UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Leonardo Lima Appio

Avaliação de pacotes computacionais para um problema de roteamento de veículos com frota heterogênea

Florianópolis
2025

Leonardo Lima Appio

Avaliação de pacotes computacionais para um problema de roteamento de veículos com frota heterogênea

Trabalho de Conclusão de Curso submetido ao curso de Ciência da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Pedro Belin Castellucci, Dr.
Coorientador: Prof. Rafael de Santiago, Dr.

Florianópolis
2025

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Appio, Leonardo Lima

Avaliação de pacotes computacionais para um problema de roteamento de veículos com frota heterogênea / Leonardo Lima Appio ; orientador, Pedro Belin Castellucci, coorientador, Rafael de Santiago, 2025.

138 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2025.

Inclui referências.

1. Ciências da Computação. 2. Roteamento de Veículos. 3. Avaliação de Pacotes Computacionais. 4. Programação Inteira. I. Castellucci, Pedro Belin. II. de Santiago, Rafael. III. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. IV. Título.

Leonardo Lima Appio

Avaliação de pacotes computacionais para um problema de roteamento de veículos com frota heterogênea

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel em Ciências da Computação e aprovado em sua forma final pelo Curso de Graduação em Ciências da Computação.

Florianópolis, 25 de novembro de 2025.

_____

Profa. Lúcia Helena Martins Pacheco, Dra

Coordenadora do Curso

Banca examinadora

_____

Prof. Pedro Belin Castellucci, Dr.

Orientador

Universidade Federal de Santa Catarina

_____

Prof. Rafael de Santiago, Dr.

Coorientador

Universidade Federal de Santa Catarina

_____

Prof. Álvaro Junio Pereira Franco, Dr.

Avaliador

Universidade Federal de Santa Catarina

_____

Prof. Eduardo Delcides Bernardes, Dr.

Avaliador

Universidade Estadual de Santa Cruz

AGRADECIMENTOS

Agradeço primeiramente à minha família, ao meu pai, Alisson, por ter me introduzido ao mundo da computação ainda criança e por todos os ensinamentos e conselhos que me guiaram ao longo desses anos. À minha mãe, Glizauda, pelo apoio incondicional, pelo carinho e por sempre acreditar no meu potencial.

Ao meu irmão, Anthony, pelas brincadeiras e energia contagiante, que tornaram todo este processo mais leve.

À minha namorada, Mariana, pelo companheirismo, amor, brincadeiras e apoio em todos os momentos.

Ao meu orientador, Prof. Pedro, pelos valiosos ensinamentos, pela orientação atenta e pela dedicação em me auxiliar no desenvolvimento deste trabalho.

Ao meu coorientador, Prof. Rafael, pelas contribuições e pelo suporte durante o processo de elaboração deste trabalho.

Aos meus grandes amigos de curso que me acompanharam durante toda essa jornada, João Paulo, Caio, Rodrigo, Arthur, Rafael Neves, João Victor e Thiago, pelo companheirismo ao longo desses anos de graduação.

Em especial, agradeço ao meu grande amigo Fillipi pela parceria de sempre desde os tempos do ensino médio. Pelas incontáveis ideias de empresas que nunca terminamos, pelas noites viradas fazendo trabalhos, pelas risadas, pelos inúmeros cachorros-quentes do Salt Dog e por tornar essa caminhada infinitamente mais divertida. Sua amizade é um dos grandes presentes que levo desta fase.

A todos que, de alguma forma, contribuíram para minha formação e para a realização deste trabalho, minha sincera gratidão.

Agradeçemos o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, 405247/2023-0).

RESUMO

O problema de roteamento de veículos com frota heterogênea (*heterogeneous vehicle routing problem,* HVRP) é uma derivação do clássico problema de roteamento de veículos (*vehicle routing problems,* VRP), que envolve a otimização das rotas de uma frota de veículos para atender a um conjunto de clientes. Diferentemente das frotas homogêneas, que possuem veículos com as mesmas capacidades e custos, no HVRP há a complexidade adicional de gerenciar uma frota composta por veículos com diferentes capacidades entre si, o que demanda abordagens computacionais mais sofisticadas para a solução eficiente do problema. Embora existam várias técnicas e pacotes computacionais (*solvers*) disponíveis para resolver problemas de otimização, não há consenso na literatura sobre qual abordagem é mais eficaz em contextos de frotas heterogêneas, especialmente considerando as variações entre diferentes instâncias do problema. Este trabalho busca avaliar comparativamente o desempenho dos *solvers* baseados em *Branch-and-Cut*, analisando como se comportam em diferentes instâncias do HVRP e identificando as circunstâncias em que um *solver* pode superar os demais.

Palavras-chave: Problema de Roteamento de Veículos, Frota Heterogênea, Programação Inteira, *Solvers* baseados em *Branch-and-Cut*.

