0,100	2,946	0,010	0,094	5,475
R209_050	9.810,0	0,049	0,102	4,176	**0,000**	**0,037**	**1,390**	0,039	0,074	2,060	0,038	0,073	4,129
R210_050	8.834,0	0,168	0,222	4,433	**0,000**	0,128	**1,715**	0,014	0,113	2,201	0,043	**0,105**	4,581
R211_050	7.899,0	0,035	0,063	4,313	**0,000**	**0,045**	**0,794**	0,014	0,073	2,378	0,013	0,065	4,878
RC201_050	14.294,2	0,025	0,088	3,793	0,001	0,091	1,756	**0,000**	0,004	**1,587**	**0,000**	**0,003**	3,641
RC202_050	13.226,0	0,038	0,064	4,005	**0,000**	0,078	2,642	0,044	**0,060**	**1,889**	0,047	0,061	4,200
RC203_050	12.772,0	0,014	**0,028**	4,469	**0,000**	0,053	2,744	0,086	0,121	**2,325**	0,086	0,129	4,587
RC204_050	10.520,0	0,033	0,072	4,915	0,076	0,137	**1,414**	**0,000**	**0,064**	2,667	0,010	0,069	5,019
RC205_050	13.219,0	**0,000**	**0,021**	3,677	0,017	0,064	2,170	0,048	0,056	**1,975**	0,048	0,056	4,168
RC206_050	12.434,0	0,035	0,076	3,769	**0,000**	0,131	**1,829**	0,055	0,064	1,962	0,055	**0,061**	3,938
RC207_050	12.011,0	0,014	0,082	3,708	**0,000**	0,154	**1,890**	0,027	**0,071**	1,949	0,028	0,076	4,099
RC208_050	10.310,7	0,007	0,077	3,553	0,030	0,087	**1,486**	**0,000**	**0,007**	2,091	**0,000**	0,012	4,110
Média	9.693,9	0,056	0,110	4,510	0,037	0,103	**1,906**	**0,022**	0,060	2,195	0,023	**0,059**	4,308

Fonte: Elaborado pelo autor. (2025).

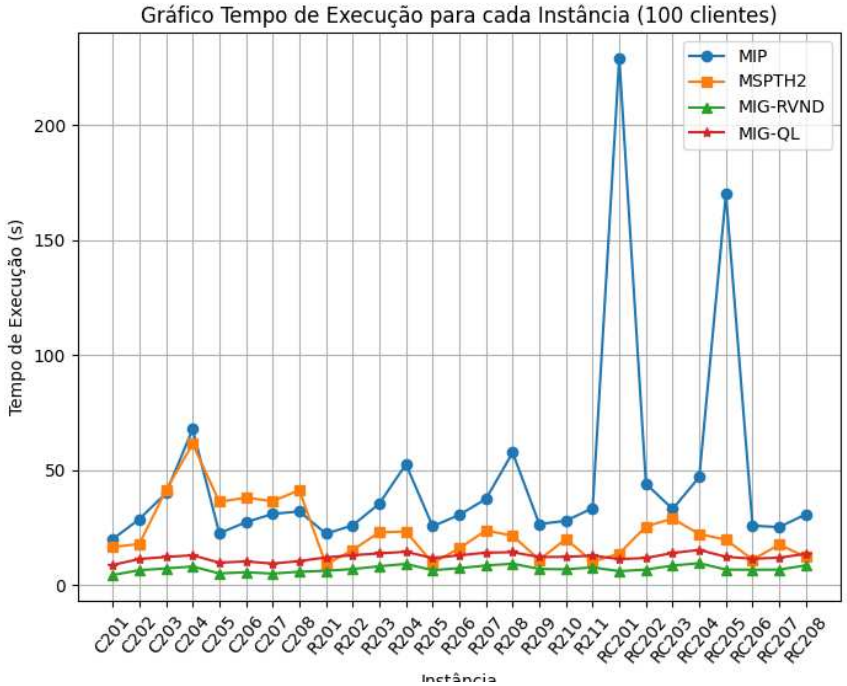
Tabela 4 – Comparação entre Algoritmos em instâncias de 100 clientes

