Universidade Federal de Juiz de Fora
Faculdade de Engenharia e Instituto de Ciências Exatas
Bacharelado em Engenharia Computacional

Lucas de Castro Fernandino

Um algoritmo Hybrid Genetic Search para o Problema de Roteamento de Veículos com Armários de Encomendas

Juiz de Fora
2025

Lucas de Castro Fernandino

Um algoritmo Hybrid Genetic Search para o Problema de Roteamento de Veículos com Armários de Encomendas

Trabalho de Conclusão de Curso apresentado ao corpo docente do curso de graduação em Engenharia Computacional da Faculdade de Engenharia e Instituto de Ciências Exatas 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

Lucas de Castro Fernandino,

Um algoritmo Hybrid Genetic Search para o Problema de Roteamento de Veículos com Armários de Encomendas/ Lucas de Castro Fernandino. - - 2025.

XI, 58 p.: il.; 29, 7cm.

Orientadora: Lorenza Leão Oliveira Moreno

Coorientadora: Luciana Brugiolo Gonçalves

Trabalho de Conclusão de Curso (bacharelado) – Universidade Federal de Juiz de Fora, ICE/Engenharia. Bacharelado em Engenharia Computacional, 2025.

Referências Bibliográficas: p. 55 – 58.

1. Roteamento de Veículos. 2. Armários de Encomendas. 3. Logística de Última Milha. 4. Metaheurísticas. 5. Hybrid Genetic Search. I. Leão Oliveira Moreno, Lorenza *et al.*. II. Universidade Federal de Juiz de Fora, Bacharelado em Engenharia Computacional.

Dedico este trabalho à minha mãe, que sempre fez o possível e o impossível para que eu chegasse até aqui.

AGRADECIMENTOS

Agradeço primeiramente à minha família, pelo apoio incondicional, pelo incentivo constante e por sempre acreditarem em mim, mesmo nos momentos mais difíceis.

Às minhas orientadoras, Lorenza e Luciana, expresso minha gratidão pela paciência e pela atenção dispensadas ao longo deste trabalho. Obrigado pela orientação e por todo o conhecimento compartilhado.

Aos meus amigos de longa data, agradeço por estarem sempre presentes, tornando a vida mais leve e oferecendo o refúgio necessário fora da vida acadêmica.

Aos colegas de graduação que caminharam ao meu lado nessa longa trajetória. Obrigado pela parceria, pelas trocas de conhecimento e pelo companheirismo nos momentos de desafio.

Por fim, estendo meus agradecimentos aos demais professores e funcionários da Universidade Federal de Juiz de Fora, que contribuíram, direta ou indiretamente, para a minha formação profissional e pessoal.

RESUMO

O uso de armários de encomendas tem se consolidado como uma alternativa relevante para a logística de última milha, motivando o estudo do Problema de Roteamento de Veículos com Armários de Encomendas (VRPPL), uma variante recente do clássico Problema de Roteamento de Veículos (VRP) que integra decisões de roteamento e de alocação de clientes a pontos de consolidação. Este trabalho avalia a aplicação de uma abordagem baseada em *Hybrid Genetic Search* (HGS) para o VRPPL, implementada com base no *framework PyVRP* e adaptada por meio de uma modelagem baseada em grupos de clientes mutuamente exclusivos e de um operador de busca local voltado à consolidação de visitas a armários. A abordagem proposta foi avaliada em instâncias da literatura e comparada com métodos baseados em *Simulated Annealing* e *Iterated Greedy*. Os experimentos mostram que o método apresenta desempenho competitivo, com vantagens claras em relação ao *Simulated Annealing* em tempo de execução e qualidade de solução (Gap) próxima à obtida pelo *Iterated Greedy*, indicando que o HGS constitui uma alternativa viável e robusta para a resolução do VRPPL.

Palavras-chave: Roteamento de Veículos. Armários de Encomendas. Logística de Última Milha. Meta-heurísticas. Hybrid Genetic Search.

ABSTRACT

Parcel lockers have become an increasingly relevant alternative for last-mile logistics, motivating the study of the Vehicle Routing Problem with Parcel Lockers (VRPPL), a recent variant of the classic Vehicle Routing Problem (VRP) that integrates routing decisions with customer-to-locker allocation. This work evaluates the application of a *Hybrid Genetic Search* (HGS)-based approach to the VRPPL, implemented using the *PyVRP* framework and adapted through a modeling strategy based on mutually exclusive customer groups and a local search operator designed to consolidate visits to parcel lockers. The proposed approach was evaluated on benchmark instances from the literature and compared with methods based on Simulated Annealing and Iterated Greedy. Experiments demonstrate that the method achieves competitive performance, showing clear advantages over Simulated Annealing regarding runtime, and solution quality comparable to that obtained by the Iterated Greedy, indicating that the HGS represents a viable and robust alternative for solving the VRPPL.

Keywords: Vehicle Routing. Parcel Lockers. Last-Mile Logistics. Metaheuristics. Hybrid Genetic Search.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

GA	Algoritmo Genético
HGS	*Hybrid Genetic Search*
TSP	Problema do Caixeiro Viajante
UFJF	Universidade Federal de Juiz de Fora
VRP	Problema de Roteamento de Veículos
VRPTW	Problema de Roteamento de Veículos com Janelas de Tempo
VRPPL	Problema de Roteamento de Veículos com Armários de Encomendas
CVRP	Problema de Roteamento de Veículos com Capacidade
VRPPD	Problema de Roteamento de Veículos com Coletas e Entregas
SREX	*Selective Route Exchange*
SA	*Simulated Annealing*
IG	*Iterated Greedy*
HC	*Home Delivery Customer*
LC	*Locker Delivery Customer*
HLC	*Home or Locker Delivery Customer*
ILS	*Iterated Local Search*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, tem-se observado um crescimento expressivo do setor de comércio eletrônico (*e-commerce*) em escala global. Esse fenômeno atua como um dos principais catalisadores para o aumento da demanda por serviços de entrega mais rápidos e eficientes, impondo desafios logísticos cada vez mais complexos às empresas. Segundo estimativas de mercado, o setor global de logística, avaliado em aproximadamente U$ 5,65 trilhões em 2024, possui projeção de crescimento para cerca de U$ 8,07 trilhões até 2033 (1). Tal expansão é impulsionada por consumidores progressivamente mais exigentes quanto à conveniência, confiabilidade e velocidade das entregas. Um exemplo recente é o anúncio, em maio de 2025, da ampliação dos serviços de *Same-Day Delivery* da Amazon em diversos países da Europa (2).

Nesse contexto, a logística de última milha (*last mile*), correspondente à etapa final do processo de distribuição entre o centro logístico e o consumidor final, destaca-se como um dos componentes mais críticos e onerosos da cadeia logística. Um planejamento inadequado dessa etapa pode resultar em aumento da congestão urbana, maior emissão de poluentes e elevação significativa dos custos operacionais. Dessa forma, a busca por estratégias mais eficientes e sustentáveis para a última milha tornou-se um tema central tanto no meio industrial quanto na comunidade acadêmica.

Uma das abordagens mais recentes para mitigar esses desafios é a adoção de armários de encomendas (*parcel lockers*). Esses sistemas consistem em armários automatizados que permitem a retirada de encomendas em horários flexíveis, funcionando como uma alternativa ao atendimento domiciliar tradicional. O uso de *parcel lockers* contribui para a redução de tentativas de entrega malsucedidas, frequentemente causadas pela ausência do cliente, além de permitir a consolidação de múltiplas entregas em pontos estratégicos. Atualmente, tais sistemas encontram-se disseminados em diversos países, sendo instalados em locais como estações de transporte, centros comerciais e áreas residenciais, desempenhando um papel relevante na melhoria do nível de serviço ao consumidor e na eficiência operacional da logística urbana (3).

A introdução dos armários de encomendas, entretanto, acrescenta novas camadas de

complexidade ao tradicional Problema de Roteamento de Veículos com Janelas de Tempo (*Vehicle Routing Problem with Time Windows* – VRPTW). Além das decisões clássicas de roteamento e sequenciamento de visitas, passam a ser consideradas escolhas relacionadas à atribuição de clientes a armários, à capacidade desses pontos de consolidação e à interação entre janelas de tempo, tempos de serviço e restrições operacionais (4). Essas características motivaram o surgimento do Problema de Roteamento de Veículos com Armários de Encomendas (VRPPL, do inglês *Vehicle Routing Problem with Parcel Lockers*), uma variante recente do VRP que reflete com maior fidelidade os desafios contemporâneos da logística de última milha.

No VRPPL, os clientes podem ser classificados em diferentes perfis, incluindo aqueles que exigem atendimento domiciliar, aqueles que utilizam exclusivamente armários de encomendas e aqueles que aceitam ambas as modalidades. O problema considera ainda restrições como capacidade dos veículos e janelas de tempo, tendo como objetivo principal a minimização do custo total de roteamento, tipicamente associado à distância percorrida ou ao tempo total de operação.

Diante desse cenário, o presente trabalho tem como objetivo avaliar a viabilidade e o desempenho de uma abordagem baseada em *Hybrid Genetic Search* (HGS) (5) para a resolução do VRPPL. A metodologia adotada fundamenta-se no uso do *framework PyVRP* (6), uma implementação moderna do HGS originalmente desenvolvida para variantes clássicas do VRP com janelas de tempo. A adaptação ao contexto do VRPPL foi realizada por meio de uma modelagem compatível com as estruturas disponíveis na ferramenta, explorando o conceito de grupos de clientes para representar alternativas mutuamente exclusivas de atendimento, bem como por meio de ajustes no processo de avaliação das rotas e da implementação de um operador específico de busca local voltado à consolidação de visitas a armários de encomendas.

Além da proposição e adaptação do método, este trabalho realiza uma análise computacional comparativa com abordagens recentes da literatura, em particular com os métodos baseados em *Simulated Annealing* e *Iterated Greedy* propostos por Yu et al. (7) e Corrêa et al. (8), respectivamente. A comparação é conduzida a partir do uso das

mesmas instâncias de teste e critérios de avaliação, permitindo uma análise consistente da qualidade das soluções e do esforço computacional requerido.

O restante desta monografia está organizado da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica relacionada ao Problema de Roteamento de Veículos, suas principais variantes e as técnicas heurísticas e meta-heurísticas relevantes. O Capítulo 3 discute trabalhos relacionados ao VRPPL e formaliza a definição do problema tratado. No Capítulo 4 são detalhados a metodologia e os componentes do algoritmo proposto. O Capítulo 5 apresenta os experimentos computacionais, as instâncias utilizadas e a análise dos resultados obtidos. Por fim, o Capítulo 6 reúne as conclusões e aponta direções para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos teóricos necessários para compreender o VRPPL e o conjunto de técnicas de otimização utilizadas para solucioná-lo. Discute-se inicialmente a formulação clássica do Problema de Roteamento de Veículos (VRP) e suas variantes mais relevantes, destacando-se sua relação com o Problema do Caixeiro Viajante (TSP), sua natureza combinatória e sua importância prática em aplicações contemporâneas de logística urbana. Em seguida, introduz-se o VRPPL de forma conceitual, evidenciando os desafios decorrentes da integração entre decisões de roteamento e de alocação de clientes a armários de encomendas. Por fim, são discutidas heurísticas, buscas locais e meta-heurísticas modernas baseadas tanto em solução única quanto em populações, com especial atenção para algoritmos genéticos, algoritmos meméticos e para o método *Hybrid Genetic Search* (HGS), que constitui a base algorítmica empregada neste trabalho.

2.1 O PROBLEMA DE ROTEAMENTO DE VEÍCULOS

O Problema do Caixeiro Viajante (TSP) é uma das formulações mais conhecidas da otimização combinatória. Nele, busca-se construir um ciclo hamiltoniano de custo mínimo que visite um conjunto de cidades exatamente uma vez. Apesar da aparente simplicidade, o TSP é classificado como NP-difícil e influencia a formulação de uma vasta gama de problemas de roteamento. O VRP pode ser interpretado como uma generalização direta do TSP: em vez de um único agente percorrendo todos os vértices, considera-se uma frota de veículos, cada qual com restrições operacionais específicas, simulando o comportamento real de sistemas logísticos (9). Essa generalização, proposta originalmente por Dantzig e Ramser na década de 1950, consolidou-se como um dos pilares da pesquisa em otimização aplicada ao transporte (10).

Em sua formulação mais básica, o VRP consiste em determinar rotas que partem de um depósito central e atendem a um conjunto de clientes, retornando ao ponto de origem, de maneira a minimizar um custo global que normalmente está relacionado à distância percorrida, ao tempo total de rota ou ao número de veículos empregados. Cada cliente demanda um serviço cuja natureza pode envolver entrega, coleta ou ambos, e cada veículo

dispõe de uma capacidade finita que não pode ser excedida. Além dessas restrições, muitas variantes adicionam elementos como janelas de tempo para atendimento, limites máximos de duração de rota, prioridades de visita e regras de precedência. A combinação dessas características torna o VRP um problema de difícil estruturação, no qual a procura por soluções exatas rapidamente se torna inviável à medida que cresce o número de clientes. Essa inviabilidade decorre diretamente de sua natureza NP-difícil, fato amplamente documentado na literatura e discutido em trabalhos clássicos como os de Toth e Vigo (9).

A importância prática do VRP estimulou o desenvolvimento de diversas variantes que modelam cenários logísticos específicos (9; 11). Entre elas, destacam-se o VRP com capacidade (CVRP), no qual o desafio central é respeitar o limite de carga dos veículos, e o VRP com janelas de tempo (VRPTW), que impõe intervalos temporalmente rígidos para atendimento de cada cliente, tornando o sequenciamento das visitas um componente tão relevante quanto a própria distância percorrida (12). Outras variantes incluem problemas com coletas e entregas, múltiplos depósitos, frotas heterogêneas, sincronização de veículos, restrições ambientais e, mais recentemente, o uso de pontos alternativos de entrega, como armários de encomendas, que fundamentam o VRPPL.

2.2 O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM ARMÁRIOS DE ENCOMENDAS

O VRPPL surge em resposta à crescente complexidade da distribuição de encomendas na etapa de *last mile* (13). Nesse contexto, armários de encomendas, ou *parcel lockers*, desempenham o papel de pontos intermediários de consolidação, proporcionando maior flexibilidade ao sistema logístico. Em vez de visitar individualmente cada cliente, o veículo pode entregar múltiplas encomendas em um único *locker*, onde posteriormente serão retiradas pelos destinatários. Esse modelo reduz deslocamentos dispersos, diminui o tempo médio de serviço por entrega e pode mitigar problemas comuns do atendimento porta a porta, tais como janelas de disponibilidade restritas, insucesso nas entregas e alta variabilidade temporal (14).

No VRPPL, os clientes são tradicionalmente classificados em três categorias (7). Há os

clientes que devem ser atendidos diretamente em suas residências, sem opção de retirada em armário; aqueles que, por suas características ou preferências, só podem receber suas encomendas via *locker*; e, por fim, clientes flexíveis, que podem realizar a retirada em um armário ou receber diretamente em casa. Essa flexibilidade aumenta substancialmente o espaço de busca do problema, pois cria decisões adicionais além do roteamento, sendo necessário determinar a forma de atendimento de cada cliente que possa ser atendido das duas formas.