ABSTRACT

The heterogeneous vehicle routing problem (HVRP) is a variation of the classic vehicle routing problem (VRP), which involves optimizing the routes for a fleet of vehicles to serve a set of customers. Unlike the case with homogeneous fleets, which have vehicles with the same capacities and costs, the HVRP introduces the additional complexity of managing a fleet composed of vehicles with different capacities, requiring more sophisticated computational approaches for the efficient solution of the problem. Although several techniques and computational packages (solvers) are available to solve optimization problems, there is no consensus in the literature on which approach is most effective in the context of heterogeneous fleets, especially considering the variations between different problem instances. This work aims to comparatively evaluate the performance of solvers based on Branch-and-Cut techniques, analyzing how they behave on different HVRP instances and identifying the circumstances in which one solver may outperform the others.

Keywords: Vehicle Routing Problem, Heterogeneous Fleet, Integer Programming, Branch-and-Cut based Solvers.

SUMÁRIO

1 INTRODUÇÃO

A otimização do transporte é um tema de grande relevância no cenário atual, especialmente em um contexto em que as empresas buscam reduzir custos e melhorar a eficiência operacional. O Problema de Roteamento de Veículos (VRP), que consiste em determinar as rotas mais eficientes para uma frota de veículos atender a um conjunto de clientes, é um problema de otimização crucial para alcançar esses objetivos. Problemas de roteamento de veículos têm sido amplamente estudados, com uma vasta literatura dedicada a maximizar a utilização de recursos e minimizar distâncias percorridas, como evidenciado em extensas revisões da área, como as de Laporte (2009) e Toth e Vigo (2014). No entanto, muitas das soluções tradicionalmente propostas focam em frotas homogêneas, compostas por veículos idênticos em capacidade e custo. Na prática, a realidade das empresas de transporte é frequentemente mais complexa, envolvendo frotas heterogêneas, com veículos de diferentes capacidades e custos operacionais, o que cria novos desafios de modelagem e solução.

O Problema de Roteamento de Veículos com Frota Heterogênea (HVRP) foi formalmente introduzido por Golden *et al.* (1984) e surge como uma variante mais sofisticada e realista do problema de roteamento clássico. Diferentemente do VRP tradicional, que frequentemente assume veículos idênticos, o HVRP aborda a complexidade de gerenciar frotas compostas por diferentes tipos de veículos, cada um com suas próprias capacidades, custos fixos e variáveis (Koç; Bektaş; Jabali; Laporte, 2016). Esta variante exige não apenas a definição das melhores rotas para atender a um conjunto de clientes, mas também a crucial decisão sobre qual tipo de veículo deve ser alocado a cada rota e, em algumas formulações, a própria composição ideal da frota a ser utilizada. Neste cenário desafiador, os pacotes computacionais (*solvers*), que implementam algoritmos de otimização, desempenham um papel fundamental na busca por soluções eficientes. Contudo, a literatura demonstra que existem diversas abordagens e algoritmos disponíveis, e o desempenho dessas ferramentas pode variar significativamente dependendo das características específicas do problema HVRP e das instâncias analisadas (Koç; Bektaş; Jabali; Laporte, 2016).

Portanto, a necessidade desta pesquisa reside na lacuna de conhecimento sobre a eficácia comparativa de diferentes *solvers* no contexto do HVRP, como não

há consenso na literatura sobre qual solução computacional é a mais eficaz para lidar com frotas heterogêneas em cenários variados, esta pesquisa busca avaliar e comparar *solvers* com o objetivo de identificar quais ferramentas oferecem os melhores resultados para diferentes instâncias do HVRP.

A pesquisa parte do pressuposto de que *solvers* baseados em *Branch-and-Cut* podem apresentar desempenhos distintos ao serem aplicados ao HVRP. A hipótese central é que a escolha do *solver* ideal depende de fatores como a composição da frota e a distribuição dos clientes e outras características das instâncias, sendo esperado que certos *solvers* tenham vantagens sobre outros em instâncias específicas. Este trabalho é relevante para a área de otimização de transportes, pois contribui para o entendimento de como diferentes pacotes computacionais lidam com a complexidade do HVRP, ajudando a identificar as condições sob as quais certos *solvers* são mais adequados e, assim, auxiliando em decisões mais informadas na escolha de ferramentas computacionais.

1.1 Objetivos

Este trabalho tem como objetivo geral avaliar o desempenho de diferentes pacotes computacionais (*solvers*) na resolução do problema de roteamento de veículos com frota heterogênea (HVRP). Para isso, são propostos os seguintes objetivos específicos.

1. Revisar a literatura sobre HVRP e os modelos de programação linear inteira frequentemente utilizados.
2. Analisar os principais pacotes computacionais aplicados à otimização em transportes.
3. Selecionar os *solvers* mais representativos; avaliar sua eficácia em um conjunto de instâncias padronizadas do HVRP.
4. Comparar os resultados obtidos em termos de tempo de solução, custo e eficiência das rotas geradas.
5. E, se possível, identificar as características das instâncias que impactam o desempenho de cada *solver*.