		MIP			MSPTH2			MIG-RVND			MIG-QL		
	Melhor	Melhor	Média	Tempo	Melhor	Média	Tempo	Melhor	Média	Tempo	Melhor	Média	Tempo
Instância	**Solução**	(%)	(%)	(s)	(%)	(%)	(s)	(%)	(%)	(s)	(%)	(%)	(s)
C201_100	13.707,0	0,120	0,301	19,988	**0,000**	**0,012**	16,601	0,058	0,060	**4,429**	0,057	0,060	8,789
C202_100	13.614,0	0,054	0,232	28,715	**0,000**	**0,012**	17,904	0,031	0,044	**6,502**	0,032	0,047	11,373
C203_100	14.122,0	**0,000**	0,128	40,012	0,083	0,136	41,388	0,008	**0,040**	**7,290**	0,023	0,042	12,261
C204_100	14.368,7	0,010	0,078	67,851	0,135	0,222	61,433	**0,000**	0,021	**8,107**	0,005	**0,020**	13,009
C205_100	13.361,0	0,034	0,106	22,564	**0,000**	**0,051**	36,288	0,082	0,084	**5,113**	0,082	0,084	9,797
C206_100	13.032,0	0,060	0,132	27,401	**0,000**	**0,045**	38,027	0,082	0,121	**5,621**	0,063	0,123	10,318
C207_100	13.185,0	0,080	0,191	30,993	**0,000**	**0,078**	36,529	0,098	0,114	**5,038**	0,101	0,112	9,387
C208_100	13.010,0	0,044	0,188	32,083	**0,000**	**0,115**	41,318	0,114	0,131	**5,834**	0,113	0,136	10,489
R201_100	15.130,5	0,093	0,145	22,397	0,075	0,142	8,979	**0,000**	0,063	**6,261**	0,011	**0,061**	12,066
R202_100	13.449,9	0,119	0,226	25,977	0,104	0,184	15,301	**0,000**	0,089	**7,000**	0,038	**0,087**	13,028
R203_100	13.096,5	0,168	0,266	35,473	0,114	0,187	22,975	**0,000**	**0,078**	**8,212**	0,049	**0,078**	13,828
R204_100	11.139,0	0,161	0,259	52,583	**0,000**	**0,058**	23,342	0,090	0,136	**9,233**	0,051	0,136	14,524
R205_100	14.066,0	0,060	0,127	25,608	**0,000**	**0,014**	9,594	0,033	0,057	**6,502**	0,031	0,052	11,784
R206_100	13.645,9	0,007	0,126	30,564	0,032	0,099	16,048	0,003	**0,030**	**7,399**	**0,000**	**0,030**	13,009
R207_100	12.240,0	0,158	0,226	37,496	**0,000**	0,131	23,801	0,044	**0,103**	**8,551**	0,083	0,114	14,120
R208_100	10.652,0	0,238	0,314	57,682	**0,000**	**0,021**	21,504	0,108	0,183	**9,326**	0,111	0,183	14,346
R209_100	14.201,0	0,046	0,095	26,569	**0,000**	**0,017**	11,077	0,019	0,053	**7,061**	0,009	0,058	12,239
R210_100	12.871,5	0,134	0,209	28,001	0,048	0,106	20,092	**0,000**	0,075	**6,903**	0,041	**0,074**	12,281
R211_100	11.530,0	0,127	0,191	33,495	**0,000**	**0,011**	10,190	0,070	0,086	**7,732**	0,054	0,086	12,895
RC201_100	19.859,4	0,256	0,368	229,104	0,124	0,237	13,755	0,003	**0,034**	**6,093**	**0,000**	0,036	11,320
RC202_100	16.562,4	0,238	0,378	44,108	0,184	0,301	25,528	0,001	**0,045**	**6,785**	**0,000**	**0,045**	11,834
RC203_100	16.177,7	0,160	0,226	33,111	0,181	0,245	28,970	**0,000**	**0,046**	**8,522**	0,015	0,058	13,980
RC204_100	14.668,7	0,169	0,223	47,009	0,138	0,263	22,204	**0,000**	**0,008**	**9,560**	**0,000**	0,011	15,337
RC205_100	19.301,3	0,237	0,314	170,022	0,187	0,244	19,796	0,015	0,094	**6,761**	**0,000**	**0,075**	12,327
RC206_100	17.865,1	0,145	0,232	25,893	0,171	0,312	11,100	**0,000**	**0,053**	**6,702**	0,021	0,062	11,513
RC207_100	16.824,7	0,073	0,151	25,231	0,141	0,252	17,657	0,002	**0,070**	**6,753**	**0,000**	0,082	11,937
RC208_100	14.519,9	0,203	0,278	30,865	0,208	0,271	12,220	0,018	0,063	**8,633**	**0,000**	**0,057**	13,662
Média	14.303,7	0,118	0,211	46,326	0,071	0,140	23,097	**0,033**	**0,073**	**7,108**	0,037	0,074	12,276

Fonte: Elaborado pelo autor. (2025).

melhores soluções em todas as 8 instâncias, sendo 6 encontradas pelo MIG-QL as outras duas pelo MIG-RVND, com empate para a instância RC204_100.

Figura 11 - Comparativo de tempo de execução entre os algoritmos para cada instância de 100 clientes



Fonte: Elaborado pelo autor. (2025).