A introdução dos armários de encomendas impõe ainda outras considerações operacionais. Os *lockers* possuem capacidade limitada, o que exige que a alocação de clientes seja compatibilizada com a disponibilidade de compartimentos. Em alguns cenários, o uso dos armários está sujeito a janelas de funcionamento, o que implica sincronizar a visita do veículo aos horários de funcionamento (abertura e fechamento) dos locais (13; 14). A entrega em lockers promove a consolidação de cargas, reduzindo significativamente o número de visitas domiciliares e o tempo total de serviço (13). Entretanto, essa estratégia cria um *trade-off* operacional: a visita ao armário só se justifica se a economia de tempo nas paradas compensar o deslocamento até sua localização específica.

Do ponto de vista algorítmico, o VRPPL combina decisões de roteamento e de alocação, que devem ser tratadas de forma conjunta para que a solução reflita adequadamente as interações entre capacidade, janelas de tempo e disponibilidade dos armários. A literatura recente tem abordado explicitamente essa integração, como no estudo de Yu et al. (7), que trata simultaneamente a designação de clientes a armários e o roteamento dos veículos.

Mais recentemente, Corrêa et al. (8) propuseram uma abordagem alternativa para o VRPPL baseada em *Iterated Greedy*, aprofundando a análise computacional e mantendo compatibilidade com as instâncias originalmente introduzidas por Yu et al (7). Esse estudo reforça a relevância do conjunto de instâncias proposto por Yu et al. (7) como *benchmark* para avaliação comparativa de diferentes paradigmas meta-heurísticos aplicados ao problema.

2.3 HEURÍSTICAS, BUSCA LOCAL E META-HEURÍSTICAS

A resolução de variantes do VRP com realismo prático exige o uso de heurísticas e meta-heurísticas capazes de produzir soluções de qualidade em tempo computacional controlado. Heurísticas construtivas, como as propostas por Clarke e Wright (15), fornecem soluções iniciais rápidas, mas raramente competitivas quando comparadas a métodos de melhoramento. A aplicação de busca local aparece como uma etapa essencial nesse processo, desempenhando papel central em praticamente todas as abordagens modernas para problemas de roteamento.

A busca local consiste em partir de uma solução candidata e explorar sua vizinhança mediante pequenas modificações estruturais (16). Para VRPs, as vizinhanças são tradicionalmente geradas pela aplicação de operadores de movimento, tais como o ***Swap*** (troca de posições entre dois clientes), o ***Relocate*** (realocação de um cliente para outra posição/rota) e o ***2-Opt*** (reversão de um subsegmento da rota). O desempenho da busca local depende não apenas da escolha dessas vizinhanças, mas também da política de exploração. Estratégias como *best-improvement* examinam sistematicamente toda a vizinhança, selecionando o melhor movimento disponível, enquanto estratégias *first-improvement* aplicam o primeiro movimento que resulta em melhora, frequentemente reduzindo significativamente o tempo de processamento (16). Por definição, a busca local leva a ótimos locais, isto é, soluções em que nenhuma alteração imposta pelo movimento melhora o custo (17). Escapar desses pontos exige a aplicação de perturbações estruturadas ou de estratégias mais amplas de exploração, como as utilizadas em algoritmos de busca iterativa.

Ainda que técnicas baseadas em solução única, como *Simulated Annealing* (7) ou *Iterated Local Search* (16), sejam eficazes em muitos cenários, problemas de roteamento de maior escala frequentemente se beneficiam de abordagens baseadas em populações. Os Algoritmos Genéticos (GAs) constituem uma dessas alternativas, inspirando-se na teoria da evolução natural para explorar simultaneamente múltiplas regiões do espaço de busca (18). Em GAs, uma população de soluções evolui ao longo de gerações mediante operadores de seleção, recombinação e mutação. No entanto, a aplicação direta de GAs

ao VRP enfrenta dificuldades significativas, sobretudo porque representações tradicionais e operadores clássicos de recombinação não preservam a estrutura de rotas e podem gerar soluções inviáveis com grande frequência.

No contexto do VRPPL, Yu et al. (7) propuseram uma abordagem baseada em *Simulated Annealing* e introduziram um conjunto de instâncias de referência com janelas de tempo. Posteriormente, Corrêa et al. (8) demonstraram que o método *Iterated Greedy* também alcança resultados competitivos nesses mesmos cenários. O sucesso dessas meta-heurísticas de solução única motiva a investigação de paradigmas baseados em populações, capazes de explorar o espaço de busca com maior diversidade.

A literatura dá destaque a algoritmos meméticos, que combinam mecanismos evolutivos com busca local intensiva, sendo amplamente aplicados a problemas de roteamento de veículos modernos (19; 5; 20). Esse casamento entre exploração global e refinamento local tornou-se um paradigma dominante para problemas de roteamento. Em algoritmos meméticos bem-sucedidos, a população evolui de maneira diversificada, enquanto cada solução é melhorada individualmente mediante mecanismos de busca local, resultando em soluções de alta qualidade mesmo sob restrições complexas.

2.4 O MÉTODO HYBRID GENETIC SEARCH

O *Hybrid Genetic Search* (HGS), desenvolvido por Thibaut Vidal e colaboradores (5), é considerado uma das meta-heurísticas mais eficazes para o VRPTW e diversas outras variantes do VRP. Seu sucesso decorre de uma integração cuidadosa entre os princípios evolutivos dos GAs e uma busca local robusta, aliada a mecanismos de controle de diversidade e penalização adaptativa. A Figura 2.1 ilustra a estrutura geral do HGS e a interação entre seus principais componentes.

A base estrutural do HGS é a representação em *giant tour*, na qual todas as visitas são concatenadas em uma única sequência. Essa representação é posteriormente decomposta em rotas viáveis por meio do algoritmo *Split*, que avalia cortes potenciais ao longo da sequência e constrói um conjunto de rotas que respeitam as restrições do problema. Essa estratégia separa a complexidade da decisão de roteamento entre duas camadas: uma

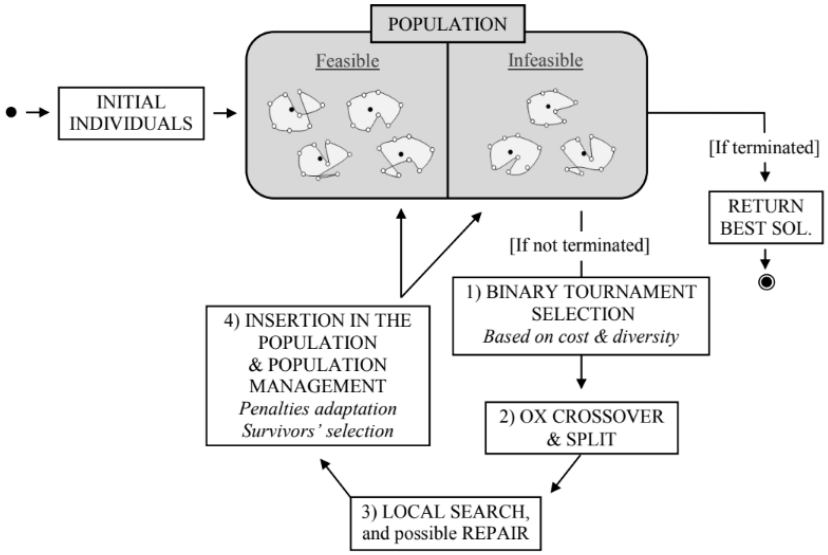


Figura 2.1: Esquema geral do método *Hybrid Genetic Search*.
Fonte: Vidal (21).

camada evolutiva que manipula sequências e uma camada de decodificação responsável por verificar e corrigir viabilidade (5; 20; 21).

A etapa evolutiva do HGS é conduzida por operadores genéticos projetados especificamente para o VRP. A recombinação é realizada de modo a preservar estruturas relevantes das rotas parentais, evitando degenerações comuns em operadores tradicionais. A seleção equilibra qualidade e diversidade, impedindo que a população convirja prematuramente para bacias de atração rasas do espaço de busca. A mutação é usada de maneira controlada, introduzindo pequenas perturbações que ajudam a explorar novas soluções sem destruir características essenciais da rota.

Cada indivíduo gerado pela etapa genética é submetido a uma busca local extensiva, baseada em uma variedade de movimentos clássicos de roteamento, como *2-opt*, *relocate*, *swap* e *cross-exchange*. O objetivo é refinar significativamente as soluções, corrigindo eventuais distorções introduzidas pelo crossover e aproximando o indivíduo de um ótimo local de alta qualidade. Essa intensificação é essencial para o desempenho do HGS, pois permite que a componente genética se concentre na exploração global, deixando à busca local o papel de ajustar detalhes finos da solução.

Outro elemento essencial do HGS são os mecanismos de penalização adaptativa,

que ajustam dinamicamente os pesos associados a violações de capacidade, janelas de tempo e outras restrições, conforme a proporção de indivíduos viáveis e inviáveis na população. Quando a população se torna excessivamente inviável, as penalidades aumentam, forçando o algoritmo a privilegiar soluções que respeitem as restrições. Quando a população se torna demasiado restrita, com poucos indivíduos capazes de explorar novas regiões, as penalidades diminuem, incentivando o algoritmo a considerar soluções menos convencionais. Esse mecanismo confere ao HGS uma capacidade notável de equilibrar intensificação e diversificação ao longo do processo de busca.

Finalmente, o controle da população é realizada por meio de políticas que removem duplicatas e reintroduzem diversidade sempre que necessário. A eliminação de indivíduos excessivamente similares mantém o algoritmo ativo e reduz o risco de convergência prematura, enquanto estratégias de reinicialização parcial evitam estagnações prolongadas. Essa arquitetura híbrida e cuidadosamente balanceada torna o HGS uma das abordagens mais robustas disponíveis para problemas de roteamento complexos, o que justifica sua adoção no contexto do VRPPL.

Mais recentemente, o método *Hybrid Genetic Search* tem sido objeto de implementações modernas que adaptam sua estrutura original a diferentes variantes do VRP e a requisitos computacionais contemporâneos, mantendo seus princípios fundamentais (6). Essas implementações mantêm os princípios centrais do HGS proposto por Vidal (21), como a integração entre algoritmos genéticos e busca local intensiva, o uso de penalidades adaptativas e o controle explícito de diversidade, ao mesmo tempo em que introduzem modificações na representação das soluções e na forma de avaliação das rotas.

Em particular, algumas abordagens substituem a representação baseada em *giant tours* e no algoritmo *Split* por representações diretas em rotas explícitas, permitindo maior flexibilidade na modelagem de restrições adicionais, como janelas de tempo complexas e estruturas de clientes mutuamente exclusivos. Essas adaptações têm se mostrado eficazes para lidar com variantes do VRP de maior complexidade estrutural, servindo como base para aplicações práticas e extensões metodológicas, como a considerada neste trabalho.

3 PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM ARMÁRIOS DE ENCOMENDAS

O Problema de Roteamento de Veículos com Armários de Encomendas (VRPPL) consiste em determinar rotas para uma frota de veículos capacitados, que partem de um depósito central para atender a clientes distribuídos geograficamente, podendo realizar entregas diretamente na residência dos clientes ou por meio de armários de encomendas (*parcel lockers*). Diferentemente do VRP clássico, o VRPPL envolve não apenas a decisão de roteamento, mas também a decisão de alocação de clientes a armários, quando essa opção estiver disponível. Essa característica acopla decisões de designação de atendimento e decisões de roteamento, alterando significativamente a estrutura do espaço de busca e a natureza combinatória do problema.

A definição formal do VRPPL apresentada neste trabalho baseia-se na modelagem proposta por Corrêa et al. (8). Dessa forma, são adotados os mesmos conjuntos, parâmetros e restrições utilizados no referido trabalho, garantindo consistência na descrição do problema e permitindo comparações diretas nos experimentos computacionais realizados.

Este capítulo tem por objetivo aprofundar a formulação do VRPPL, oferecendo uma descrição detalhada de seus elementos operacionais e apresentando a modelagem matemática que captura suas principais restrições e interdependências. Após a modelagem, discute-se o panorama da literatura relacionada, contextualizando o VRPPL no âmbito da logística de última milha.

3.1 DESCRIÇÃO DO PROBLEMA

O VRPPL é definido sobre um grafo completo não direcionado $G = (N, E)$, no qual N representa o conjunto de nós e E o conjunto de arestas. O conjunto de nós é particionado em três subconjuntos: o conjunto de depósitos D, o conjunto de clientes C e o conjunto de armários de encomendas (*parcel lockers*) P, de modo que $N = D \cup C \cup P$. Para todo par distinto de nós $i, j \in N$, considera-se a existência de uma aresta $\{i, j\} \in E$, associada a um custo de deslocamento c_{ij} e a um tempo de viagem t_{ij}, ambos não negativos.

Os clientes são classificados em três tipos de acordo com a preferência de entrega:

- **Clientes domiciliares (C_H):** Devem ser atendidos obrigatoriamente em suas residências.
- **Clientes de locker (C_S):** Devem ser atendidos obrigatoriamente através de um armário designado.
- **Clientes flexíveis (C_F):** Podem ser atendidos tanto em domicílio quanto em um armário associado.
- Note que $C = C_H \cup C_S \cup C_F$.

Cada cliente $i \in C_S \cup C_F$ possui um armário associado ρ_i, especificado na instância do problema. Além disso, cada cliente possui uma demanda d_i, um tempo de serviço s_i e uma janela de tempo $[a_i, b_i]$ para o atendimento domiciliar.

O mecanismo de atendimento via armário introduz uma diferença estrutural importante: o veículo deixa de visitar o cliente diretamente para visitar o armário associado. Essa mudança altera a sequência de visitas e a dinâmica temporal da rota, pois, embora a parada no *locker* demande tempo, o atendimento fica desacoplado da janela de tempo do cliente. Essa flexibilidade permite que o veículo realize a entrega no momento mais eficiente para o roteamento, sendo a visita ao *locker* condicionada à existência de pelo menos um pacote a ser entregue naquele local.

Os veículos, representados pelo conjunto V, possuem capacidade Q, partem do depósito, percorrem a rede e retornam ao depósito. O objetivo é determinar simultaneamente o modo de atendimento dos clientes flexíveis e as rotas que minimizam o custo total de deslocamento.

3.2 MODELAGEM MATEMÁTICA

A seguir apresenta-se a formulação matemática do VRPPL, conforme proposta por Corrêa et al. (8). A Tabela 3.1 resume a notação utilizada.