O restante do documento está organizado da seguinte forma. No Capítulo 2, são apresentados os conceitos fundamentais que sustentam o estudo, abordando a Pesquisa Operacional, os métodos de solução e os principais *solvers* utilizados. O

Capítulo 3 reúne os trabalhos relacionados, com ênfase em pesquisas anteriores sobre o HVRP e comparações entre pacotes computacionais. No Capítulo 4, define-se formalmente o problema estudado, incluindo a formulação matemática adotada. O Capítulo 5 descreve a metodologia utilizada na condução dos experimentos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais para compreensão do Problema de Roteamento de Veículos - VRP (da nomenclatura em inglês *Vehicle Routing Problem*) - e suas variantes, incluindo metodologias de solução exata e heurística, bem como ferramentas computacionais empregadas na resolução desses problemas. A fundamentação teórica sustenta a formulação do problema e embasa a escolha dos métodos aplicados no presente trabalho.

2.1 Pesquisa Operacional

A Pesquisa Operacional trata da modelagem e solução de problemas complexos de tomada de decisão, utilizando métodos matemáticos e computacionais. Segundo Arenales *et al.* (2007), a Pesquisa Operacional busca otimizar sistemas com múltiplas restrições e objetivos concorrentes, sendo aplicada em diversas áreas, incluindo logística e transporte.

Ainda segundo Arenales *et al.* (2007), historicamente, a Pesquisa Operacional ganhou destaque durante a Segunda Guerra Mundial, quando equipes de cientistas eram convocadas para resolver problemas relacionados ao uso ótimo de recursos militares, planejando rotas de comboios, posicionamento de radares e gerenciamento de suprimentos. Após o conflito, as técnicas desenvolvidas evoluíram e se espalharam por setores civis, como a indústria de transporte, o planejamento logístico e a gestão de serviços.

O VRP (*Vehicle Routing Problem*) é um problema clássico da Pesquisa Operacional que visa determinar um conjunto ótimo de rotas para uma frota de veículos atender a um conjunto de clientes, minimizando custos operacionais e respeitando restrições como a capacidade dos veículos (Laporte, 2009). Sua relevância se deve à sua aplicação prática em logística, distribuição de mercadorias, coleta de lixo e transporte de passageiros. Diferentes abordagens para solução deste problema têm sido desenvolvidas ao longo dos anos, variando desde técnicas exatas até heurísticas e metaheurísticas (discutidas na Seção 2.2.3).

2.1.1 Processo de modelagem

O processo de modelagem na Pesquisa Operacional, de forma simplificada, segue algumas etapas fundamentais, como destacado por Goldbarg e Luna (2005) na Figura 1:

- **Definição do Problema:** Envolve compreender profundamente o sistema real a ser estudado, identificando seus objetivos, limitações operacionais, variáveis críticas e os critérios de desempenho. Nesta etapa deve ser garantido que o problema a ser modelado seja bem delimitado e reflita as necessidades reais da organização.
- **Formulação e Construção do Modelo Inicial:** Após compreender o problema, passa-se à sua formalização matemática. Isso envolve definir as variáveis de decisão, estabelecer a função objetivo (a ser maximizada ou minimizada) e identificar as restrições do sistema. É nessa etapa que o problema real é abstraído para um modelo matemático, permitindo o uso de técnicas de otimização.
- **Validação do Modelo:** A validação verifica se o modelo representa corretamente o comportamento do sistema e se os resultados gerados fazem sentido no contexto real. Isso pode ser feito por meio da comparação com dados históricos.
- **Simulação do Modelo:** A simulação permite analisar o comportamento do sistema sem riscos reais, testando a robustez das decisões e a sensibilidade das variáveis. Essa etapa é fundamental para antecipar problemas e garantir a confiabilidade da solução proposta.
- **Reformulação do Modelo:** Com base na simulação e validação, ajustes e melhorias são realizados para refinar a representação do problema e garantir maior aderência ao sistema real.
- **Aplicação do Modelo:** Após validar e refinar o modelo, ele é finalmente implementado no ambiente real de decisão. Isso pode envolver a integração com sistemas computacionais, o treinamento de usuários e o acompanhamento dos resultados.

Esse processo não é linear, mas cíclico. A qualquer momento, pode-se retornar a etapas anteriores caso sejam identificadas inconsistências, novas informações ou mudanças no ambiente do problema. A flexibilidade e a adaptabilidade são características essenciais de um processo de modelagem eficiente e realista.