Para evidenciar a escalabilidade dos algoritmos MIG, foi feito um gráfico comparativo entre os tempos médios de execução para cada instância de 100 clientes, onde a quantidade de clientes permite que seja observado uma diferença acentuada entre esses tempos. Na Figura 11, evidencia-se o tempo de execução entre os diferentes algoritmos para cada instância de 100 clientes. Observa-se como os algoritmos MIG conseguiram tempos de execução menores que os apresentados na literatura. Destaca-se o MIG-RVND, que conseguiu os menores tempos de execução para todas as instâncias. O MIG-QL teve uma performance um pouco abaixo em relação à variação com o RVND e, para 5 instâncias, teve tempo de execução minimamente maior do que o MSPTH2. É importante ressaltar que mesmo com tempo de execução menor, os resultados se mostraram competitivos em relação ao da literatura, conseguindo melhores resultados em 8 instâncias de 100 clientes, para o MIG-RVND e em 7 instâncias para o MIG-QL, conforme Tabela 4.

O MIG-QL foi executado em instâncias selecionadas considerando o multiplicador $Q(t,a)_{num}$ no cálculo da recompensa $R(t,a)$, visto na Seção **4.5.1**. Para essas mesmas instâncias, o MIG-QL foi executado sem esse fator no cálculo de $R(t,a)$, ou seja, $R(t,a) = custo(S) - custo(S^*)$. A ideia do uso é de valorizar iterações avançadas quando a recompensa tende a diminuir.

Para essa comparação, foram selecionadas 21 instâncias de maneira aleatória, sendo

nove do tipo C: três com 25 clientes, três com 50 clientes e três com 100 clientes, nove do tipo R: três com 25 clientes, três com 50 clientes e três com 100 clientes e nove do tipo RC: três com 25 clientes, três com 50 clientes e três com 100 clientes. Os resultados podem ser vistos na Tabela 5, onde as instâncias são agrupadas por tipo e calculadas a média do grupo.

Tabela 5 – Comparação entre o algoritmo MIG-IG com e sem multiplicador

		MIG-QL Multiplicador			**MIG-QL sem Multiplicador**		
Tipo de Instância	**Melhor Solução (Média)**	**Melhor (%)**	**Média (%)**	**Tempo (s)**	**Melhor (%)**	**Média (%)**	**Tempo (s)**
25 clientes	5989,144	**0,004**	**0,042**	1,114	0,008	0,044	**1,016**
50 clientes	10426,767	**0,001**	**0,035**	4,174	**0,001**	**0,035**	**3,795**
100 clientes	14549,611	0,008	**0,051**	12,336	**0,006**	0,053	**11,503**
Tipo C	8927,320	0,006	**0,031**	5,254	**0,001**	**0,031**	**4,921**
Tipo R	9881,246	0,007	**0,061**	6,433	**0,005**	**0,061**	**5,969**
Tipo RC	12156,957	**0,000**	**0,037**	5,937	0,009	0,040	**5,424**

Fonte: Elaborado pelo autor. (2025).

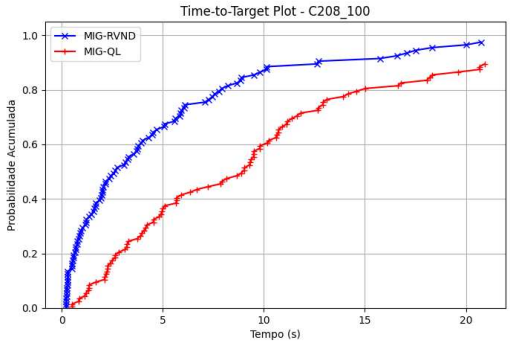
Conforme visto na Tabela 5, usar o multiplicador $Q(t, a)_{num}$ contribuiu positivamente para a média das soluções encontradas. Ou seja, em média, as soluções encontradas pelo MIG-QL usando o multiplicador são melhores do que sem usar. Apesar disso, tem-se uma pequena perda de desempenho, evidenciada na coluna "Tempo (s)". Nota-se também que para todas as instâncias RC, o MIG-QL com multiplicador encontrou soluções melhores do que sem.

A fim de comparar o algoritmo MIG-QL com o MIG-RVND, foi feito um gráfico *Time-to-target* (TTT) (49, 50). O gráfico TTT mostra no eixo das ordenadas a probabilidade de se encontrar uma solução que seja tão boa quando uma solução alvo pré-definida dado um tempo de execução. Esse gráfico permite a comparação entre diferentes algoritmos de otimização.