Tabela de Notação

Constantes	**Descrição**
N	Conjunto de nós.
D	Conjunto de depósitos, $D \in N$.
P	Conjunto de *Parcel Lockers*, $P \in N$.
C	Conjunto de clientes, $C \subset N$.
C_H	Conjunto de clientes atendidos em casa, $C_H \subset C$.
C_S	Conjunto de clientes atendidos em *lockers*, $C_S \subset C$.
C_F	Conjunto de clientes flexíveis, $C_F \subset C$.
V	Conjunto de veículos.
Q	Capacidade do veículo.
a_i	Limite inferior da janela de tempo do cliente i, $i \in N$.
b_i	Limite superior da janela de tempo do cliente i, $i \in N$.
t_{ij}	Tempo de viagem de i para j, $i, j \in N$.
s_i	Tempo de serviço do cliente/*locker* i, $i \in C \cup P$.
d_i	Demanda do cliente i, $i \in C$.
ρ_i	O *Parcel Locker* escolhido pelo cliente $i \in C_S \cup C_F$.
Variáveis	**Descrição**
x_{ijk}	Indica se o veículo k utiliza a aresta (i, j). $i, j \in N, k \in V, x_{ijk} \in \{0, 1\}$.
h_i	Indica se o cliente i é atendido em casa. $i \in C_H \cup C_F, h_i \in \{0, 1\}$.
l_i	Indica se o cliente i é atendido no PL. $i \in C_S \cup C_F, l_i \in \{0, 1\}$.
ψ_{pk}	Quantidade de itens entregues no PL p pelo veículo k. $k \in V, p \in P$.
∂_k	Momento em que o veículo k deixa o depósito. $k \in V$.
μ_{ik}	Momento em que o veículo k inicia o serviço no nó i. $i \in C, k \in V$.
y_{ip}	Indica se o cliente i é atendido no *locker* p. $i \in C_S \cup C_F, p \in P, y_{ip} \in \{0, 1\}$.

Tabela 3.1: Tabela de símbolos do modelo matemático.

Fonte: Traduzido de (8).

Formulação

O objetivo do problema é minimizar a distância total percorrida pela frota:

$$\text{Minimizar } Z = \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{3.1}$$

Sujeito a:

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \qquad \forall i \in C_H \tag{3.2}$$

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 0, \qquad \forall i \in C_S \tag{3.3}$$

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = h_i, \qquad \forall i \in C_F \tag{3.4}$$

$$\sum_{j \in C \cup P} x_{Djk} \leq 1, \qquad \forall k \in V \tag{3.5}$$

$$\sum_{i \in N,\ i \neq j} x_{ijk} - \sum_{i \in N,\ i \neq j} x_{jik} = 0, \qquad \forall k \in V,\ \forall j \in N \tag{3.6}$$

$$\sum_{i \in N} \sum_{\substack{j \in C \\ i \neq j}} d_j\, x_{ijk} + \sum_{p \in P} \psi_{pk} \leq Q, \qquad \forall k \in V \tag{3.7}$$

$$\partial_k + t_{Dj} - \mu_{jk} \leq M(1 - x_{Djk}), \qquad \forall k \in V, \forall j \in C \cup P \tag{3.8}$$

$$\mu_{ik} + s_i + t_{ij} - \mu_{jk} \leq M(1 - x_{ijk}), \qquad \forall k \in V, \forall i, j \in C \cup P \tag{3.9}$$

$$a_i\, h_i \leq \sum_{k \in V} \mu_{ik} \leq b_i\, h_i, \qquad \forall k \in V, i \in C \tag{3.10}$$

$$h_i = 1, \qquad \forall i \in C_H \tag{3.11}$$

$$l_i = 1, \qquad \forall i \in C_S \tag{3.12}$$

$$l_i + h_i = 1, \qquad \forall i \in C_F \tag{3.13}$$

$$\sum_{p \in P} y_{ip} = l_i, \qquad \forall i \in C \tag{3.14}$$

$$y_{ip} \leq 1, \qquad \forall i \in C,\ p = \rho_i \tag{3.15}$$

$$y_{ip} = 0, \qquad \forall i \in C,\ p \neq \rho_i \tag{3.16}$$

$$\sum_{k \in V} \psi_{pk} = \sum_{i \in C_S \cup C_F} d_i \, y_{ip}, \qquad \forall p \in P \tag{3.17}$$

$$\sum_{i \in N} Q \, x_{ipk} \geq \psi_{pk}, \qquad \forall p \in P, \ \forall k \in V \tag{3.18}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall i, j \in N, \ \forall k \in V \tag{3.19}$$

$$h_i, \, l_i \in \{0, 1\}, \qquad \forall i \in C \tag{3.20}$$

$$y_{ip} \in \{0, 1\}, \qquad \forall i \in C, \ \forall p \in P \tag{3.21}$$

$$\psi_{pk} \geq 0, \qquad \forall p \in P, \ \forall k \in V \tag{3.22}$$

$$\mu_{ik} > 0, \qquad \forall i \in N, \ \forall k \in V \tag{3.23}$$

$$\partial_k \geq 0, \qquad \forall k \in V \tag{3.24}$$

A Função Objetivo (3.1) visa minimizar o custo total de deslocamento da frota. As restrições (3.2)-(3.4) controlam o fluxo de visitação de acordo com a categoria do cliente: clientes C_H devem ser visitados obrigatoriamente, clientes C_S não recebem visita direta e clientes C_F são visitados apenas se a variável h_i indicar atendimento domiciliar.

As restrições (3.5) e (3.6) garantem a conservação de fluxo dos veículos, assegurando que cada rota inicie e termine no depósito. A limitação de capacidade é imposta pela restrição (3.7), que considera tanto a demanda das visitas diretas quanto o volume consolidado de entregas nos armários (ψ_{pk}).

O aspecto temporal é modelado pelas restrições (3.8)-(3.10), que estabelecem a propagação dos tempos de chegada e garantem que o início do serviço ocorra dentro das janelas de tempo especificadas para os clientes atendidos em casa.

As restrições (3.11)-(3.13) definem a consistência lógica do modo de atendimento para cada subconjunto de clientes. O vínculo entre o cliente e seu armário específico é tratado em (3.14)-(3.16), assegurando que, se a opção de *locker* for escolhida ($l_i = 1$), o pacote seja atribuído ao armário correto ρ_i.

Por fim, a restrição (3.17) contabiliza a carga total entregue em um armário, enquanto (3.18) condiciona essa entrega à visita do veículo ao respectivo *locker*. As demais equações definem os domínios das variáveis de decisão.

3.3 TRABALHOS RELACIONADOS

O estudo do VRPPL insere-se em uma linha crescente de pesquisa sobre logística de última milha e pontos alternativos de entrega. Boysen et al. (13) discutem diversas estruturas operacionais baseadas em armários e pontos de retirada, analisando seus impactos no custo logístico e na satisfação do consumidor. Iwan et al. (14) realizam análises empíricas do desempenho dos sistemas de armários, reforçando seu papel na redução de falhas de entrega e na consolidação de cargas.

Especificamente sobre o VRPPL, o trabalho de Yu et al. (7) apresenta a formulação pioneira do problema e propõe uma heurística baseada em *Simulated Annealing*, capaz de tratar simultaneamente decisões de atendimento e de roteamento.

Nesse contexto, Corrêa et al. (8) apresentam uma abordagem recente para o VRPPL baseada em *Iterated Greedy*. Os autores refinam a modelagem matemática (adotada neste capítulo) e exploram um esquema de destruição e reconstrução iterativa, fornecendo resultados comparativos que servem como base para a avaliação de novas propostas meta-heurísticas aplicadas ao problema.

Por fim, a literatura de meta-heurísticas híbridas, em especial os trabalhos de Vidal et al. (5; 20; 21), destaca o potencial de algoritmos baseados em populações aprimorados por busca local para resolver problemas de roteamento de alta complexidade. A adoção dessas estratégias no contexto do VRPPL constitui uma linha de pesquisa promissora, que fundamenta a contribuição algorítmica deste trabalho.

4 ALGORITMO PROPOSTO

Este capítulo apresenta em detalhes o algoritmo utilizado para resolver o VRPPL. O método baseia-se nos princípios do *Hybrid Genetic Search* (HGS), originalmente proposto em (5), que alia a capacidade de exploração global dos algoritmos evolutivos ao refinamento intensivo da busca local. A base computacional deste trabalho, contudo, fundamenta-se na implementação moderna e unificada do algoritmo, disponibilizada como código aberto e descrita em (21).

Nesta versão mais recente, Vidal destaca uma reestruturação completa da arquitetura do código visando simplicidade e acessibilidade, sem renunciar à eficiência que posicionou o método no estado da arte. A implementação distingue-se por aliar o gerenciamento robusto da diversidade populacional (fundamental para evitar a convergência prematura) à introdução do operador de vizinhança *SWAP**, projetado para avaliar trocas de clientes entre rotas com alto desempenho computacional, mesmo em instâncias de grande porte.

Para a aplicação ao VRPPL neste trabalho, utilizou-se o pacote PyVRP (6). Esta biblioteca funciona como uma interface de alto nível construída diretamente sobre o núcleo em C++ do código moderno de Vidal. A escolha pelo PyVRP justifica-se por unir a eficiência do motor original, preservando o operador *SWAP** e a gestão de população, à flexibilidade necessária para modelar as restrições de armários. Um diferencial decisivo para este trabalho é que a biblioteca estende o algoritmo original ao incorporar suporte nativo a janelas de tempo e ao mecanismo de *time warp*. Essa funcionalidade, que permite a admissão controlada de penalidades por atraso (detalhado na Seção 4.2), é essencial para o problema abordado e não está presente na versão original do código aberto de Vidal, a qual é restrita ao problema capacitado clássico (CVRP).

As próximas seções apresentam, em ordem, a arquitetura geral do algoritmo, o mecanismo de avaliação de soluções, a modelagem do VRPPL no PyVRP, os principais mecanismos evolutivos (seleção, recombinação e reparação), os operadores de busca local e, por fim, os componentes de penalização adaptativa, controle de diversidade e critério de parada.

4.1 ARQUITETURA GERAL DO HGS

O PyVRP, como se baseia no HGS de Vidal, adota uma arquitetura evolutiva baseada na interação entre dois componentes principais: um algoritmo genético, responsável pela exploração global do espaço de soluções, e uma busca local intensiva, responsável pela intensificação e refinamento estrutural das rotas. As soluções são representadas diretamente como sequências de visitas completas, organizadas em rotas explícitas. Assim, diferentemente do HGS clássico de Vidal, que opera sobre um *giant tour* posteriormente particionado por um algoritmo de *split*, o PyVRP utiliza uma representação mais natural e intuitiva, na qual cada solução é composta por um conjunto de rotas explícitas, ou seja, uma sequência de clientes entre depósitos para cada veículo.

Uma característica importante do HGS é a manutenção de duas subpopulações paralelas: uma de soluções viáveis e outra de soluções inviáveis. Esta estratégia permite ao algoritmo preservar diversidade estrutural, evitar convergência prematura e explorar regiões promissoras que violam temporariamente algumas restrições. A migração entre as duas subpopulações ocorre de forma natural conforme os processos de recombinação, reparação e melhoria vão gerando novos indivíduos.

No HGS, soluções inviáveis desempenham um papel ativo na busca, pois permitem explorar regiões do espaço de soluções que estariam inacessíveis caso apenas indivíduos estritamente factíveis fossem considerados. Pequenas violações de capacidade ou de janelas de tempo podem, por exemplo, facilitar rearranjos estruturais das rotas que posteriormente conduzem a soluções viáveis de melhor qualidade.

Quando o tamanho de uma subpopulação ultrapassa o limite máximo definido como parâmetro, é acionado um procedimento de descarte (*purge*). Nesse procedimento, soluções duplicadas são removidas em primeiro lugar. Em seguida, o *biased fitness* de todos os indivíduos remanescentes é atualizado. O *biased fitness* é uma métrica de classificação que combina qualidade e diversidade, consistindo na soma ponderada do ranking de custo (função objetivo) e do ranking de diversidade (por exemplo, a distância média para um conjunto de soluções próximas). Enquanto o tamanho da subpopulação estiver acima do valor alvo, o algoritmo remove iterativamente as soluções com pior valor

dessa métrica, preservando assim o equilíbrio entre qualidade e diversidade na população.

A estrutura lógica global do algoritmo, consolidando as etapas descritas, é apresentada no Pseudocódigo 1.

Pseudocódigo 1: HGS (versão PyVRP)

```
1  Inicializar população P de forma aleatória;
2  Avaliar e separar viáveis / inviáveis;
3  while critério de parada não satisfeito do
4  |  Selecionar pais por torneio (biased fitness + diversidade);
5  |  Filho ← SREX(pai_1, pai_2);
6  |  Aplicar busca local (fase normal);
7  |  if Filho inviável e sorteio de reparo positivo then
8  |  |  Aplicar busca local com penalidades reforçadas (Repair);
9  |  end if
10 |  Atualizar penalidades adaptativas;
11 |  Inserir Filho em P e aplicar purge se necessário;
12 |  Atualizar melhor solução conhecida;
13 end while
14 return melhor solução conhecida;
```

Conforme detalhado no Pseudocódigo 1, o processo inicia-se com a geração de uma população de soluções aleatórias (linha 1), as quais são avaliadas e imediatamente segregadas nas subpopulações de indivíduos viáveis e inviáveis (linha 2). Em seguida, o método entra em seu ciclo evolutivo principal (linhas 3-13). A cada iteração, dois pais são selecionados por torneio binário (linha 4) e submetidos ao operador de recombinação SREX (linha 5). Embora operadores de cruzamento possam gerar múltiplos descendentes, a estratégia adotada nesta implementação produz um único filho por iteração, o qual é imediatamente submetido à fase de busca local (linha 6) para refinamento intensivo.

Esse descendente passa então por uma primeira fase de busca local granular com penalidades padrão (linha 6). Caso a solução permaneça inviável após essa etapa, o algoritmo verifica a necessidade de aplicar uma segunda fase de busca local com penalidades temporariamente reforçadas (mecanismo de reparo) (linhas 7-9). Por fim, as penalidades adaptativas são atualizadas (linha 10) e o descendente é inserido na população, momento em que o controle de diversidade e o mecanismo de descarte (*purge*) são executados (linhas 11-12). A melhor solução global é atualizada sempre que um novo indivíduo factível de menor custo é encontrado (linha 13).

Todo o processo é regido por uma função objetivo estendida que combina custo primário (distância percorrida) e penalidades por violações de capacidade e janelas de tempo, as quais são admitidas na subpopulação inviável. Diferentemente dessas restrições relaxáveis, a obrigatoriedade de visita e a seleção do local de entrega são tratadas de forma estrutural. A representação das soluções assegura que todo cliente seja atendido e impõe uma escolha mutuamente exclusiva para os grupos flexíveis, garantindo que a entrega ocorra ou na residência ou no *locker*, mas jamais em ambos.

As seções seguintes deste capítulo dedicam-se a detalhar as diversas etapas do algoritmo. Nelas, são discutidos em maior profundidade os mecanismos específicos de avaliação, os operadores genéticos e as estratégias de busca local que compõem a arquitetura apresentada.

4.2 AVALIAÇÃO DE SOLUÇÕES: CUSTO BASE E PENALIDADES

A avaliação de uma solução é realizada de forma incremental, permitindo calcular rapidamente o impacto de cada modificação realizada durante a busca local. O custo total avaliado pelo PyVRP é composto por duas parcelas principais: o custo base e as penalidades adaptativas associadas a violações de restrições.