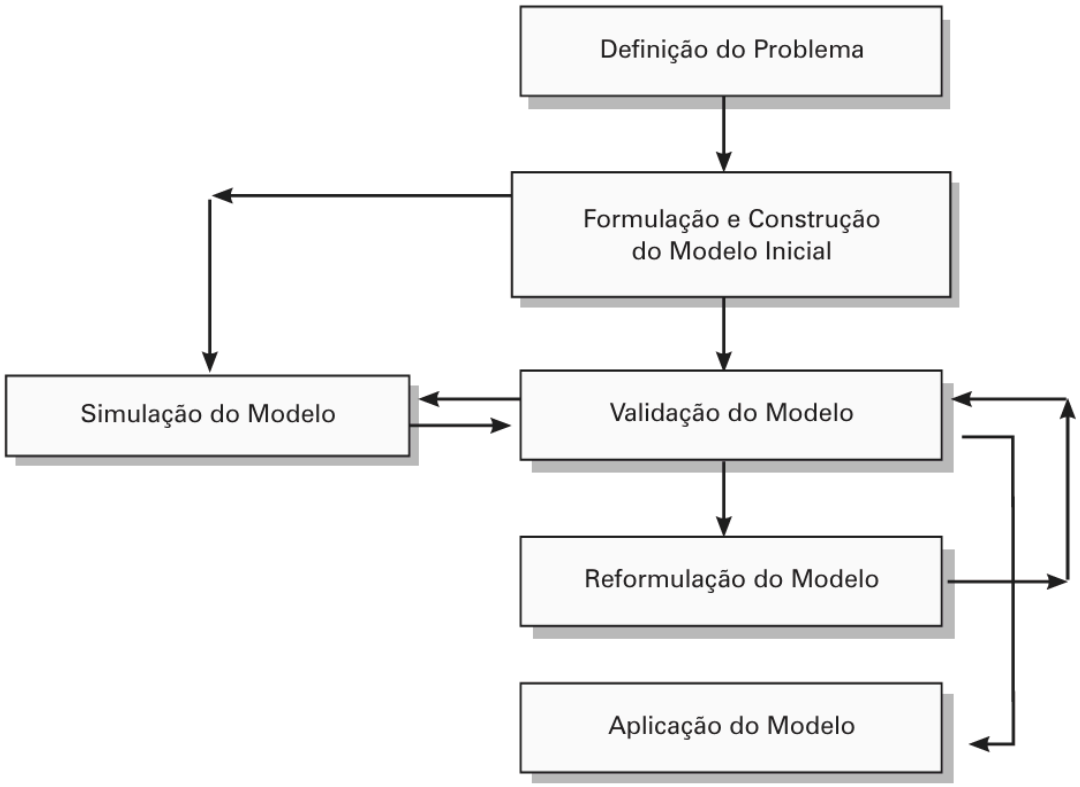

Figura 1 – Ciclo do processo de modelagem em Pesquisa Operacional.

Fonte: Goldbarg e Luna (2005, p. 8).

2.1.2 Aplicações e importância

A Pesquisa Operacional tem aplicações em praticamente todos os domínios onde a alocação de recursos escassos e o planejamento de atividades sejam relevantes. Entre as áreas mais notáveis, destacam-se transporte, produção, energia, telecomunicações e serviços públicos, conforme apresentado por Wolsey (2021, Seção 1.1).

No campo da Logística e Transporte, destacam-se atividades como o planejamento de rotas — incluindo o próprio Problema de Roteamento de Veículos (VRP) —, o dimensionamento de frotas, o agendamento de entregas e a organização de equipes de manutenção. Na Produção e Manufatura, a Pesquisa Operacional é empregada na programação de linhas de produção, no escalonamento de tarefas, no controle de estoques e no planejamento de processos industriais. Já nas áreas de Finanças e Economia, as aplicações envolvem análise de portfólios, previsão de demanda, precificação de ativos e suporte à tomada de decisões de investimento. Por fim, na área de Saúde e Serviços Públicos, a Pesquisa Operacional é utilizada na alocação de equipes médicas, na roteirização de ambulâncias, na elaboração de escalas de trabalho em hospitais e no gerenciamento de filas em serviços de atendimento.

O sucesso e a difusão da Pesquisa Operacional se devem não apenas ao seu arcabouço matemático sólido, mas também à evolução das tecnologias computacionais. À medida que os computadores se tornam mais poderosos, é possível resolver problemas cada vez maiores e mais complexos, fazendo com que os métodos de Pesquisa Operacional sejam constantemente aprimorados para aproveitar essa capacidade adicional.

2.1.3 Pesquisa Operacional e o VRP

Dentro desse contexto mais amplo, o VRP é reconhecido como um dos problemas centrais em Pesquisa Operacional, notável por sua utilidade prática e pela complexidade teórica. Desde o trabalho pioneiro de Dantzig e Ramser (1959), o VRP tem sido objeto de intenso estudo, impulsionando o desenvolvimento de diversas técnicas e algoritmos em otimização combinatória.

Segundo Laporte (2009), sua relevância deriva diretamente de sua forte aplicação prática em logística e transporte, especialmente no ambiente competitivo em que as empresas precisam entregar bens e serviços de forma rápida e econômica. Com o VRP, busca-se determinar rotas que minimizem custos operacionais (por exemplo, distância percorrida ou tempo total de viagem), mas que atendam simultaneamente a restrições como capacidade de carga, janelas de tempo e zonas de acesso restrito.

Outro ponto de destaque é a quantidade de variantes que surgiram a partir do problema clássico. A cada nova característica do mundo real — por exemplo, frota heterogênea, necessidade de entrega e coleta simultânea, limitação de emissão de poluentes, entre outras, introduz-se uma nova variante de VRP, que passa a demandar métodos de resolução dedicados.

Assim, a Pesquisa Operacional fornece a estrutura conceitual e metodológica para abordar o VRP e suas variantes, definindo métodos exatos (como *Branch-and-Cut* e *Branch-and-Bound*) e soluções aproximadas via heurísticas e metaheurísticas. Esse amplo conjunto de metodologias consolidou o que hoje se reconhece como um campo plenamente amadurecido no domínio da logística e do planejamento de transportes.