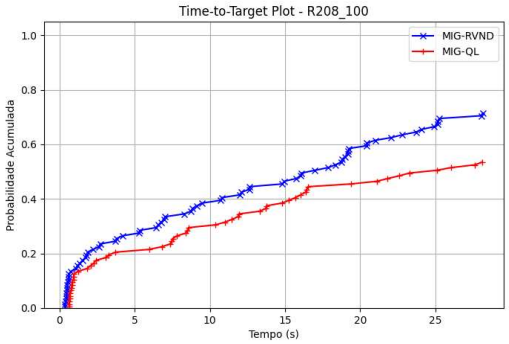
A solução alvo a ser atingida pelos algoritmos foi definida como a solução média encontrada pelo MIG-QL. O tempo limite foi definido como o dobro do tempo médio de execução para o MIG-QL, mostrado nas Tabelas 2, 3 e 4. Cada algoritmo foi executado 100 vezes, e os resultados foram exibidos nos gráficos TTT. Foram testados para 21 instâncias selecionadas aleatoriamente, sendo 3 do tipo C, R e RC e 3 para 25, 50 e 100 clientes. Nos gráficos 12, 13 e 14, são exibidas três instâncias específicas para cada tamanho de problema, apresentadas em gráficos Time-to-Target (TTT). O valor da solução alvo está indicado na legenda de cada subfigura correspondente. As instâncias apresentadas foram selecionadas para representar as principais categorias do conjunto de dados, contemplando uma instância do tipo C, uma do tipo R e uma do tipo RC.

No gráfico 12a, observa-se que o MIG-RVND tem desempenho superior ao MIG-QL, evidenciada pela curva azul estar acima da curva vermelha, ou seja, a todo instante de

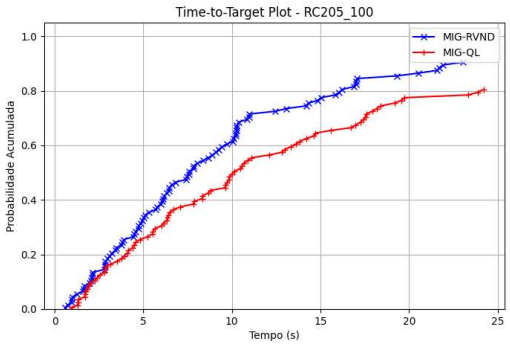
Figura 12 - Gráficos Time-to-Target para três instâncias de 100 clientes



(a) Instância C208_100, Alvo: 14776.863



(b) Instância R208_100, Alvo: 12602.29



(c) Instância RC205_100, Alvo: 20742.107

Fonte: Elaborado pelo autor. (2025).

tempo, a probabilidade de se encontrar a solução alvo é maior para o MIG-RVND do que para o MIG-QL.

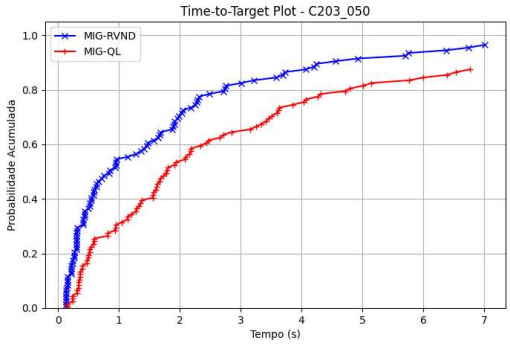
No gráfico 12b, nota-se dificuldade dos dois algoritmos para encontrar a solução alvo dentro do tempo limite, evidenciado por nenhum dos dois conseguirem sucesso em pelo menos 80% das execuções. Apesar disso, nota-se que o MIG-RVND tem mais eficiência para essa instância do que o MIG-QL.

No gráfico 12c, observa-se a similaridade de eficiência dos algoritmos até aproximadamente 5 segundos. Após esse tempo, o MIG-RVND apresentou clara vantagem em relação ao MIG-QL.

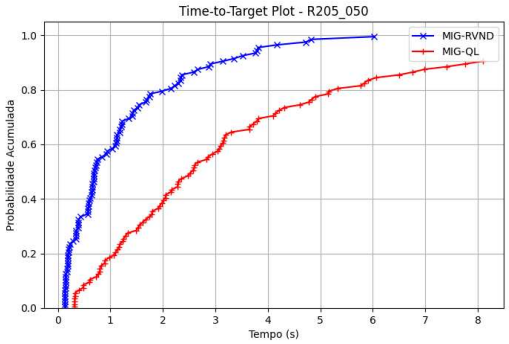
Nos gráficos 13a, 13b e 13c observa-se vantagem do algoritmo MIG-RVND sobre o MIG-QL, onde para qualquer instante de tempo, a probabilidade acumulada de se alcançar a solução alvo é maior para o MIG-RVND, na curva azul, do que para o MIG-QL, na curva vermelha. Fica evidenciado também que na instância mostrada no gráfico 13b o MIG-RVND conseguiu 100% de sucesso em aproximadamente 6 segundos, ao passo que o MIG-QL não conseguiu em mais de 8 segundos.