Custo base

O custo base corresponde ao somatório dos custos de deslocamento ao longo das rotas. Rotas são formadas por sequências de vértices, e cada arco (i, j) contribui com um custo c_{ij} e um tempo de viagem t_{ij}, definidos a partir das matrizes de distância. Tempos de serviço são incorporados diretamente no tempo acumulado da rota, que é usado para os cálculos de janelas de tempo.

Time Warp e Janelas de Tempo

Janelas de tempo são tratadas no algoritmo por meio do mecanismo de *time warp*, que consiste em uma relaxação controlada dessas restrições. Cada nó i possui um intervalo de atendimento permitido $[a_i, b_i]$. Se o veículo chega antes de a_i, este simplesmente aguarda

até o início da janela. Quando a chegada ocorre após o limite superior b_i, o atraso é convertido em *time warp*, dado por: $a_i - b_i$. Esse valor representa o quanto a solução viola a janela de atendimento desse nó e é incorporado à função de custo como penalização. Ao mesmo tempo, para propagar o cronograma ao longo da rota, o algoritmo considera que o serviço no nó i inicia efetivamente em

$min\{a_i, b_i\}$, isto é, o relógio é projetado de volta para b_i quando ocorre atraso, de modo que o tempo de serviço subsequente não acumule esse excesso.

Essa interpretação do *time warp*, distinguindo a linha do tempo real da linha do tempo efetiva considerada pelo algoritmo, é ilustrada na Figura 4.1. Em vez de descartar soluções com pequenas violações, o *time warp* permite atrasos moderados, que são penalizados com um peso adaptativo. Esse peso aumenta quando as violações se tornam frequentes na população e diminui quando a população volta a produzir soluções factíveis.

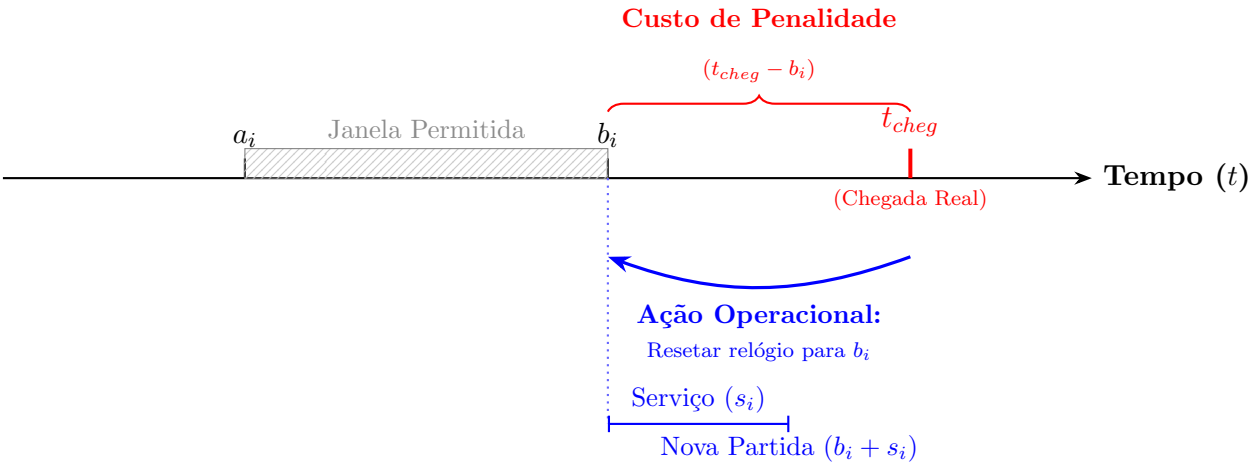


Figura 4.1: Mecanismo unificado do *time warp*. O atraso real (vermelho) gera custo, enquanto o ajuste operacional (azul) permite o início do serviço no fim da janela.

Fonte: Elaborado pelo autor.

Penalidades adaptativas

A função objetivo estendida inclui termos de penalização para:

- violações de capacidade dos veículos;
- atrasos (*time warp*) em janelas de tempo.

Seja S uma solução composta por um conjunto de rotas. O custo total avaliado pelo *evaluator* do PyVRP pode ser escrito, de forma simplificada, como

$$F(S) \;=\; C_{\text{base}}(S) \;+\; \lambda_{\text{cap}} \times \text{viol}_{\text{cap}}(S) \;+\; \lambda_{\text{tw}} \times \text{TW}(S), \tag{4.1}$$

em que $C_{\text{base}}(S)$ representa o custo base, correspondente ao somatório dos custos de deslocamento das rotas, $\text{viol}_{\text{cap}}(S)$ é uma medida agregada de violações de capacidade e $\text{TW}(S)$ é o *time warp* total acumulado ao longo das rotas.

Os coeficientes λ_{cap} e λ_{tw} são pesos de penalização adaptativos, gerenciados automaticamente por um gerenciador de penalidades. Esses pesos são ajustados periodicamente de acordo com a proporção de soluções viáveis observadas na população, de modo a manter um equilíbrio entre exploração de soluções inviáveis e intensificação na região viável. Já as restrições de completude (todos os clientes obrigatórios visitados) e de consistência dos grupos de clientes são tratadas como restrições duras e estruturais. Dessa forma, soluções que violem tais condições são imediatamente descartadas ou reparadas pelos operadores de busca, não sendo admitidas em nenhuma das subpopulações do algoritmo.

A configuração específica dos parâmetros de penalidade, bem como a justificativa para os ajustes adotados visando favorecer a exploração no VRPPL, são detalhadas na Seção 5.2, no contexto da configuração experimental.

4.3 ESTRATÉGIA DE MODELAGEM E REPRESENTAÇÃO NO PYVRP

O VRPPL estende o VRPTW tradicional ao incluir clientes que podem ser atendidos em domicílio, via *locker* ou ambos, dependendo do tipo. A estratégia de modelagem adotada neste trabalho para tratar o VRPPL dentro do *framework* PyVRP baseia-se na representação de nós individuais e no uso de grupos de clientes mutuamente exclusivos. Para implementar esse comportamento, cada localização física é representada como um vértice do grafo, com atributos de demanda, tempo de serviço e janela de disponibilidade.

Nessa estrutura, clientes do tipo HC são modelados como nós que obrigatoriamente

devem compor as rotas, enquanto clientes do tipo LC são representados exclusivamente por nós correspondentes aos armários designados. Já os clientes híbridos (HLC), que podem ser atendidos tanto no domicílio quanto no locker associado, são tratados através do conceito de *grupos de clientes*. Cada cliente híbrido i é modelado por dois nós distintos:

- C_i^{home}, correspondente ao atendimento domiciliar;
- C_i^{locker}, correspondente ao atendimento via armário associado.

Ambas as alternativas pertencem a um mesmo grupo mutuamente exclusivo, cuja cardinalidade desejada é igual a 1. Assim, uma solução é considerada viável em relação a esse grupo quando exatamente uma de suas alternativas está presente na rota.

A escolha entre atendimento domiciliar e via *locker* não é feita por um módulo dedicado, mas emerge da interação entre recombinação e busca local. Durante a recombinação, podem surgir descendentes contendo nenhuma, uma ou várias alternativas do mesmo grupo (por exemplo, domicílio e *locker* simultaneamente, ou nenhum dos dois). Em seguida, um operador específico de grupos na busca local inspeciona cada grupo para corrigir a viabilidade: se o grupo é obrigatório e está vazio, ele insere alguma alternativa; se há múltiplas alternativas ativas, remove o excesso deixando apenas uma, priorizando a manutenção daquela cuja remoção seria menos vantajosa.

Além da correção, esse operador também atua na otimização: ele pode substituir a alternativa atualmente ativa por outra do grupo sempre que essa troca reduzir o custo da solução. Dessa forma, a combinação entre recombinação, operador de grupos e penalidades adaptativas conduz, ao longo das iterações, à seleção da alternativa que oferece melhor equilíbrio entre custo de distância e violações de janelas de tempo (*time warp*).

Por fim, a estratégia de modelagem aborda a consolidação do tempo de serviço. No VRPPL, diversos clientes podem compartilhar o mesmo armário, mas na estrutura padrão do PyVRP cada nó possui seu próprio tempo de serviço, o que faria com que múltiplos clientes associados ao mesmo *locker* resultassem em múltiplas contagens de tempo ao visitar essa localização física.

Para evitar esse comportamento, foi realizada uma modificação no núcleo C++ do PyVRP, consolidando em um único tempo de serviço todos os nós que compartilham a

mesma localização de *locker* quando aparecem como visitas consecutivas em uma rota. Assim, um bloco de visitas ao mesmo armário incorre em apenas um tempo de serviço (caso o veículo retorne ao mesmo armário em outro momento da rota, o tempo é adicionado novamente, pois ocorreria uma segunda parada física). Essa adaptação garante que o modelo represente de forma realista o processo de descarregamento em armários, evitando a superestimação dos tempos de atendimento.

4.4 MECANISMOS EVOLUTIVOS

Para assegurar a evolução da população em direção a soluções de alta qualidade, o algoritmo emprega um conjunto coordenado de operadores genéticos. As subseções a seguir detalham três etapas fundamentais desse ciclo evolutivo: o mecanismo de seleção dos progenitores, o operador de recombinação responsável pela geração de descendentes e a estratégia de reparação utilizada para recuperar indivíduos inviáveis e reinseri-los no processo de busca.

4.4.1 Operador de Seleção

A seleção de pais no algoritmo é realizada por meio de torneios binários. Nesse esquema, cada pai é escolhido em duas etapas: (i) sorteiam-se duas soluções da população, misturando indivíduos viáveis e inviáveis; (ii) compara-se o valor de *fitness* de ambas e a solução com melhor desempenho é declarada vencedora do torneio e selecionada como pai. Esse procedimento introduz uma pressão seletiva moderada, pois soluções de alta qualidade têm maior probabilidade de serem escolhidas, mas indivíduos medianos ainda podem vencer torneios ocasionais, preservando diversidade genética.

No algoritmo, o *fitness* utilizado é um *biased fitness* que combina custo penalizado e diversidade dentro de cada subpopulação. Além disso, a seleção do segundo pai inclui uma verificação adicional de distância estrutural em relação ao primeiro, que é medida pela métrica de *broken pairs*: o algoritmo tenta, por algumas tentativas, escolher um segundo pai cuja distância fique dentro de um intervalo desejado. Isso evita que cruzamentos ocorram entre soluções quase idênticas ou excessivamente distintas, contribuindo para

um melhor equilíbrio entre intensificação e diversificação.

4.4.2 Recombinação por SREX

O operador de recombinação adotado no algoritmo é o *Selective Route Exchange* (SREX), originalmente proposto por Nagata et al. (22). Diferentemente de operadores tradicionais que realizam trocas pontuais de clientes, o SREX atua em nível de rotas completas, selecionando subconjuntos de rotas de uma solução parental e transferindo-os diretamente para o descendente. Essa estratégia permite preservar sequências de visitas espacialmente coesas, que frequentemente correspondem a estruturas de boa qualidade no espaço de busca.

Após a troca dos blocos de rotas selecionados, o SREX constrói um único descendente a partir de uma combinação desses blocos com as rotas remanescentes dos pais, buscando preservar ao máximo as subestruturas de menor custo penalizado. Como as operações são realizadas diretamente sobre rotas completas, alguns clientes podem temporariamente deixar de ser atendidos, de modo que a reparação dessas lacunas e o refinamento da solução ficam a cargo das etapas subsequentes de busca local. Para uma descrição formal detalhada sobre a mecânica de identificação dessas subestruturas e a complexidade do operador, recomenda-se consultar o trabalho de Nagata et al.(22).

Ao operar diretamente sobre rotas completas, o SREX evita a destruição excessiva da estrutura das soluções parentais e gera descendentes já próximos de regiões promissoras do espaço de busca. Essa característica torna o SREX particularmente adequado para problemas em que a organização espacial das rotas é relevante, como o VRPPL. No contexto deste trabalho, o operador integra-se de forma natural ao modelo com clientes híbridos, nos quais a escolha entre atendimento domiciliar e via *locker* influencia diretamente a composição e a estrutura das rotas.

4.4.3 Reparação

No PyVRP, a reparação não é um módulo isolado que atua antes da busca local, mas uma segunda fase de melhoria que ocorre após uma busca local inicial. Em cada iteração

do algoritmo genético, dois pais são selecionados e recombinados pelo operador SREX, produzindo um descendente. Este descendente passa primeiramente por uma busca local granular utilizando o avaliador de custo padrão, o que já pode, em muitos casos, eliminar violações e melhorar substancialmente a solução.

Se, após essa primeira etapa de busca local, a solução ainda for inviável, o algoritmo pode acionar um mecanismo de reparação com certa probabilidade pré-definida por um parâmetro. Nessa segunda fase, aplica-se novamente a busca local, mas agora utilizando um avaliador de custo com penalidades temporariamente amplificadas (*repair booster*). Esse avaliador aumenta os pesos associados a violações de capacidade, janelas de tempo e distância máxima, tornando movimentos que reduzem essas violações muito mais atraentes.

O efeito prático é que, durante a reparação, a busca local passa a priorizar movimentos que removem clientes de rotas sobrecarregadas, realocam atendimentos que geram grandes atrasos ou corrigem grupos inconsistentes (por exemplo, clientes híbridos atendidos simultaneamente em casa e no locker). Caso a solução reparada se torne viável, ela é reinserida na população como um novo indivíduo competitivo. Dessa forma, a reparação atua como uma etapa de intensificação focada na recuperação de soluções promissoras, aproximando-as da região viável sem abrir mão da diversidade gerada pelo crossover.

4.5 BUSCA LOCAL

A busca local do PyVRP é responsável pela maior parte da melhoria das soluções no VRPPL e concentra o esforço computacional do algoritmo. Em cada chamada, ela atua sobre uma solução completa, composta por todas as rotas da frota, e executa iterativamente duas fases: uma fase de movimentos em nível de clientes e uma fase de movimentos em nível de rotas. Essas fases são repetidas até que não se encontre mais nenhum movimento que reduza o custo penalizado, caracterizando um ótimo local para aquela solução.

Na fase de clientes, o algoritmo percorre os vértices de atendimento em uma ordem aleatória. Quando o cliente considerado pertence a um grupo mutuamente exclusivo

(caso dos clientes HLC, que podem ser atendidos em casa ou em um *locker*), a busca local aplica primeiro um mecanismo específico de grupos. Esse mecanismo garante que exista exatamente uma alternativa ativa de cada grupo obrigatório na solução e pode trocar, na mesma posição de rota, a alternativa atualmente utilizada (domicílio ou *locker*) por outra mais vantajosa, desde que isso reduza o custo penalizado, que incorpora distância percorrida, capacidade dos veículos, janelas de tempo e o tratamento consolidado do tempo de serviço nos armários.