2.1.4 Desafios atuais em Pesquisa Operacional

Embora os avanços teóricos e computacionais tenham ampliado significativamente a aplicabilidade da Pesquisa Operacional (PO), diversos desafios contemporâneos permanecem, principalmente no contexto de problemas de grande escala e elevada complexidade dinâmica.

Em ambientes logísticos reais, como centros de distribuição e operações de transporte urbano, os dados relevantes ao processo de tomada de decisão são frequentemente atualizados em tempo real, como a chegada de novos pedidos, alterações nas condições de tráfego ou falhas em veículos. Esse dinamismo exige a construção de modelos matemáticos que suportam múltiplas etapas de otimização frequente e adaptação contínua às mudanças do ambiente. Koç; Bektaş; Jabali; Laporte (2016) destacam que, na literatura sobre problemas de roteamento com coleta e entrega simultâneas (VRPSPD), as abordagens dinâmicas têm se tornado cada vez mais relevantes diante da necessidade de decisões responsivas em cenários práticos.

A crescente preocupação com aspectos ambientais e de sustentabilidade tem impulsionado a inclusão de critérios como emissões de gases poluentes, consumo de combustível e impacto ambiental em modelos de otimização, particularmente em problemas de roteamento de veículos. Essa incorporação aumenta a complexidade dos modelos, exigindo abordagens multicritério ou restrições adicionais. Koç;

Bektaş; Jabali; Laporte, (2016) enfatizam que, no contexto dos problemas de roteamento com frotas heterogêneas (HVRP), a consideração de fatores ambientais não apenas altera os critérios de custo, como também influencia diretamente a escolha do tipo de veículo e a estrutura das rotas, tornando os problemas mais desafiadores do ponto de vista computacional e de modelagem.

A integração entre análise de dados em tempo real e técnicas de otimização torna-se cada vez mais crítica, sobretudo com o avanço da Internet das Coisas e de sistemas de rastreamento. Segundo Gleixner *et al.* (2021), a análise empírica de algoritmos de Programação Linear Inteira Mista (PLIM) demonstra que o desempenho dos *solvers* pode variar consideravelmente em ambientes com alta variabilidade e grande volume de dados, reforçando a importância de abordagens flexíveis e escaláveis.

Tais desafios reforçam o papel da Pesquisa Operacional como área interdisciplinar, em constante evolução e adaptação, que se integra a domínios como ciência de dados, sustentabilidade e engenharia de sistemas, com foco na resolução de problemas decisórios em ambientes complexos e dinâmicos.

2.2 Métodos de Solução

Duas abordagens típicas na literatura para resolver o Problema de Roteamento de Veículos (VRP) e suas variantes: os métodos exatos, que garantem a obtenção da solução ótima, e as heurísticas (ou metaheurísticas), que buscam soluções de boa qualidade em tempo reduzido, embora sem a garantia de otimalidade. A escolha entre essas abordagens depende de fatores como o tamanho e a complexidade da instância, os recursos computacionais disponíveis e o nível de precisão desejado na solução. Muitos desses problemas, incluindo o VRP e suas variantes, pertencem à classe dos problemas NP-difíceis, o que faz com que, à medida que o tamanho das instâncias aumenta, os métodos exatos se tornem computacionalmente inviáveis em muitos casos práticos, especialmente para instâncias de grande porte.

2.2.1 Métodos exatos

Métodos exatos garantem a obtenção da solução ótima do problema, ainda que possam ser computacionalmente dispendiosos.

O Branch-and-Bound explora uma árvore de decisão para encontrar a solução ótima, eliminando regiões inviáveis da solução (Yaman, 2005). O *Branch-and-Bound* divide o problema original em subproblemas menores, aplicando relaxação linear e limitantes para acelerar a busca por uma solução ótima.

O Branch-and-Cut expande o *Branch-and-Bound* com planos de corte que visam melhorar o limitante inferior - de um problema de minimização - potencialmente aumentando o número de nós que são podados da árvore de *Branch-and-Bound* (Arenales *et al.* 2007). Este método melhora a eficiência da exploração da árvore de busca ao adicionar restrições que cortam partes inviáveis do espaço de solução.

O Branch-Cut-and-Price integra geração de colunas ao *Branch-and-Cut* para lidar com formulações extensas (Pessoa *et al.*, 2018). Esse método é um dos mais eficientes para VRPs complexos, permitindo a solução de instâncias grandes por meio de uma combinação inteligente de relaxação de variáveis e adição de cortes. É comum realizar uma decomposição do problema original em problema mestre e subproblema. Cada rota elegível passa a ser uma coluna na formulação do problema mestre. O subproblema consiste na geração de rotas (*route generation problem*), resolvido por métodos de busca em grafos, normalmente com estratégias de busca de menor custo ou programação dinâmica.

2.2.2 Planos de corte e sua importância

Os planos de corte são restrições adicionais utilizadas para eliminar porções inviáveis da relaxação linear de um problema de Programação Inteira. Eles são parte essencial do método de *Branch-and-Cut*, que combina *Branch-and-Bound* com geração dinâmica de cortes durante a resolução do problema (Wolsey, 2021).