Nos gráficos 14a, 14b e 14c, de 25 clientes, também é possível perceber a eficiência do

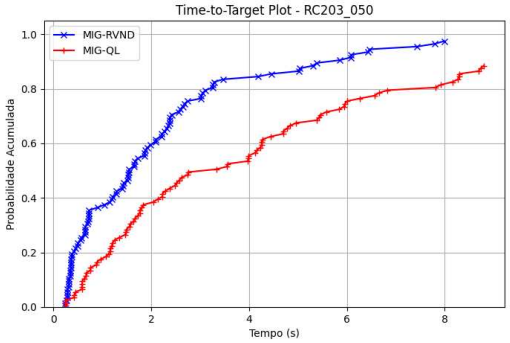
Figura 13 - Gráficos Time-to-Target para três instâncias de 50 clientes



(a) Instância C203_050, Alvo: 7902.847



(b) Instância R205_050, Alvo: 10603.097



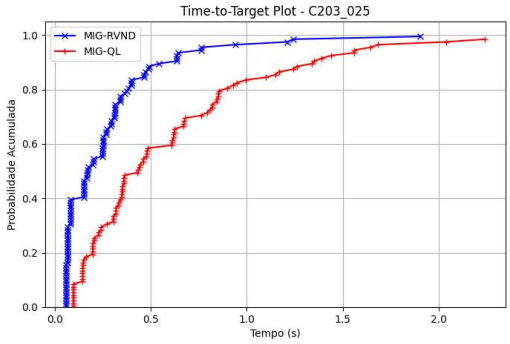
(c) Instância RC203_050, Alvo: 14419.887

Fonte: Elaborado pelo autor. (2025).

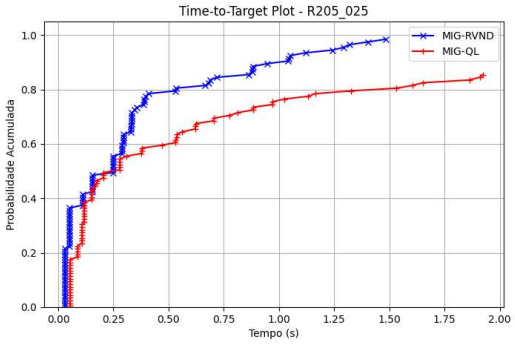
MIG-RVND sobre o MIG-QL. Apenas na instância de tipo R, do gráfico 14b o desempenho do MIG-QL foi próximo ao apresentado pelo MIG-RVND até aproximadamente 0.25 segundos. Fora isso, ambos os algoritmos conseguiram aproximadamente 100% de sucesso em diferenças de aproximadamente 0.5 segundos entre eles para as outras duas instâncias, nos gráficos 14a e 14c.

Após análises dos gráficos, observa-se que o MIG-RVND se mostra superior em relação ao MIG-QL, evidenciado pela curva do MIG-RVND no gráfico TTT estar acima da curva apresentada pelo MIG-QL em todas as instâncias analisadas.

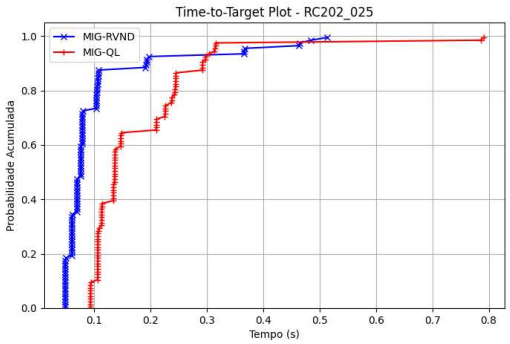
Figura 14 - Gráficos Time-to-Target para três instâncias de 25 clientes



(a) Instância C203_025, Alvo: 4813.4



(b) Instância R205_025, Alvo: 6973.803



(c) Instância RC202_025, Alvo: 8353.785

Fonte: Elaborado pelo autor. (2025).

6 CONCLUSÃO

Para solucionar o problema de roteamento de veículos capacitados com drones e janela de tempo (VRP-DTW), este trabalho propôs um *Multi-Start Iterated Greedy* (MIG). Foram desenvolvidos dois algoritmos de refinamento diferentes, um utilizando um método Q-Learning (MIG-QL) e o outro utilizando um método Random Variable Neighborhood Descent (MIG-RVND). A partir de uma solução gerada pelo algoritmo guloso randomizado, o MIG aplica etapas de remoção parcial de clientes para depois reinserí-los, potencialmente melhorando a solução anterior. A partir dessa solução, o Q-Learning busca escolher as melhores combinações de movimentos de forma a melhorar a qualidade da solução de forma inteligente, através do Aprendizado por Reforço. Alternativamente, pelo método RVND, atua de forma similar ao Q-learning, exceto que a escolha dos movimentos é feito de forma aleatória.