Em seguida, para esse mesmo cliente, são avaliados movimentos envolvendo outros clientes considerados "próximos" em termos espaciais e temporais. Essa proximidade é pré-computada a partir de informações de distância, tempos de viagem e janelas de tempo, de modo que cada cliente interaja apenas com um subconjunto reduzido de vizinhos relevantes. Para cada par de clientes assim formado, são testados movimentos clássicos de roteamento, tanto dentro da mesma rota quanto entre rotas diferentes, e aplica-se qualquer movimento que produza uma melhora imediata no custo penalizado. Essa fase é, portanto, uma busca local granular em torno da solução corrente, que explora apenas vizinhanças promissoras.

Na fase de rotas, o algoritmo passa a considerar pares de rotas completos, avaliando movimentos que transferem ou reagrupam subconjuntos de clientes entre veículos. Esses movimentos têm efeito mais global, pois podem alterar significativamente a repartição de carga e a distribuição de atendimentos domiciliares e em *lockers* ao longo da frota. Tal como na fase de clientes, somente são aceitos movimentos que reduzam o custo penalizado, e as duas fases se alternam até que nenhuma modificação adicional traga benefício, resultando em uma solução localmente estável para o VRPPL.

A eficácia da busca local depende diretamente do conjunto de movimentos de vizinhança disponíveis para explorar o espaço de soluções. Esses movimentos são classificados de acordo com seu escopo de atuação e impacto na estrutura da solução. As próximas subseções descrevem essas duas categorias principais: os operadores de nós, focados em refinamentos granulares e ajustes finos na sequência de visitas, e os operadores de rotas, responsáveis por alterações estruturais mais amplas nas soluções.

4.5.1 Operadores de Nós

Os operadores de nós constituem o principal mecanismo de refinamento fino das rotas durante a busca local. Esses operadores atuam sobre clientes individuais ou pequenos blocos de clientes, promovendo ajustes incrementais na sequência de visitas, tanto dentro de uma mesma rota quanto entre rotas distintas. No contexto do VRPPL tratado neste trabalho, são empregados operadores clássicos amplamente consolidados na literatura de roteamento de veículos.

Entre os principais operadores utilizados, destacam-se:

- *Relocate*: remove um cliente de sua posição atual e o reinsere em outra posição potencialmente mais promissora, podendo ocorrer na mesma rota ou em uma rota diferente;
- *Swap*: troca a posição de dois clientes pertencentes à mesma rota ou a rotas distintas;
- *Exchange*: realiza trocas envolvendo pequenos blocos de clientes, permitindo rearranjos mais expressivos do que a troca simples ponto a ponto;
- *2-opt*: operações equivalentes à troca de caudas de rota, nas quais segmentos finais de duas rotas são permutados, alterando de forma mais significativa a estrutura do trajeto.

Esses movimentos são avaliados por meio da função de custo penalizado, que considera distância percorrida, tempos de viagem, tempos de serviço, violações de capacidade, atrasos em janelas de tempo (*time warp*) e penalidades associadas a grupos de clientes. A ordem de aplicação dos operadores e de varredura dos clientes e rotas é embaralhada a cada iteração da busca local, evitando um padrão fixo e favorecendo a exploração do espaço de soluções. Apenas movimentos que resultam em melhoria do custo penalizado são aceitos, caracterizando uma busca local do tipo *first improvement*.

Além desses operadores clássicos, foi implementado neste trabalho um operador adicional específico ao contexto do VRPPL, denominado *locker reducer*. Esse operador atua exclusivamente no interior de uma mesma rota e explora uma particularidade do

problema: a possibilidade de múltiplos clientes estarem associados ao mesmo armário de encomendas.

Em determinadas soluções, uma rota pode conter múltiplas visitas ao mesmo locker, separadas por outros atendimentos intermediários. O operador *locker reducer* tenta consolidar essas visitas, reunindo em um único ponto da rota todos os clientes associados ao mesmo armário, sempre que esse rearranjo reduz o custo penalizado da solução, considerando conjuntamente custo de deslocamento, tempos de serviço e penalidades associadas a violações de capacidade e janelas de tempo. Dessa forma, o operador busca reduzir redundâncias no atendimento a armários, além de consolidar um único tempo de serviço na visita, explorando de maneira mais eficiente a consolidação de entregas, característica central do VRPPL.

Um exemplo ilustrativo do funcionamento desse operador é apresentado na Figura 4.2. Na parte superior da figura, a rota contém duas visitas ao mesmo *locker* L_A separadas por atendimentos domiciliares H_i intermediários. Ao ser aplicado, o *lockerReducer* identifica esses dois blocos de visitas ao mesmo local físico e desloca o bloco mais à direita para imediatamente após o bloco mais à esquerda, preservando a ordem interna de ambos.

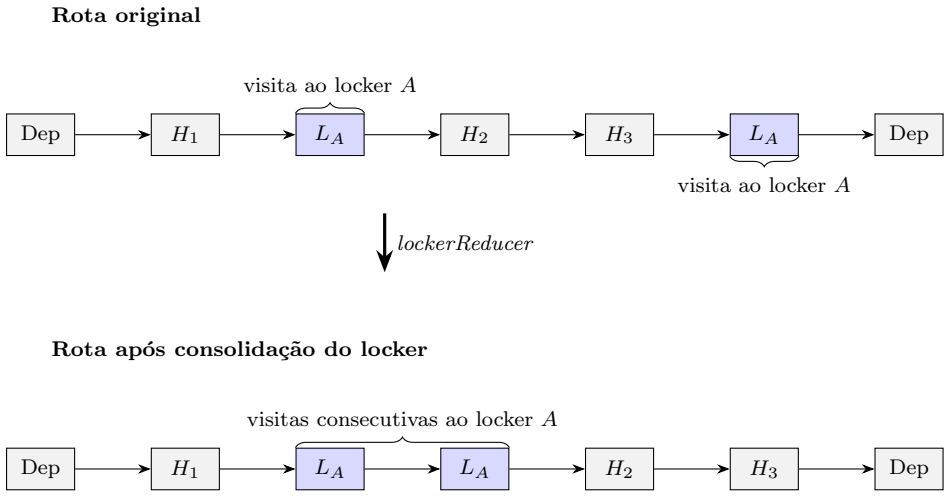


Figura 4.2: Funcionamento do operador *lockerReducer*.
Fonte: Elaborado pelo autor.

Esse operador não substitui os movimentos clássicos, mas os complementa, introduzindo um mecanismo direcionado à estrutura específica do problema tratado. Sua aplicação ocorre no mesmo ciclo de avaliação incremental da busca local, sendo aceito

apenas quando promove uma melhoria efetiva da solução.

4.5.2 Operadores de rotas

Os operadores de rotas atuam em pares de rotas completos e têm impacto mais global sobre a solução. Movimentos desse tipo podem, por exemplo, trocar blocos de clientes entre veículos, realocar sequências inteiras de visita ou reconfigurar, de forma coordenada, duas rotas que atendem regiões próximas.

Neste trabalho, são empregados dois operadores de rotas: *SwapRoutes*, que troca rotas ou grandes segmentos de rotas entre veículos de forma direta, e *SWAP**, tal como proposto por Vidal (21) e aqui considerado no contexto de janelas de tempo. Esse operador reavalia simultaneamente múltiplas posições possíveis para clientes pertencentes a rotas distintas, permitindo rearranjos mais expressivos do que simples trocas par-a-par. No contexto do VRPPL, esses operadores são particularmente úteis para redistribuir atendimentos domiciliares e entregas via *lockers* entre os veículos, explorando diferentes padrões de consolidação de entregas sem violar a consistência dos grupos mutuamente exclusivos. Mais detalhes do SWAP* podem ser encontrados em (21).

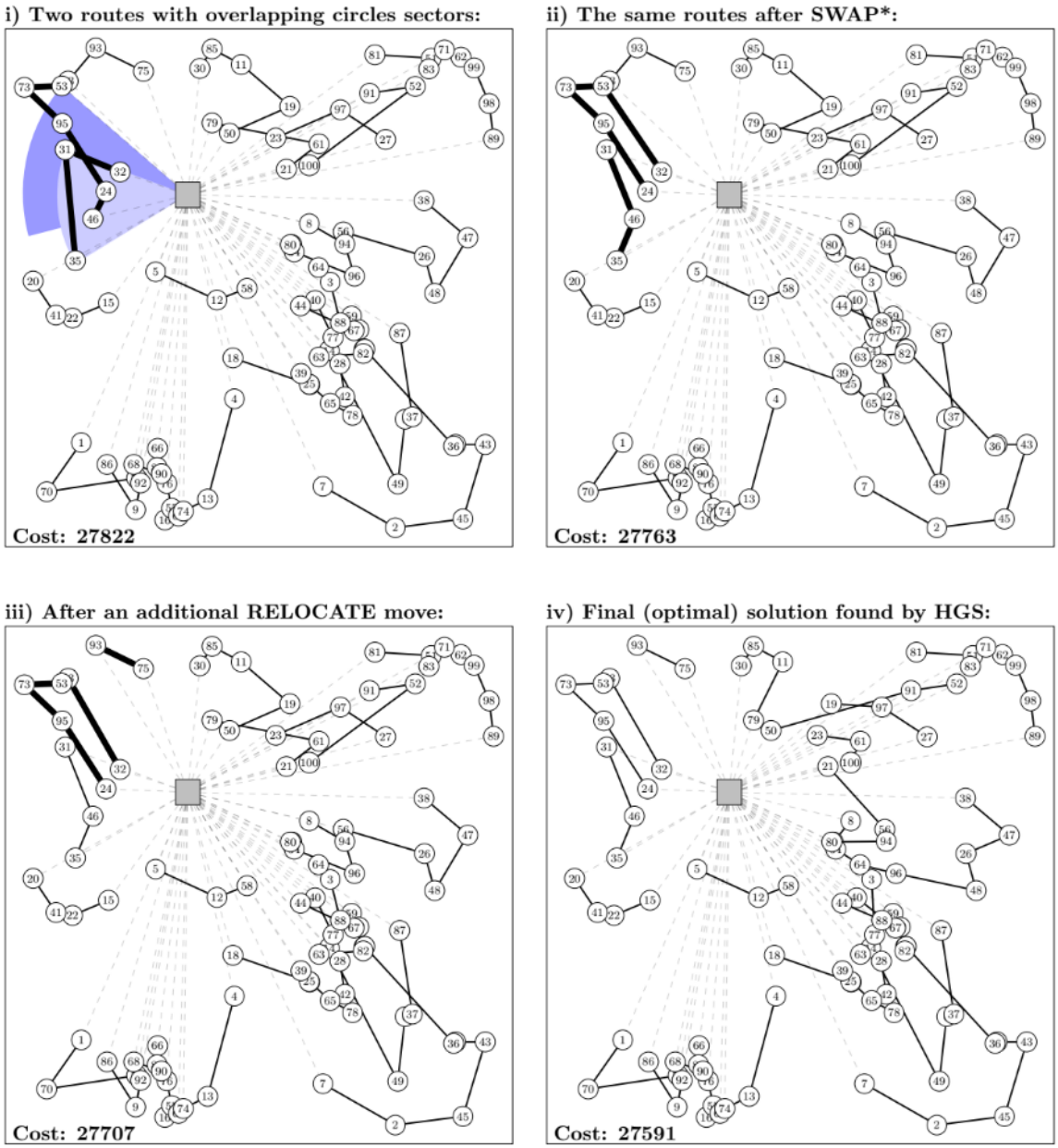
A Figura 4.3 ilustra o funcionamento clássico desse operador em quatro etapas: (i) Duas rotas com setores angulares sobrepostos; (ii) solução após a aplicação do operador *SWAP**; (iii) refinamento adicional com um movimento de *relocate*; (iv) solução final obtida pelo HGS.

4.6 CONTROLE DE DIVERSIDADE

Os pesos das penalidades de capacidade e *time warp* são ajustados dinamicamente para manter um equilíbrio entre soluções viáveis e inviáveis. Se restrições são violadas com frequência, o peso correspondente aumenta; caso contrário, é reduzido. Esse mecanismo, inspirado no HGS clássico e detalhado na Seção 4.2, permite ao algoritmo alternar entre exploração (aceitando temporariamente violações moderadas) e intensificação (privilegiando soluções estritamente viáveis) conforme a necessidade.

O controle de diversidade, por sua vez, utiliza uma métrica de distância baseada em

Figura 4.3: Funcionamento do operador *SWAP**.



Fonte: Vidal (21).

"pares quebrados" (*broken pairs*), estimulando a manutenção de soluções estruturalmente distintas na população. Isso reduz o risco de convergência prematura, especialmente em instâncias densas ou com forte concentração geográfica.

Métrica de diversidade: *broken-pairs distance*

A diversidade estrutural entre soluções no HGS é medida pela distância de *broken pairs*, que compara duas soluções por meio das adjacências presentes em suas rotas. Seja N o

conjunto de nós do problema. Para cada localização $v \in N$, consideram-se seu antecessor $\text{pred}_S(v)$ e seu sucessor $\text{succ}_S(v)$ em uma solução S, isto é, os nós imediatamente anterior e posterior a v na rota. Sempre que uma dessas adjacências difere entre duas soluções, diz-se que ocorre um "par quebrado".

A distância pode ser expressa como

$$d_{\text{BP}}\big(S^{(1)}, S^{(2)}\big) = \frac{1}{2|N|} \sum_{v \in N} \Big(\mathbb{I}[\text{pred}_{S^{(1)}}(v) \neq \text{pred}_{S^{(2)}}(v)] + \mathbb{I}[\text{succ}_{S^{(1)}}(v) \neq \text{succ}_{S^{(2)}}(v)] \Big), \tag{4.2}$$

onde $\mathbb{I}[\cdot]$ é a função indicadora. Valores próximos de 0 indicam soluções praticamente idênticas em estrutura de rotas, enquanto valores próximos de 1 indicam soluções altamente distintas.

Essa métrica é utilizada tanto no cálculo do *biased fitness* quanto na seleção do segundo pai no torneio binário, que tenta escolher soluções cuja distância estrutural esteja dentro de um intervalo desejado. Isso evita tanto cruzamentos entre soluções quase idênticas quanto entre soluções excessivamente diferentes, fornecendo um bom equilíbrio entre intensificação e diversificação.

4.7 CRITÉRIO DE PARADA

O algoritmo é executado até que seja atingido um número máximo de iterações consecutivas sem melhoria da melhor solução conhecida. Diferentemente de um limite fixo de iterações, adotou-se neste trabalho um critério de parada proporcional ao tamanho da instância, definido como $100 \times n$, em que n representa o número de clientes da instância.

Esse critério visa tornar o esforço computacional mais equilibrado entre instâncias de diferentes portes. Em instâncias menores, evita-se a execução excessiva após a convergência, enquanto em instâncias maiores o algoritmo dispõe de um número suficiente de iterações para explorar adequadamente o espaço de soluções. Esse ajuste contribui para uma comparação mais justa dos resultados experimentais, especialmente em termos de tempo de execução e qualidade das soluções.