A escolha dos cortes aplicados, o momento de inserção e a quantidade a ser adicionada influenciam diretamente o desempenho do algoritmo, sendo fatores que variam entre diferentes implementações de *solvers* comerciais e gratuitos. Isso

explica, em parte, as diferenças de desempenho observadas entre ferramentas como CPLEX, Gurobi, SCIP e CBC.

A Figura 2 ilustra graficamente como uma desigualdade válida pode eliminar regiões fracionárias sem excluir soluções inteiras viáveis.

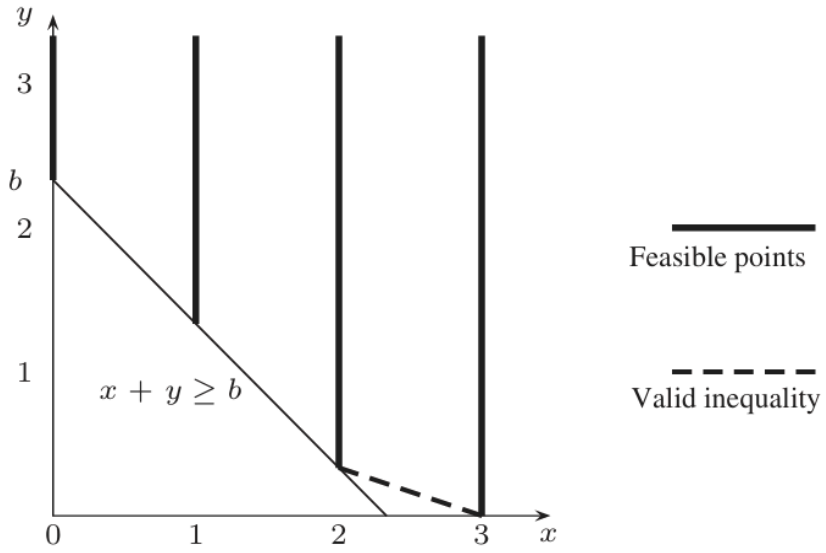

Figura 2 - Exemplo de plano de corte eliminando soluções fracionárias.

Fonte: Wolsey (2021, p. 154).

A Figura 2, apresenta uma representação gráfica de um plano de corte básico aplicado a um problema de programação inteira mista. A linha tracejada (*valid inequality*) representa uma desigualdade válida adicionada para eliminar soluções fracionárias da relaxação linear, mantendo o conjunto de soluções viáveis.

A aplicação desses cortes pode ser feita dinamicamente durante a resolução do problema, reduzindo o espaço viável e aumentando a eficiência computacional. A combinação eficiente desses métodos permite resolver instâncias cada vez maiores e mais complexas tanto do VRP quanto de outros problemas.

2.2.3 Heurísticas e metaheurísticas

Dada a complexidade computacional envolvida nos métodos exatos, heurísticas e metaheurísticas surgem como alternativas populares para a obtenção de soluções de boa qualidade em tempos reduzidos. Um dos exemplos mais

clássicos de heurísticas construtivas para o VRP é o algoritmo de economias de Clarke e Wright, que constrói soluções iniciais de forma rápida a partir de uma configuração trivial — como aquela em que cada cliente é atendido por uma rota exclusiva — e, em seguida, realiza fusões progressivas dessas rotas até formar um conjunto mais eficiente e reduzido de rotas viáveis (Gonçalves *et al.*, 2005).

Já as metaheurísticas têm como principal objetivo refinar as soluções ao longo de múltiplas iterações. Essas técnicas exploram regiões promissoras do espaço de soluções sem garantir a otimalidade, mas frequentemente alcançam soluções de alta qualidade em tempos computacionais aceitáveis (Koç; Bektaş; Jabali; Laporte, 2016). São exemplos de metaheurísticas: Busca Tabu, os Algoritmos Genéticos e o *Iterated Local Search* (ILS). A Busca Tabu, proposta por Glover (1986), destaca-se por empregar uma memória adaptativa que impede a repetição de soluções já visitadas, promovendo a diversificação e permitindo escapar de ótimos locais. Os Algoritmos Genéticos, por sua vez, baseiam-se nos princípios de seleção natural e reprodução biológica para evoluir uma população de soluções ao longo do tempo (Holland, 1975). O ILS (*Iterated Local Search*), por outro lado, introduz perturbações controladas sobre soluções obtidas por busca local, permitindo assim uma exploração mais abrangente do espaço de busca (Lourenço, Martin & Stützle, 2003).

Além disso, algoritmos híbridos têm sido desenvolvidos com o intuito de combinar as vantagens dos métodos heurísticos e exatos. Nessas abordagens, é comum utilizar uma heurística para gerar rapidamente uma solução inicial, que depois é aprimorada por meio de técnicas exatas, como o *Branch-and-Cut*, aplicadas sobre uma parte das variáveis do problema. Essa combinação visa equilibrar o tempo de processamento com a qualidade da solução obtida, aproveitando o poder de generalização das heurísticas e a precisão dos métodos exatos.

2.3 Ferramentas Computacionais

A resolução de problemas de otimização complexos, tais como o VRP e suas variantes, requer ferramentas computacionais (*solvers*) capazes de lidar

eficientemente com grande quantidade de variáveis e restrições. Entre as principais opções disponíveis, destacam-se Gurobi, CPLEX, SCIP e CBC, cada uma com estratégias específicas para a exploração da árvore de *Branch-and-Cut* e diferentes políticas de adição de planos de corte.