Assim, o algoritmo proposto consiste numa etapa *Multi-Start*, onde a cada iteração é definido o número de rotas iniciais para geração da solução. Com isso, a solução é criada passo a passo através de um algoritmo guloso randomizado, onde na primeira etapa clientes que só podem ser atendidos por caminhão são distribuídos pelas rotas e na segunda etapa os clientes restantes são atendidos por drones ou caminhões. Após essa etapa, a solução entra num método *Iterated Greedy* (IG), onde a cada iteração há uma destruição parcial das rotas, uma reconstrução pelo algoritmo guloso randomizado e, por fim, um refinamento, na qual a solução passa por pequenas alterações através de buscas locais de forma a diminuir o custo gradativamente.

Foram implementadas nove buscas locais, voltadas ao refinamento da solução por meio de alterações locais em sua estrutura. Os movimentos incluem trocas de clientes dentro de uma mesma rota de caminhão (*intra*-rota), bem como operações baseadas na remoção de um cliente e sua reinserção por meio de caminhão ou drone, tanto no interior de uma mesma rota quanto entre rotas distintas (*inter*-rota).

O algoritmo implementado foi testado para um conjunto de 81 instâncias, variando em quantidade e tipo. São 27 instâncias de 25, 50 e 100 clientes cada, 24 instâncias do tipo clusterizada, 33 do tipo randomizada e 24 do tipo randomizada e clusterizada. Os resultados dos algoritmos MIG foram comparados com os resultados obtidos pelo MSPTH2 e pelo MIP, de (44). Também foi feito uma breve análise sobre o uso de um multiplicador no cálculo da recompensa imediata no Q-learning, que tem como objetivo valorizar ações com maior recorrência. Por fim, foi feito um comparativo entre o uso do Q-Learning e o RVND.

O algoritmo proposto obteve resultados competitivos com o da literatura, sobretudo para instâncias de 50 e 100 clientes. Em média os algoritmos MIG obtiveram resultados melhores do que o MIP e o MSTPH2, em tempos de 2 a 4 vezes menores. O algoritmo

também se mostrou escalável, onde em um aumento de 50 para 100 clientes o tempo médio de execução aumentou 2.8 vezes para o MIG-QL, e 3.2 para o MIG-RVND, ao passo que para os outros algoritmos o tempo aumentou 10.2 e 12.1 vezes, respectivamente. Dessa forma, o MIG se mostrou eficiente para instâncias grandes, de 100 clientes.

O uso do multiplicador influenciou no tempo de execução, onde o seu uso aumenta o tempo de execução em aproximadamente 0.4 segundos para alguns grupos de instâncias. Em relação ao comparativo do MIG-RVND com o MIG-QL, o MIG-QL da forma com a qual está implementado não se mostrou mais eficiente, conforme evidenciado pelos gráficos *time-to-target*.

O modelo MIG mostra-se uma técnica eficiente para tratar o VRP-DTW. Em relação ao refinamento, o RVND conseguiu resultados melhores do que o Q-Learning. Portanto, como trabalhos futuros, sugere-se utilizar outros tipos de recompensa imediata por utilizar certa ação, bem como propor outras políticas de escolha desta ação, além do ϵ-greedy. Sugere-se também avaliar a solução atual como um estado no Q-learning, ao invés de ser a ação tomada no passado. Também recomenda-se a implementação de outros métodos de aprendizado por reforço como por exemplo o *Double Learning Q-Learning* que implementa duas matrizes $Q(t, a)$, com o objetivo de evitar o viés de maximização presente no Q-Learning, que decide qual ação é a melhor a ser tomada pela recompensa máxima esperada (40).

Em relação ao problema, sugere-se estudar como o MIG atua em outras variações do VRP-DTW, como o VRP com crowd-shipping, apresentado por (19), ou o VRP-DTW com frota de caminhões heterogênea, onde há caminhões a combustão e elétricos. Também pode-se considerar o problema num campo tri-dimensional, onde as casas e prédios influenciam no trajeto dos drones, entre outras variações.

REFERÊNCIAS

1 PAULO Coelho: 'I had an enormous amount of fun being a hippy'. <https://www.theguardian.com/books/2014/jan/25/paulo-coelho-this-much-i-know>. Acesso em: 2025-03-05.

2 WONG, R. T. Vehicle routing for small package delivery and pickup services. In: _____. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Boston, MA: Springer US, 2008. p. 475–485. ISBN 978-0-387-77778-8. Disponível em: <https://doi.org/10.1007/978-0-387-77778-8_21>.

3 IRNICH, S.; TOTH, P.; VIGO, D. Chapter 1: The family of vehicle routing problems. In: _____. *Vehicle Routing*. [s.n.]. p. 1–33. Disponível em: <https://epubs.siam.org/doi/abs/10.1137/1.9781611973594.ch1>.

4 NEWS, B. *Amazon testing drones for deliveries*. 2013. Acessado em 23 de fevereiro de 2025. Disponível em: <https://www.bbc.com/news/technology-25180906>.