4.8 SÍNTESE DA ADAPTAÇÃO PROPOSTA

Este capítulo detalhou a estrutura do algoritmo evolutivo híbrido utilizado. O método fundamenta-se na arquitetura eficiente do PyVRP, cuja versatilidade permitiu modelar a natureza dual dos clientes híbridos (HLC) através do recurso nativo de grupos mutuamente exclusivos, sem a necessidade de reestruturação do núcleo de otimização. Adicionalmente, a estrutura do *framework* facilitou a parametrização do algoritmo, permitindo a realização de ajustes finos que colaboraram para o desempenho da busca.

No entanto, para capturar plenamente as especificidades operacionais do VRPPL, a ferramenta foi estendida com contribuições desenvolvidas especificamente neste trabalho, como a implementação do operador *lockerReducer* e a modificação no núcleo de avaliação para a consolidação de tempos de serviço. Essa combinação entre uma estratégia de modelagem que explora funcionalidades existentes e o desenvolvimento de mecanismos dedicados demonstra a adequação da abordagem proposta para solucionar o problema, equilibrando os custos logísticos entre o atendimento domiciliar e o uso consolidado de armários de encomendas.

5 EXPERIMENTOS COMPUTACIONAIS

Este capítulo apresenta os experimentos computacionais realizados para avaliar o desempenho do algoritmo proposto para o VRPPL. Os resultados obtidos são comparados com abordagens de referência da literatura, em especial o método baseado em *Simulated Annealing* (SA) proposto por Yu et al. (7) e o algoritmo *Iterated Greedy* (IG) apresentado por Corrêa et al. (8). A análise contempla tanto a qualidade das soluções obtidas quanto o esforço computacional necessário, permitindo avaliar o equilíbrio entre custo e tempo de execução.

5.1 INSTÂNCIAS DE TESTE

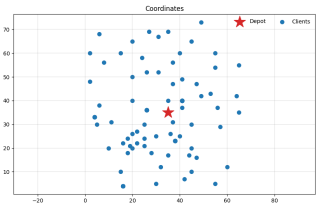
Os experimentos foram conduzidos utilizando as instâncias propostas por Yu et al. (7), que constituem, até o momento, o principal benchmark público para o VRPPL nesta modelagem apresentada. Essas instâncias são derivadas das clássicas instâncias de Solomon para o VRPTW e adaptadas para incorporar armários de encomendas e tipos de clientes (HC, LC e HLC).

As instâncias são classificadas em três categorias, de acordo com a distribuição espacial dos clientes:

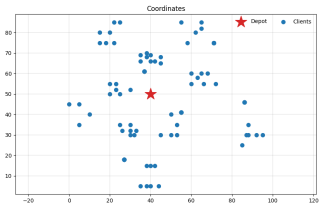
- **C**: clientes agrupados geograficamente;
- **R**: clientes distribuídos de forma aleatória;
- **RC**: combinação de padrões agrupados e aleatórios.

Além disso, são considerados três tamanhos de instância, com $n = 25$, $n = 50$ e $n = 100$ clientes, totalizando um conjunto diversificado de cenários logísticos. Para ilustrar essas diferenças, na Figura 5.1 são apresentadas figuras ilustrativas da distribuição espacial dos clientes para instâncias do tipo C, R e RC com 100 clientes.

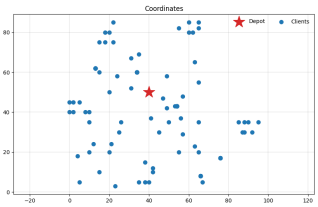
Figura 5.1: Exemplos de instâncias com 100 clientes e diferentes distribuições espaciais.



(a) Instância randomizada - R



(b) Instância clusterizada - C



(c) Instância randomizada clusterizada - RC

Fonte: Elaborado pelo autor.

5.2 CONFIGURAÇÃO EXPERIMENTAL

Todos os experimentos foram executados em um computador com sistema operacional Windows 10 Pro, utilizando o ambiente *Windows Subsystem for Linux* (WSL 2) com a distribuição Ubuntu 22.04 LTS. O *hardware* dispõe de um processador Intel Core i5-13600K e 32 GB de memória RAM. O algoritmo proposto foi implementado em Python 3.12, utilizando a biblioteca PyVRP versão 0.12.1.

Para garantir uma comparação justa com a literatura, considerou-se as diferenças de hardware entre o ambiente utilizado neste trabalho e o utilizado por (7) (Intel Core i7-4770). Utilizando o *benchmark PassMark CPU Mark (Single Thread)*[1], verifica-se que o processador i5-13600K possui uma pontuação de aproximadamente 4.120, enquanto o i7-4770 possui 2.170, resultando em uma razão de desempenho de aproximadamente 1,9.

Dessa forma, os tempos de execução do algoritmo proposto (HGS) apresentados nas tabelas a seguir foram normalizados (multiplicados por um fator de 1,90) para simular o desempenho equivalente na máquina de referência da literatura. Vale ressaltar que os resultados do algoritmo IG (8) também foram reportados seguindo essa mesma metodologia de normalização em relação ao trabalho de (7). Portanto, as comparações de tempo apresentadas neste capítulo refletem um cenário equalizado para a arquitetura do processador Intel Core i7-4770.

O critério de parada adotado foi ajustado de acordo com o tamanho da instância, definido como $100 \times n$ iterações sem melhoria da melhor solução conhecida. Cada instância foi executada 30 vezes com sementes (*seeds*) distintas.

[1]Dados obtidos em: `https://www.cpubenchmark.net/`. Acesso em: dezembro de 2025.

Dado o limite de tempo para a finalização do trabalho, não foi empregado um procedimento automático de calibração de parâmetros (como o IRACE(23)). Em substituição, a definição das configurações foi realizada de forma empírica, fundamentada em testes exploratórios preliminares e na observação do comportamento de convergência do método nas instâncias analisadas.

Essa abordagem manual priorizou o ajuste dos parâmetros associados às penalidades adaptativas, que foram modificados em relação aos valores padrão do PyVRP para favorecer uma exploração mais ampla do espaço de busca. Optou-se por reduzir a agressividade do aumento das penalidades e limitar seus valores máximos, permitindo que soluções temporariamente inviáveis permaneçam ativas por mais tempo na população. Tal estratégia mostrou-se adequada à estrutura combinatória do VRPPL, na qual a rigidez excessiva das penalidades tende a dificultar a alternância entre modos de atendimento (domiciliar ou armário) e a superação de ótimos locais.

Para garantir a reprodutibilidade dos experimentos, a Tabela 5.1 detalha os valores exatos configurados. Nota-se que os parâmetros estruturais da população foram mantidos conforme o padrão do método HGS original, enquanto os controles de penalidade refletem a calibração empírica descrita.

5.3 RESULTADOS E ANÁLISE

As Tabelas 5.2, 5.3 e 5.4 apresentam os resultados obtidos para instâncias pequenas ($n = 25$), médias ($n = 50$) e grandes ($n = 100$), respectivamente. Para cada instância, são reportados o número de veículos (NV), o tempo médio de execução, o custo médio e o melhor custo obtidos pelo HGS, bem como os valores correspondentes reportados pelo SA (7) e pelo IG (8).Os *gaps* percentuais apresentados referem-se à comparação entre o melhor custo obtido pelo método proposto e o melhor custo reportado por cada abordagem de referência.

O *gap* é definido como:

$$\text{GAP}(\%) = \frac{C_{\text{HGS}} - C_{\text{ref}}}{C_{\text{ref}}} \times 100, \tag{5.1}$$

Tabela 5.1: Parâmetros configurados para o algoritmo HGS proposto.

Parâmetro	**Valor Padrão**	**Valor Adotado**	**Efeito**
Estrutura da População			
Tam. Mín. População	25	**25**	Mantido padrão da literatura.
Tamanho da Geração	40	**40**	Mantido padrão da literatura.
Penalidades Adaptativas			
Penalty Increase	1.34	**1.10**	Aumenta penalidade mais lentamente.
Penalty Decrease	0.32	**0.60**	Reduz penalidade mais rapidamente.
Target Feasible	0.43	**0.30**	Aceita maior proporção de inviáveis.
Repair Booster	12	**8**	Menor intensidade no reparo imediato.
Max Penalty	100.000	**10.000**	Limita o custo artificial máximo.
Min Penalty	0.10	**0.01**	Permite penalidades residuais menores.

onde C_{HGS} representa o melhor custo obtido pelo método proposto e C_{ref} corresponde ao melhor custo obtido pelo método de referência (SA ou IG). Valores positivos indicam soluções de maior custo em relação à referência, enquanto valores negativos indicam soluções melhores.

De maneira geral, observa-se que o algoritmo proposto apresenta desempenho competitivo em todas as classes de instância analisadas. Para as instâncias pequenas ($n = 25$), detalhadas na Tabela 5.2, os resultados indicam uma performance muito próxima às abordagens de referência, registrando um *gap* médio de apenas 0,38% em relação ao SA e 0,39% em relação ao IG. Em termos de eficiência computacional, o tempo médio de 5,95 segundos posiciona o método proposto de forma vantajosa em relação ao SA (8,09 segundos), embora superior ao tempo residual reportado pelo IG (0,004 segundos). Apesar dessa consistência geral, o HGS proposto tendeu a custos ligeiramente superiores em casos pontuais, destacando-se a instância R205, na qual o *gap* em relação à literatura atingiu 4,75%, indicando uma dificuldade específica para atingir os resultados de referência neste cenário.

Nas instâncias médias ($n = 50$), detalhadas na Tabela 5.3, o método proposto mantém

Tabela 5.2: Resultados computacionais para instâncias pequenas ($n = 25$).

	SA				IG				HGS				GAP (%)	
Instância	Média	Tempo(s)	Melhor	NV	Média	Tempo(s)	Melhor	NV	Média	Tempo(s)	Melhor	NV	SA	IG
C101	199.800	7.600	**199.800**	3	199.800	0.004	**199.800**	3	200.205	5.830	200.205	3	0.20	0.20
C102	155.080	19.100	**143.500**	3	143.567	0.006	**143.500**	3	143.952	9.290	143.952	3	0.31	0.31
C103	203.900	10.400	**203.900**	3	203.900	0.005	**203.900**	3	204.406	7.200	204.406	3	0.25	0.25
C104	181.800	11.600	**181.800**	3	187.060	0.005	182.100	3	185.656	11.930	183.937	3	1.18	1.01
C105	206.900	9.100	**206.900**	3	206.900	0.005	**206.900**	3	210.236	9.540	210.236	3	1.61	1.61
C106	208.800	7.500	**208.800**	3	208.800	0.006	**208.800**	3	210.718	10.170	210.254	3	0.70	0.70
C107	206.900	8.900	**206.900**	3	206.900	0.005	**206.900**	3	209.979	11.650	209.876	3	1.44	1.44
C108	159.380	8.600	**155.300**	3	155.300	0.004	**155.300**	3	155.930	8.020	155.930	3	0.41	0.41
C109	143.460	11.400	141.500	3	140.967	0.005	**140.500**	3	140.844	6.080	140.844	3	-0.46	0.24
C201	195.500	5.400	**195.500**	2	195.500	0.005	**195.500**	2	196.172	4.960	196.172	2	0.34	0.34
C202	119.600	8.800	**119.600**	1	119.600	0.002	**119.600**	1	119.868	5.760	119.868	1	0.22	0.22
C203	146.400	9.500	**146.400**	1	146.400	0.003	**146.400**	1	146.855	5.430	146.855	1	0.31	0.31
C204	145.900	6.500	**145.900**	1	145.900	0.004	**145.900**	1	146.322	4.070	146.322	1	0.29	0.29
C205	168.200	8.200	**168.200**	2	168.200	0.004	**168.200**	2	168.813	4.260	168.813	2	0.36	0.36
C206	171.300	7.400	**171.300**	2	171.300	0.004	**171.300**	2	171.731	6.140	171.731	2	0.25	0.25
C207	121.300	11.900	**121.300**	1	121.300	0.004	**121.300**	1	121.731	4.280	121.731	1	0.36	0.36
C208	162.200	7.400	**162.200**	1	162.200	0.004	**162.200**	1	162.553	5.680	162.553	1	0.22	0.22
R101	178.600	6.400	**178.600**	2	178.600	0.003	**178.600**	2	178.845	6.120	178.845	2	0.14	0.14
R102	187.800	8.100	**187.800**	2	187.800	0.003	**187.800**	2	188.066	5.780	188.066	2	0.14	0.14
R103	197.600	4.800	**197.600**	2	197.600	0.003	**197.600**	2	197.987	4.980	197.987	2	0.20	0.20
R104	238.900	9.000	238.940	3	238.900	0.006	**238.900**	3	239.513	5.700	239.513	3	0.24	0.26
R105	194.100	6.700	**194.100**	2	194.100	0.003	**194.100**	2	194.491	5.550	194.491	2	0.20	0.20
R106	177.000	5.800	**177.000**	2	177.000	0.004	**177.000**	2	177.303	4.350	177.303	2	0.17	0.17
R107	238.900	8.700	**238.900**	3	238.900	0.005	**238.900**	3	239.513	6.780	239.513	3	0.26	0.26
R108	251.280	10.500	**250.800**	2	251.200	0.005	**250.800**	2	252.491	6.900	252.491	2	0.67	0.67
R109	260.700	7.900	**260.700**	3	260.700	0.004	**260.700**	3	261.564	7.240	261.564	3	0.33	0.33
R110	239.200	9.400	**239.200**	3	239.200	0.005	**239.200**	3	239.798	6.140	239.798	3	0.25	0.25
R111	245.300	5.800	**245.300**	2	245.300	0.004	**245.300**	2	246.858	5.950	246.007	2	0.29	0.29
R112	185.600	6.600	**185.600**	2	185.600	0.003	**185.600**	2	186.087	4.770	186.087	2	0.26	0.26
R201	304.700	5.600	**304.700**	3	304.700	0.004	**304.700**	3	305.194	5.340	305.194	3	0.16	0.16
R202	265.320	8.600	**257.800**	2	257.800	0.005	**257.800**	2	258.404	5.680	258.404	2	0.23	0.23
R203	185.600	7.000	**185.600**	2	185.600	0.004	**185.600**	2	186.010	4.620	186.010	2	0.22	0.22
R204	154.200	6.300	**154.200**	1	154.200	0.004	**154.200**	1	154.567	4.260	154.567	1	0.24	0.24
R205	225.520	10.000	**219.100**	1	219.100	0.005	**219.100**	1	229.505	5.240	229.505	1	4.75	4.75
R206	195.800	7.900	**195.800**	1	195.800	0.004	**195.800**	1	196.136	3.820	196.136	1	0.17	0.17
R207	197.960	7.800	**197.900**	1	197.900	0.004	**197.900**	1	198.278	3.710	198.278	1	0.19	0.19
R208	163.800	5.900	**163.800**	1	163.800	0.003	**163.800**	1	164.017	3.710	164.017	1	0.13	0.13
R209	232.600	8.100	**232.600**	1	232.600	0.004	**232.600**	1	233.122	3.710	233.122	1	0.22	0.22
R210	279.100	6.300	279.180	2	279.100	0.004	**279.100**	2	279.627	4.770	279.627	2	0.16	0.19
R211	131.500	6.400	**131.500**	1	131.500	0.003	**131.500**	1	131.777	4.430	131.777	1	0.21	0.21
RC101	332.600	5.400	**332.600**	3	332.600	0.004	**332.600**	3	333.153	4.710	333.153	3	0.17	0.17
RC102	297.900	9.100	**297.900**	3	297.900	0.004	**297.900**	3	298.514	7.750	298.499	3	0.20	0.20
RC103	243.800	6.200	**243.800**	3	243.800	0.003	**243.800**	3	244.200	5.660	244.200	3	0.16	0.16
RC104	309.400	12.600	**309.400**	3	309.640	0.005	**309.400**	3	309.931	5.990	309.931	3	0.17	0.17
RC105	309.700	4.700	**309.700**	3	309.700	0.003	**309.700**	3	310.292	7.180	310.292	3	0.19	0.19
RC106	279.100	7.800	**279.100**	4	279.880	0.005	**279.100**	4	279.569	7.810	279.539	4	0.16	0.16
RC107	300.560	7.900	**299.900**	3	299.900	0.005	**299.900**	3	300.659	6.350	300.659	3	0.25	0.25
RC108	275.700	6.400	**275.700**	3	275.700	0.004	**275.700**	3	276.238	5.970	276.238	3	0.20	0.20
RC201	270.560	8.100	**270.200**	2	270.207	0.006	**270.200**	2	270.765	6.420	270.765	2	0.21	0.21
RC202	294.620	13.100	**293.900**	3	293.900	0.006	**293.900**	3	294.638	6.060	294.638	3	0.25	0.25
RC203	208.000	5.800	**208.000**	1	208.000	0.004	**208.000**	1	208.391	4.660	208.391	1	0.19	0.19
RC204	175.700	6.000	**175.700**	1	175.700	0.003	**175.700**	1	175.928	3.480	175.928	1	0.13	0.13
RC205	284.480	5.600	**283.200**	3	283.200	0.005	**283.200**	3	284.071	6.330	284.071	3	0.31	0.31
RC206	247.400	11.400	**246.700**	2	246.700	0.005	**246.700**	2	247.482	5.760	247.482	2	0.32	0.32
RC207	250.500	8.100	**250.500**	2	250.500	0.004	**250.500**	2	251.122	5.190	251.122	2	0.25	0.25
RC208	182.300	6.100	**182.300**	1	182.300	0.003	**182.300**	1	182.773	4.200	182.773	1	0.26	0.26
Média		8.093				0.004				5.950			0.38	0.39