A escolha do *solver* mais adequado para problemas de otimização, como o Problema de Roteamento de Veículos (VRP) e suas variantes, é uma decisão crítica e depende de diversos fatores, incluindo o tamanho e a complexidade da instância, os recursos computacionais disponíveis, o orçamento e a necessidade de customização. Cada *solver* adota diferentes estratégias para explorar a árvore de *Branch-and-Cut*, variando nas políticas de separação de cortes, heurísticas de pesquisa e suporte a multiprocessamento. As características comparativas descritas neste trabalho foram obtidas a partir da documentação oficial de cada ferramenta.

2.3.1 Gurobi

O Gurobi é um *solver* comercial amplamente reconhecido pelo seu alto desempenho na resolução de problemas de Programação Linear Inteira e Mista (MILP). Sua aplicação é difundida tanto na indústria quanto na academia, principalmente por sua rapidez na obtenção de soluções e pela extensa variedade de parâmetros ajustáveis que permitem uma personalização precisa da resolução dos problemas (Gurobi, 2025).

Em termos de estratégia algorítmica, o Gurobi implementa algoritmos sofisticados de *Branch-and-Cut*. Ele incorpora heurísticas internas que buscam rapidamente soluções viáveis iniciais, como heurísticas de início e heurísticas de mergulho (*diving heuristics*). Além disso, utiliza cortes automáticos derivados de análises da relaxação linear do problema, incluindo cortes de capacidade, cortes de clique, cortes do tipo *lift-and-project*, entre outros (Gurobi, 2025).

A política de aplicação de planos de corte no Gurobi é adaptativa, isto é, o *solver* ativa ou desativa determinadas classes de cortes com base em estimativas de benefício computacional para cada instância. Adicionalmente, os usuários podem controlar a agressividade na aplicação desses cortes por meio do parâmetro *CuttingLevel*, o que oferece maior flexibilidade na configuração do processo de otimização.

Gurobi é amplamente reconhecido pela sua velocidade e robustez. Possui heurísticas e algoritmos de corte altamente otimizados, com suporte completo a processamento paralelo (*multicore*). Seu diferencial é o desempenho superior em problemas de grande escala, além de permitir ajuste fino de parâmetros (*tuning*) para cenários específicos. É um *software* comercial, mas oferece versões gratuitas para uso acadêmico.

2.3.2 CPLEX

O CPLEX, desenvolvido pela IBM, é outro *solver* comercial que se destaca como líder de mercado na resolução de problemas de Programação Linear Inteira e Mista (MILP). Reconhecido por seu excelente desempenho, especialmente em instâncias de grande porte, o CPLEX também oferece uma notável flexibilidade no ajuste fino de seus parâmetros de controle, o que o torna uma ferramenta poderosa tanto para aplicações industriais quanto acadêmicas (IBM CPLEX, 2025).

Em relação à estratégia de resolução, o CPLEX adota uma abordagem de *Branch-and-Cut* semelhante àquela implementada pelo Gurobi. Contudo, diferencia-se ao oferecer ao usuário um controle mais detalhado sobre cada etapa do processo. É possível, por exemplo, configurar diretamente a estratégia de seleção de variáveis para ramificação por meio de parâmetros específicos (*strategy parameters*), assim como ajustar a frequência com que os cortes são gerados durante a execução do algoritmo (IBM CPLEX, 2025).

No que diz respeito à política de aplicação de planos de corte, o CPLEX também realiza detecção automática de oportunidades de corte durante o processo de resolução. É avaliado dinamicamente o impacto de cada classe de corte sobre o tempo de processamento e toma decisões com base nesses dados. Suas rotinas heurísticas são combinadas com uma identificação eficaz de cortes, o que frequentemente resulta na redução do tamanho do nó-raiz e, por consequência, na diminuição da profundidade da árvore de *branch*, otimizando o desempenho global do processo de busca (IBM CPLEX, 2025).

CPLEX é competitivo com Gurobi, especialmente em problemas que exigem personalização detalhada. Oferece uma ampla gama de parâmetros configuráveis em todas as fases do *Branch-and-Bound* e também suporta processamento paralelo.

Seu uso acadêmico também é gratuito mediante licença. Destaca-se pela estabilidade e pelo suporte a múltiplas linguagens de programação.

2.3.3 SCIP

O SCIP (*Solving Constraint Integer Programs*) é um *solver* de código aberto mantido pelo grupo de otimização do Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB). Ele é amplamente reconhecido por sua elevada flexibilidade, sendo particularmente adequado para fins acadêmicos e de pesquisa. Entre suas principais características está a possibilidade de criação de *plugins* personalizados e a manipulação avançada de restrições, o que o torna uma ferramenta poderosa para usuários que desejam explorar estruturas algorítmicas mais complexas ou não convencionais (SCIP, 2025).