5 POPPER, B. *Drones could make Amazon's dream of free delivery profitable*. 2015. Acessado em 23 de fevereiro de 2025. Disponível em: <https://www.theverge.com/2015/6/3/8719659/amazon-prime-air-drone-delivery-profit-free-shipping-small-items>.

6 BUTLER, S.; SWENEY, M. *Amazon asks permission to launch drones in north-east of England*. 2025. Acessado em 23 de fevereiro de 2025. Disponível em: <https://www.theguardian.com/technology/2025/jan/28/amazon-drones-england-darlington-county-durham>.

7 MöLLER, P. *Future of Freight: The Use of Drones in Logistics*. 2024. Acessado em 23 de fevereiro de 2025. Disponível em: <https://dhl-freight-connections.com/en/solutions/future-of-freight-the-use-of-drones-in-logistics/>.

8 BERMINGHAM, F. *FedEx Researching Drone Delivery But Not For Widespread Use*. 2014. Acessado em 23 de fevereiro de 2025. Disponível em: <https://www.ibtimes.co.uk/fedex-researching-drone-delivery-not-widespread-use-1471063>.

9 ANAC, A. de Comunicação da. *ANAC concede a primeira autorização para entregas comerciais com drone*. 2022. Acessado em 23 de fevereiro de 2025. Disponível em: <https://www.gov.br/anac/pt-br/noticias/2022/anac-concede-a-primeira-autorizacao-para-entregas-comerciais-com-drone>.

10 WANG, Z.; SHEU, J.-B. Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, v. 122, p. 350–364, 2019. ISSN 0191-2615.

11 WANG, X.; POIKONEN, S.; GOLDEN, B. The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, v. 11, n. 4, p. 679–697, Apr 2017. ISSN 1862-4480.

12 MURRAY, C. C.; CHU, A. G. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, v. 54, p. 86–109, 2015. ISSN 0968-090X.

13 CHIANG, W.-C. et al. Impact of drone delivery on sustainability and cost: Realizing the uav potential through vehicle routing optimization. *Applied Energy*, v. 242, p. 1164–1175, 2019. ISSN 0306-2619.

14 RITCHIE, H. Cars, planes, trains: where do co emissions from transport come from? *Our World in Data*, 2020. Https://ourworldindata.org/co2-emissions-from-transport.

15 FARIA, E. B. R. et al. Um algoritmo híbrido para o problema de roteamento de caminhões e drones com janela de tempo. In: *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*. Juiz de Fora: [s.n.], 2022. v. 54.

16 PUGLIESE, L. D. P.; MACRINA, G.; GUERRIERO, F. Trucks and drones cooperation in the last-mile delivery process. *Networks*, v. 78, n. 4, p. 371–399, 2021.

17 Queiroz dos Santos, J. P. et al. Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. *Expert Systems with Applications*, v. 41, n. 10, p. 4939–4949, 2014. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417414000645>.

18 ALICASTRO, M. et al. A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems. *Computers Operations Research*, v. 131, p. 105272, 2021. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054821000642>.

19 PUGLIESE, L. D. P. et al. Combining variable neighborhood search and machine learning to solve the vehicle routing problem with crowd-shipping. *Optimization Letters*, v. 17, n. 9, p. 1981–2003, Dec 2023. ISSN 1862-4480. Disponível em: <https://doi.org/10.1007/s11590-021-01833-x>.

20 LEI, D.; CUI, Z.; LI, M. A dynamical artificial bee colony for vehicle routing problem with drones. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 107, p. 104510, 2022.

21 SACRAMENTO, D.; PISINGER, D.; ROPKE, S. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, v. 102, p. 289–315, 2019. ISSN 0968-090X.

22 EUCHI, J.; SADOK, A. Hybrid genetic-sweep algorithm to solve the vehicle routing problem with drones. *Physical Communication*, v. 44, p. 101236, 2021. ISSN 1874-4907.

23 SCHERMER, D.; MOEINI, M.; WENDT, O. A variable neighborhood search algorithm for solving the vehicle routing problem with drones. *Technacal Report*, 2018.

24 DAKNAMA, R.; KRAUS, E. *Vehicle Routing with Drones*. 2017.

25 TOTH, P.; VIGO, D. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002. Disponível em: <https://epubs.siam.org/doi/abs/10.1137/1.9780898718515>.

26 BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 35, n. 3, p. 268–308, set. 2003. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/937503.937505>.

27 TALPUR, K. et al. Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, v. 52, 12 2019.

28 BIANCHI, L. et al. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, v. 8, n. 2, p. 239–287, Jun 2009. ISSN 1572-9796. Disponível em: <https://doi.org/10.1007/s11047-008-9098-4>.