Tabela 5.3: Resultados computacionais para instâncias médias ($n = 50$).

	SA				IG				HGS				GAP (%)	
Instância	Média	Tempo(s)	Melhor	NV	Média	Tempo(s)	Melhor	NV	Média	Tempo(s)	Melhor	NV	SA	IG
C101	371.240	206.300	368.500	5	371.760	0.052	366.800	5	366.893	57.910	**362.376**	5	-1.66	-1.21
C102	359.080	184.100	357.700	5	357.303	0.050	**351.200**	5	355.629	41.000	355.130	5	-0.72	1.12
C103	323.920	228.900	317.900	5	313.103	0.040	**306.900**	5	308.941	40.110	307.825	5	-3.17	0.30
C104	348.240	233.700	344.900	5	343.270	0.048	341.700	5	343.349	76.110	**341.657**	5	-0.94	-0.01
C105	376.740	205.100	374.900	5	369.473	0.044	**349.500**	5	352.159	66.310	351.837	5	-6.15	0.67
C106	337.060	234.500	321.900	5	327.387	0.038	**321.900**	5	322.961	31.520	322.961	5	0.33	0.33
C107	330.560	181.500	**327.300**	5	329.730	0.039	**327.300**	5	328.139	49.480	327.985	5	0.21	0.21
C108	339.760	222.800	338.800	5	338.730	0.041	**338.500**	5	339.528	36.160	339.109	5	0.09	0.18
C109	341.380	252.900	335.400	5	330.027	0.051	**329.000**	5	329.808	21.680	329.808	5	-1.67	0.25
C201	283.740	173.800	**282.200**	2	282.200	0.032	**282.200**	2	283.973	28.390	283.025	2	0.29	0.29
C202	242.020	188.100	**227.600**	2	233.500	0.027	**227.600**	2	228.425	13.430	228.425	2	0.36	0.36
C203	270.700	123.900	**257.800**	2	264.500	0.024	**257.800**	2	258.441	25.250	258.441	2	0.25	0.25
C204	277.140	150.100	**274.400**	2	274.423	0.048	**274.400**	2	275.158	19.060	275.158	2	0.28	0.28
C205	268.000	148.000	**266.100**	2	267.000	0.023	**266.100**	2	266.570	28.390	266.570	2	0.18	0.18
C206	236.140	159.600	235.000	2	234.600	0.025	**234.600**	2	235.204	21.830	235.204	2	0.09	0.26
C207	288.700	145.900	**288.700**	2	288.703	0.041	**288.700**	2	289.838	21.810	289.823	2	0.39	0.39
C208	288.100	148.000	288.100	3	285.300	0.036	**285.300**	2	286.144	16.260	286.144	2	-0.68	0.30
R101	455.360	206.400	**449.400**	5	453.500	0.033	**449.400**	5	456.063	19.340	456.063	6	1.48	1.48
R102	423.320	161.100	423.200	5	422.910	0.046	**422.900**	5	427.471	32.090	427.471	5	1.01	1.08
R103	326.320	144.800	320.500	4	318.260	0.035	**316.600**	4	319.692	26.450	317.750	4	-0.86	0.36
R104	377.840	171.700	377.800	4	378.603	0.044	377.800	4	379.106	26.280	**377.041**	4	-0.20	-0.20
R105	414.460	171.500	**413.500**	5	413.500	0.031	**413.500**	5	420.826	45.300	420.638	5	1.73	1.73
R106	450.700	206.400	448.100	5	449.347	0.035	**447.900**	5	449.305	51.680	449.041	5	0.21	0.25
R107	391.320	189.100	389.000	4	389.163	0.049	**388.700**	5	391.483	43.830	389.499	5	0.13	0.21
R108	443.960	184.700	429.600	4	444.053	0.062	**426.500**	4	432.490	25.180	432.151	4	0.59	1.32
R109	486.560	163.700	479.700	5	479.640	0.054	**475.300**	4	495.607	20.810	494.684	5	3.12	4.08
R110	396.800	152.800	**396.100**	4	396.253	0.038	**396.100**	4	399.743	42.750	398.511	5	0.61	0.61
R111	370.940	230.300	364.100	4	358.100	0.031	**358.100**	4	359.032	31.240	359.032	4	-1.39	0.26
R112	367.800	134.000	**367.800**	4	371.690	0.033	**367.800**	4	368.649	24.190	368.649	4	0.23	0.23
R201	413.140	156.200	**409.200**	4	409.200	0.046	**409.200**	4	411.547	20.730	410.127	4	0.23	0.23
R202	328.460	156.800	320.200	2	320.120	0.027	**320.100**	2	320.914	29.030	320.914	2	0.22	0.25
R203	315.820	196.200	312.700	3	307.700	0.035	**307.700**	3	308.572	26.370	308.572	3	-1.32	0.28
R204	305.720	128.200	302.600	1	297.100	0.031	**297.100**	2	297.788	17.650	297.788	2	-1.59	0.23
R205	356.640	158.800	**352.700**	4	352.700	0.047	**352.700**	4	353.848	29.170	353.848	4	0.33	0.33
R206	386.760	182.800	384.800	2	383.800	0.042	**383.800**	2	384.767	30.380	384.767	2	-0.01	0.25
R207	322.040	177.800	**312.700**	2	312.700	0.040	**312.700**	2	311.367	22.340	311.367	1	-0.43	-0.43
R208	251.200	144.000	**247.200**	1	247.200	0.034	**247.200**	1	248.161	16.020	248.161	1	0.39	0.39
R209	370.440	137.900	**358.600**	2	359.063	0.043	**358.600**	2	364.843	29.490	364.843	2	1.74	1.74
R210	322.340	168.500	320.200	2	319.000	0.046	**319.000**	2	320.184	22.930	320.184	2	-0.00	0.37
R211	331.280	140.000	**329.400**	2	332.067	0.054	**329.400**	2	336.524	24.040	336.524	2	2.16	2.16
RC101	628.680	124.600	**627.700**	5	628.243	0.038	**627.700**	5	631.729	35.210	631.729	6	0.64	0.64
RC102	552.480	149.100	**538.300**	5	538.300	0.043	**538.300**	5	539.757	46.760	539.543	5	0.23	0.23
RC103	468.300	108.800	**468.300**	5	468.300	0.026	**468.300**	5	469.151	26.810	469.151	5	0.18	0.18
RC104	546.100	143.100	**546.100**	5	546.100	0.038	**546.100**	5	552.530	24.910	552.530	5	1.18	1.18
RC105	581.680	180.800	579.800	6	583.627	0.036	**578.300**	6	580.285	25.920	579.608	6	-0.03	0.23
RC106	555.600	115.200	**555.600**	5	556.773	0.037	**555.600**	5	556.579	37.130	556.579	5	0.18	0.18
RC107	578.800	143.100	**578.800**	5	578.800	0.038	**578.800**	5	596.606	23.240	596.263	5	3.02	3.02
RC108	532.820	105.900	**532.800**	5	532.800	0.034	**532.800**	5	533.570	28.730	533.570	5	0.14	0.14
RC201	539.140	145.600	**538.500**	4	538.500	0.041	**538.500**	4	539.543	25.520	539.543	4	0.19	0.19
RC202	415.240	126.800	**412.600**	3	412.600	0.029	**412.600**	3	413.392	23.520	413.392	3	0.19	0.19
RC203	415.460	110.700	**415.000**	3	415.233	0.043	**415.000**	3	415.566	26.180	415.566	3	0.14	0.14
RC204	332.300	99.400	**332.300**	1	332.300	0.029	**332.300**	1	334.348	13.660	334.348	1	0.62	0.62
RC205	441.040	195.000	**440.800**	4	440.800	0.040	**440.800**	4	441.633	33.160	441.633	4	0.19	0.19
RC206	470.840	200.900	**470.200**	3	470.253	0.045	**470.200**	3	471.646	21.830	471.003	3	0.17	0.17
RC207	355.760	120.700	**352.500**	1	352.500	0.028	**352.500**	1	353.055	20.440	353.055	1	0.16	0.16
RC208	243.700	160.400	**243.700**	1	243.700	0.021	**243.700**	1	244.101	17.880	244.101	1	0.16	0.16
Média		165.732				0.039				30.050			0.06	0.52

Tabela 5.4: Resultados computacionais para instâncias grandes ($n = 100$).

	SA				IG				HGS				GAP (%)	
Instância	Média	Tempo(s)	Melhor	NV	Média	Tempo(s)	Melhor	NV	Média	Tempo(s)	Melhor	NV	SA	IG
C101	849.260	514.400	827.600	10	811.903	0.477	**797.100**	10	804.160	216.580	798.844	10	-3.47	0.22
C102	1046.960	622.800	1026.700	11	945.220	0.675	929.500	10	935.700	287.360	**928.455**	10	-9.57	-0.11
C103	890.740	737.600	879.000	10	840.683	0.524	**833.300**	10	841.455	324.220	836.628	10	-4.82	0.40
C104	710.060	444.700	700.400	10	695.173	0.455	**688.200**	10	694.070	153.620	691.549	10	-1.26	0.49
C105	989.520	618.900	950.500	10	948.460	0.531	**931.100**	10	944.518	218.140	934.149	10	-1.72	0.33
C106	932.500	593.000	921.800	12	874.200	0.644	863.800	10	864.284	223.990	**861.712**	10	-6.52	-0.24
C107	1081.220	414.200	1030.500	11	974.247	0.717	**960.600**	10	962.448	147.120	962.423	10	-6.61	0.19
C108	789.740	675.500	786.700	10	777.873	0.393	**768.800**	10	773.092	210.310	771.692	10	-1.91	0.38
C109	832.280	471.100	819.500	10	788.870	0.585	783.900	10	778.474	233.680	**778.011**	10	-5.06	-0.75
C201	564.300	533.400	**511.800**	3	511.800	0.381	**511.800**	3	517.304	49.000	517.304	3	1.08	1.08
C202	513.860	362.900	513.000	4	475.773	0.269	**471.800**	3	473.726	137.640	473.448	3	-7.71	0.35
C203	528.280	558.400	497.100	3	479.030	0.362	**478.400**	3	480.508	139.190	480.430	3	-3.35	0.42
C204	527.660	418.600	512.200	3	490.907	0.466	**488.100**	3	489.536	109.250	489.536	3	-4.42	0.29
C205	474.560	449.800	453.800	3	429.500	0.231	**429.500**	3	431.111	119.780	430.939	3	-5.04	0.34
C206	652.200	729.800	636.200	4	579.457	0.512	**574.200**	3	576.295	121.640	575.135	3	-9.60	0.16
C207	476.680	338.200	461.500	3	460.993	0.373	**460.700**	3	462.738	89.510	462.210	3	0.15	0.33
C208	561.440	477.300	550.900	4	514.070	0.421	**505.200**	3	507.047	89.740	506.861	3	-7.99	0.33
R101	776.900	701.400	760.100	11	756.243	0.477	**753.500**	10	768.202	88.840	763.329	11	0.42	1.30
R102	735.780	692.000	718.000	9	687.470	0.528	**683.600**	8	704.322	112.010	698.629	8	-2.70	2.20
R103	750.480	912.900	738.400	11	727.143	0.630	**723.600**	9	733.805	156.010	731.537	9	-0.93	1.10
R104	697.100	609.300	676.300	8	656.363	0.552	**646.400**	8	655.517	147.160	652.918	8	-3.46	1.01
R105	733.920	551.700	725.800	9	708.527	0.421	**702.400**	8	712.721	136.460	712.193	8	-1.87	1.39
R106	677.000	545.800	651.800	8	648.970	0.430	**630.400**	8	654.321	140.110	647.414	8	-0.67	2.70
R107	728.000	543.900	719.100	8	689.573	0.490	**672.300**	8	675.365	83.240	674.887	8	-6.15	0.38
R108	544.060	590.700	533.000	8	528.640	0.358	**527.800**	8	529.532	113.260	529.532	8	-0.65	0.33
R109	687.100	598.700	673.800	8	657.093	0.480	**649.300**	8	662.942	141.850	656.221	8	-2.61	1.07
R110	666.640	554.000	652.600	8	646.947	0.472	**642.900**	8	655.290	249.320	648.936	8	-0.56	0.94
R111	756.640	440.400	739.400	9	709.333	0.500	706.400	8	708.007	84.590	**703.810**	8	-4.81	-0.37
R112	698.120	507.000	687.900	8	688.777	0.580	**676.000**	8	678.426	251.010	676.873	8	-1.60	0.13
R201	707.640	599.500	700.100	6	679.743	0.579	**678.100**	5	683.002	184.930	680.636	5	-2.78	0.37
R202	612.320	356.900	603.200	4	591.040	0.386	**588.100**	4	592.807	74.370	592.807	4	-1.72	0.80
R203	592.300	615.000	585.600	4	549.760	0.502	**549.000**	4	551.003	69.050	551.003	4	-5.91	0.36
R204	405.320	514.100	387.200	2	383.300	0.376	**383.300**	2	384.938	77.790	384.938	2	-0.58	0.43
R205	496.400	483.300	491.400	3	478.500	0.406	**478.500**	3	480.354	143.050	480.354	3	-2.25	0.39
R206	549.380	513.000	544.500	3	519.650	0.444	**516.800**	3	523.895	175.100	523.895	3	-3.78	1.37
R207	503.920	343.400	492.100	2	474.200	0.442	**473.700**	2	476.334	140.640	475.380	2	-3.40	0.35
R208	359.300	533.500	354.800	2	354.500	0.310	**354.500**	2	356.029	131.180	355.950	2	0.32	0.41
R209	454.660	569.100	442.300	3	434.350	0.439	**432.300**	3	438.738	142.290	438.738	3	-0.81	1.49
R210	555.900	455.900	551.700	4	547.303	0.496	**546.100**	5	551.209	237.670	550.773	4	-0.17	0.86
R211	444.020	466.000	430.700	3	433.840	0.457	**426.000**	3	427.783	131.780	427.783	3	-0.68	0.42
RC101	1029.200	540.900	1023.500	11	981.857	0.515	**968.600**	10	975.766	75.510	975.299	10	-4.71	0.69
RC102	976.580	708.600	945.300	10	926.733	0.472	**912.500**	9	921.457	224.090	921.170	10	-2.55	0.95
RC103	1090.580	527.800	1082.100	11	995.807	0.526	**976.000**	9	987.005	169.420	986.131	9	-8.87	1.04
RC104	914.040	522.500	889.400	10	858.350	0.526	839.500	9	831.534	187.610	**828.867**	9	-6.81	-1.27
RC105	1104.820	564.600	1082.900	11	1029.283	0.673	**1010.100**	10	1032.647	99.730	1025.673	11	-5.28	1.54
RC106	825.080	755.800	817.000	10	804.057	0.320	803.800	9	810.368	201.060	**794.713**	9	-2.73	-1.13
RC107	824.920	513.000	817.100	10	807.550	0.400	**802.800**	9	804.911	175.670	804.537	9	-1.54	0.22
RC108	971.220	758.600	964.100	9	944.720	0.575	927.100	9	929.120	89.980	**926.361**	9	-3.91	-0.08
RC201	729.580	569.700	720.000	6	697.600	0.365	**697.600**	5	706.813	127.640	700.008	5	-2.78	0.35
RC202	627.940	505.700	599.200	5	585.700	0.492	**585.100**	5	604.262	245.960	596.764	5	-0.41	1.99
RC203	704.360	496.300	684.300	4	635.700	0.460	**635.700**	3	643.210	204.760	642.647	3	-6.09	1.09
RC204	503.340	491.800	491.400	2	482.580	0.407	**481.800**	2	483.084	115.600	482.899	2	-1.73	0.23
RC205	770.640	531.700	762.900	5	743.960	0.550	**743.700**	5	746.122	167.280	745.664	5	-2.26	0.26
RC206	588.400	584.000	573.700	5	554.633	0.386	**553.600**	4	555.393	98.150	555.393	4	-3.19	0.32
RC207	611.140	468.300	599.500	4	595.130	0.650	**592.900**	4	594.673	191.180	594.673	4	-0.81	0.30
RC208	507.260	437.700	499.800	3	496.890	0.416	**496.200**	2	497.923	135.110	497.923	3	-0.38	0.35
Média		547.055				0.473				153.410			-3.20	0.54