29 TAILLARD, É. D. Local search. In: _____. *Design of Heuristic Algorithms for Hard Optimization: With Python Codes for the Travelling Salesman Problem*. Cham: Springer International Publishing, 2023. p. 103–129. ISBN 978-3-031-13714-3. Disponível em: <https://doi.org/10.1007/978-3-031-13714-3_5>.

30 HANSEN, P.; MLADENOVIć, N. First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, v. 154, n. 5, p. 802–817, 2006. ISSN 0166-218X. IV ALIO/EURO Workshop on Applied Combinatorial Optimization. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0166218X05003070>.

31 JACOBS, L. W.; BRUSCO, M. J. Note: A local-search heuristic for large set-covering problems. *Naval Research Logistics (NRL)*, v. 42, n. 7, p. 1129–1140, 1995. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/1520-6750%28199510%2942%3A7%3C1129%3A%3AAID-NAV3220420711%3E3.0.CO%3B2-M>.

32 RUIZ, R.; STüTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, v. 44, p. 2033–2049, 03 2007.

33 ZHAO, Z.; ZHOU, M.; LIU, S. Iterated greedy algorithms for flow-shop scheduling problems: A tutorial. *IEEE Transactions on Automation Science and Engineering*, v. 19, n. 3, p. 1941–1959, 2022.

34 STÜTZLE, T.; RUIZ, R. Iterated greedy. In: _____. *Handbook of Heuristics*. Cham: Springer International Publishing, 2018. p. 547–577. ISBN 978-3-319-07124-4. Disponível em: <https://doi.org/10.1007/978-3-319-07124-4_10>.

35 BRIMBERG, J. et al. Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem. *Operations Research*, v. 48, n. 3, p. 444–460, 2000. Disponível em: <https://doi.org/10.1287/opre.48.3.444.12431>.

36 MLADENOVIć, N.; HANSEN, P. Variable neighborhood search. *Computers Operations Research*, v. 24, n. 11, p. 1097–1100, 1997. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054897000312>.

37 PéREZ, J. A. M.; HANSEN, P.; MLADENOVIć, N. Parallel variable neighborhood search. In: _____. *Parallel Metaheuristics*. John Wiley Sons, Ltd, 2005. cap. 11, p. 247–266. ISBN 9780471739388. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/0471739383.ch11>.

38 LEAL, J.; JUNIOR, O. S. A multiple ant colony system with random variable neighborhood descent for the vehicle routing problem with time windows. *International Journal of Logistics Systems and Management*, v. 1, p. 1, 01 2020.

39 DUARTE, A. et al. Variable neighborhood descent. In: _____. *Handbook of Heuristics*. Cham: Springer International Publishing, 2018. p. 341–367. ISBN 978-3-319-07124-4. Disponível em: <https://doi.org/10.1007/978-3-319-07124-4_9>.

40 SUTTON, R. S.; BARTO, A. G. *Reinforcement learning, Second edition: An introduction*. 2nd. ed. [S.l.]: Bradford Book, 2018.

41 WATKINS, C. Learning from delayed rewards. 01 1989.

42 JúNIOR, F.; NETO, A. D.; MELO, J. Hybrid metaheuristics using reinforcement learning applied to salesman traveling problem. In: _____. [S.l.: s.n.], 2010. ISBN 978-953-307-426-9.

43 PUGLIESE, L. D. P.; GUERRIERO, F. Last-mile deliveries by using drones and classical vehicles. In: . [S.l.: s.n.], 2017. p. 557–565. ISBN 978-3-319-67307-3.

44 FARIA, E. B. R. *Heurísticas para o Problema de Roteamento de Veículos com Drones e Janelas de Tempo*. Tese (Monografia) — Universidade Federal de Juiz de Fora, Juiz de Fora, 2022.

45 SCHMIDT, J. *The Vehicle Routing Problem with Drones and Time Windows: Minimizing Route Duration*. Mainz, Germany, 2024.

46 HAN, J.; LIU, Y.; LI, Y. Vehicle routing problem with drones considering time windows and dynamic demand. *Applied Sciences*, v. 13, n. 24, 2023. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/13/24/13086>.

47 SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, v. 35, n. 2, p. 254–265, 1987.

48 LóPEZ-IBáñEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43–58, 2016. ISSN 2214-7160.

49 KLEFSJö, B. Ttt-plotting — a tool for both theoretical and practical problems. *Journal of Statistical Planning and Inference*, v. 29, n. 1, p. 99–110, 1991. ISSN 0378-3758. Disponível em: <https://www.sciencedirect.com/science/article/pii/037837589290125C>.

50 AIEX, R. M.; RESENDE, M. G. C.; RIBEIRO, C. C. Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, v. 1, n. 4, p. 355–366, Sep 2007. ISSN 1862-4480. Disponível em: <https://doi.org/10.1007/s11590-006-0031-4>.