qualidade de solução comparável à do SA, apresentando um *gap* médio de 0,06%, embora ambos fiquem ligeiramente abaixo do desempenho do IG (gap médio de 0,52%). O destaque, contudo, reside na eficiência computacional. Enquanto o SA demanda, em média, 165,73 segundos para concluir a busca, o HGS consegue seu resultado em 30,05 segundos, mesmo considerando a penalização de desempenho aplicada para a equiparação de hardware. Esse comportamento sugere que a abordagem baseada em HGS consegue explorar o espaço de soluções de forma mais ágil à medida que a dimensão do problema aumenta, beneficiando-se tanto de sua arquitetura populacional quanto das adaptações na busca local projetadas para explorar eficientemente a consolidação de entregas em armários.

O comportamento mais relevante é observado nas instâncias grandes ($n = 100$), cujos dados detalhados encontram-se na Tabela 5.4. Nesse cenário, o algoritmo proposto supera consistentemente o método SA, alcançando um *gap* médio de -3,20% (indicando custos inferiores) com apenas uma fração do tempo de execução (média de 153,41 segundos contra 547,06 segundos do SA). Em relação ao IG, embora o desempenho médio deste último seja superior, com um gap médio de 0,54%, o método proposto demonstrou competitividade, obtendo custos menores ou equivalentes em 7 das 56 instâncias analisadas, com destaque para os cenários RC104 e RC106, onde o HGS superou o IG com um *gap* de mais de 1%. Tais resultados evidenciam a robustez da abordagem proposta para problemas de maior escala, equilibrando qualidade de solução e tempo de resposta de forma eficaz.

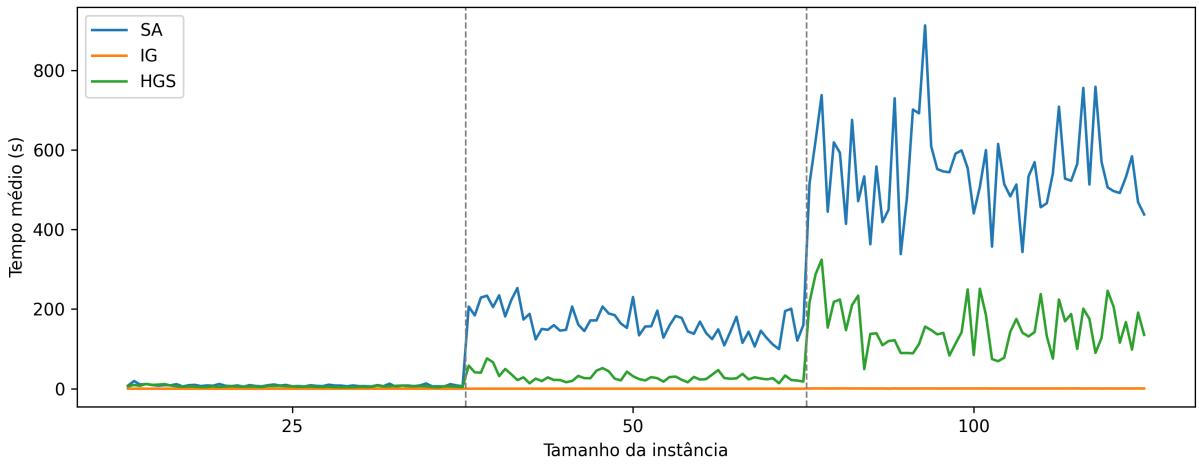
Complementarmente, uma análise focada na média das execuções reforça a consistência do método, especialmente em cenários de maior complexidade. Observa-se que o HGS obteve uma média de custo inferior (melhor) à do IG em 19 das 56 instâncias de grande porte ($n = 100$), com dados detalhados na Tabela 5.4. Além disso, nota-se que a diferença entre o custo médio e o melhor custo encontrado tende a ser consideravelmente menor no HGS em comparação ao IG na maioria dos casos analisados para esse grupo de instâncias (39 das 56 instâncias). Por exemplo, na instância RC108, o IG apresenta uma oscilação de aproximadamente 17,6 unidades entre a média e o melhor valor, enquanto no HGS essa diferença cai para apenas 2,8 unidades. Esse comportamento indica que, embora

o IG possa atingir picos de qualidade ligeiramente superiores em situações ideais, o HGS apresenta menor variabilidade entre execuções, oferecendo uma performance mais previsível e estável.

Esses resultados indicam que, mesmo frente a uma abordagem reconhecidamente eficiente do ponto de vista computacional, como o IG proposto por Corrêa et al. (8), o uso de um algoritmo baseado em HGS, aliado a adaptações pontuais na busca local para capturar características do VRPPL, constitui uma alternativa viável e robusta.

A Figura 5.2 apresenta uma comparação gráfica do tempo médio de execução entre os métodos avaliados. Observa-se que o IG apresenta os menores tempos computacionais em todas as classes de instância, enquanto o HGS ocupa uma posição intermediária entre o IG e o SA. Em particular, nota-se que o tempo de execução do HGS cresce de forma mais controlada do que o do SA à medida que o tamanho das instâncias aumenta, o que reforça sua viabilidade computacional em cenários de maior escala.

Figura 5.2: Comparação do tempo médio de execução entre SA, IG e HGS para instâncias de diferentes tamanhos.



Fonte: Elaborado pelo autor.

6 CONCLUSÃO

Este trabalho investigou o Problema de Roteamento de Veículos com Armários de Encomendas (VRPPL), uma variante recente e relevante do Problema de Roteamento de Veículos (VRP) motivada por desafios práticos da distribuição urbana na etapa de última milha. O objetivo principal consistiu em avaliar a viabilidade e o desempenho de uma abordagem baseada em *Hybrid Genetic Search* (HGS) para esse problema, analisando sua competitividade frente a métodos consolidados e mais recentes da literatura.

A abordagem adotada baseou-se na utilização do PyVRP, um *framework* com implementação moderna e eficiente do HGS originalmente desenvolvida para variantes clássicas do VRP com janelas de tempo. O VRPPL foi tratado por meio de uma modelagem compatível com as estruturas disponíveis na ferramenta, explorando o conceito de grupos de clientes para representar alternativas mutuamente exclusivas de atendimento e incorporando ajustes no processo de avaliação das rotas, de modo a refletir adequadamente o tempo de serviço consolidado em armários de encomendas.

Além das adaptações de modelagem, foi implementado um operador adicional de busca local, denominado *locker reducer*, voltado à consolidação de múltiplas visitas a um mesmo armário ao longo de uma rota. Esse operador explora uma característica específica do problema e complementa os movimentos clássicos de busca local, contribuindo para um tratamento mais apropriado da consolidação de entregas em armários.

Os experimentos computacionais indicam que o método proposto é competitivo em todas as classes de instância avaliadas. Em comparação com o *Simulated Annealing*, o HGS apresentou desempenho superior sobretudo em termos de tempo de execução, mantendo qualidade de solução comparável ou melhor à medida que o tamanho das instâncias aumenta. Quando comparado ao *Iterated Greedy*, os resultados obtidos foram, em média, próximos, com o IG apresentando menor tempo computacional e vantagem em qualidade global, enquanto o HGS alcançou desempenho similar e, em algumas instâncias, resultados equivalentes ou superiores. Esses resultados evidenciam que o HGS representa um bom compromisso entre qualidade de solução e tempo de execução no contexto do VRPPL.

Como limitações do presente trabalho, destaca-se a dependência da arquitetura e dos operadores disponibilizados pelo PyVRP, o que restringe o grau de especialização do algoritmo em relação às particularidades do VRPPL. Além disso, a análise experimental concentrou-se exclusivamente nas instâncias propostas por Yu et al. (7), não contemplando cenários adicionais ou dados reais que poderiam reforçar a avaliação prática da abordagem.

Como perspectivas para trabalhos futuros, sugere-se a investigação de operadores de busca local especializados para o contexto de armários e a hibridização do HGS com a meta-heurística *Iterated Greedy* (IG). Devido ao custo computacional reduzido do IG, sua integração poderia refinar as soluções sem onerar o tempo de execução. Além disso, recomenda-se o estudo de estratégias automáticas de calibração de parâmetros e a aplicação do método a instâncias de maior escala ou dados reais de logística urbana, visando uma avaliação mais ampla da viabilidade do algoritmo em cenários operacionais.

REFERÊNCIAS

[1] IMARC, "Logistics market size, share, trends and forecast by model type, transportation mode, end use, and region, 2025-2033," tech. rep., IMARC, 2025.

[2] Amazon, "Amazon is expanding and accelerating same-day delivery across europe," 2025. Acesso em: 2025.

[3] J. H. R. van Duin, B. W. Wiegmans, B. van Arem, and Y. van Amstel, "From home delivery to parcel lockers: A case study in amsterdam," *Transportation Research Procedia*, vol. 46, pp. 37–44, 2020.

[4] Y. Deutsch and B. Golany, "A parcel locker network as a solution to the logistics last mile problem," *International Journal of Production Research*, vol. 56, pp. 251–261, 2017.

[5] T. Vidal, T.-G. Crainic, M. Gendreau, and C. Prins, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems," *Computers & Operations Research*, vol. 37, no. 11, pp. 184–195, 2012.

[6] N. A. Wouda, L. Lan, and W. Kool, "PyVRP: a high-performance VRP solver package," *INFORMS Journal on Computing*, vol. 36, no. 4, pp. 943–955, 2024.

[7] V. F. Yu, H. Susanto, P. Jodiawan, T.-W. Ho, S.-W. Lin, and Y.-T. Huang, "A simulated annealing algorithm for the vehicle routing problem with parcel lockers," *IEEE Access*, vol. 10, pp. 20764–20782, 2022.

[8] R. F. Corrêa, S. S. Soares, L. B. Gonçalves, and L. L. Moreno, "Iterated greedy algorithm for vehicle routing problem with parcel lockers." Submitted for publication, 2024.

[9] P. Toth and D. Vigo, eds., *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications, Philadelphia, PA: SIAM, 2002.

[10] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 1–140, 1959.

[11] G. Laporte, "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 407–548, 2009.

[12] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[13] N. Boysen, D. Briskorn, and S. Schwerdfeger, "Last-mile delivery concepts: a survey from an operational research perspective," *OR Spectrum*, vol. 43, no. 1, pp. 1–58, 2021.

[14] S. Iwan, K. Kijewska, and J. Lemke, "Analysis of parcel lockers' efficiency as the last mile delivery solution – the results of the research in poland," *Transportation Research Procedia*, vol. 12, pp. 644–655, 2016.

[15] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, 1964.

[16] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of Metaheuristics* (F. Glover and G. A. Kochenberger, eds.), pp. 320–353, Boston, MA: Springer, 2003.

[17] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[18] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[19] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Tech. Rep. 826, Caltech Concurrent Computation Program, Pasadena, CA, 1989.

[20] T. Vidal, T.-G. Crainic, M. Gendreau, and C. Prins, "A unified solution framework for multi-attribute vehicle routing problems," *European Journal of Operational Research*, vol. 234, no. 3, pp. 658–673, 2014.

[21] T. Vidal, "Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood," *Computers & Operations Research*, vol. 140, p. 105643, 2022.

[22] Y. Nagata and S. Kobayashi, "A memetic algorithm for the pickup and delivery problem with time windows using selective route exchange crossover," in *International Conference on Parallel Problem Solving from Nature*, pp. 536–545, Springer, 2010.

[23] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.