No que tange à estratégia de *Branch-and-Cut*, o SCIP, embora geralmente não alcance a mesma velocidade de execução observada nos *solvers* comerciais como Gurobi ou CPLEX, oferece aos pesquisadores uma liberdade significativa para experimentar novas abordagens. Ele disponibiliza interfaces específicas para a adição de separadores de cortes (*cut separators*), bem como para a definição de comportamentos customizados durante a exploração da árvore de decisão, o que o torna especialmente atrativo para o desenvolvimento e teste de inovações algorítmicas (SCIP, 2025).

Em relação à política de aplicação de planos de corte, o SCIP permite ao usuário não apenas configurar estratégias de corte já existentes, mas também implementar do zero abordagens completamente novas. Essa característica é particularmente vantajosa em projetos de pesquisa que envolvem variantes específicas do Problema de Roteamento de Veículos (VRP), nos quais o uso de cortes especializados pode ser determinante para a obtenção de soluções de melhor qualidade ou para a redução da complexidade computacional (SCIP, 2025).

SCIP é uma ferramenta voltada à pesquisa e desenvolvimento de novos algoritmos. Sua principal vantagem é a flexibilidade: permite a criação de plugins personalizados, facilitando a implementação de novas estratégias de separação de cortes, heurísticas ou regras de *branching*. Embora seu desempenho médio seja tipicamente inferior à dos *solvers* comerciais em grandes instâncias, devido ao seu

comportamento padrão ser sequencial (sem paralelismo), sua capacidade de customização o torna ideal para prototipagem e testes acadêmicos.

2.3.4 CBC

O CBC (*Coin-or Branch-and-Cut*) é um *solver* de código aberto que integra o projeto COIN-OR e é voltado para a resolução de problemas de Programação Linear Inteira Mista (MILP). Embora seu desempenho não atinja o mesmo nível de *solvers* comerciais como Gurobi ou CPLEX, o CBC apresenta bons resultados em aplicações de porte médio. Uma de suas principais vantagens é o fato de não exigir licenças comerciais, o que o torna uma alternativa bastante atrativa para uso acadêmico e para empresas de menor porte que desejam soluções viáveis sem custos de licenciamento (CBC, 2025).

A estratégia de *Branch-and-Cut* adotada pelo CBC é modular, o que permite a ativação ou desativação seletiva de heurísticas internas e de diferentes tipos de cortes. Essa modularidade oferece certa flexibilidade ao usuário e está bem integrada à biblioteca CGL (*Cut Generation Library*), a qual fornece diversas rotinas de geração de cortes padrão. Essa integração facilita a aplicação de técnicas clássicas de corte com relativa simplicidade (CBC, 2025).

Quanto à política de planos de corte, o CBC implementa cortes clássicos, como os cortes de Gomory, *knapsack covers* e cortes de clique, todos gerenciados por seu *cut manager*. Apesar de seu desempenho poder ser ajustado por meio de parâmetros como o número máximo de rodadas de corte, o CBC geralmente não oferece o mesmo grau de customização avançada presente em *solvers* como o SCIP. Ainda assim, sua simplicidade e estabilidade fazem dele uma opção sólida para diversas aplicações práticas (CBC, 2025).

CBC é gratuito e de código aberto, sendo uma opção prática para aplicações em que simplicidade e custo são fatores decisivos. Seu desempenho é razoável em instâncias de tamanho moderado, mas tende a ser inferior à de Gurobi e CPLEX em problemas maiores ou mais complexos. É adequado para usuários que não necessitam de suporte avançado a personalizações, mas ainda assim desejam uma solução robusta e sem custo.

2.3.5 Considerações Práticas

A decisão sobre qual *solver* utilizar deve considerar diversos aspectos práticos relacionados às características do problema e às condições do ambiente de desenvolvimento. Um dos principais fatores é o tamanho e a complexidade do problema. *Solvers* comerciais como Gurobi e CPLEX tendem a ser mais indicados para problemas de grande escala, devido ao seu alto desempenho e recursos avançados. Já *solvers* como SCIP e CBC podem ser mais apropriados para instâncias menores ou para projetos com foco em pesquisa, onde a flexibilidade e o custo-benefício são mais relevantes.

Outro ponto importante diz respeito ao orçamento e ao modelo de licenciamento. Gurobi e CPLEX são soluções comerciais que exigem licenças pagas, embora ofereçam versões gratuitas para uso acadêmico sob determinadas condições. Em contrapartida, SCIP e CBC são *solvers* de código aberto e podem ser utilizados livremente, sem custos de licenciamento, o que os torna especialmente atrativos para instituições educacionais e empresas de menor porte.

A flexibilidade de implementação também deve ser considerada, sobretudo em projetos de pesquisa. O SCIP se destaca nesse aspecto, oferecendo ampla liberdade para a customização de heurísticas, cortes e estratégias de ramificação. O CBC também permite certo grau de modificação, embora em menor escala. Por outro lado, Gurobi e CPLEX, apesar de menos flexíveis nesse sentido, disponibilizam APIs robustas e bem documentadas, facilitando sua integração e uso em diferentes plataformas e linguagens.

Por fim, a integração com ferramentas e linguagens de programação é um fator decisivo, especialmente em ambientes voltados ao desenvolvimento de aplicações. Todos os *solvers* analisados oferecem suporte para linguagens como Python, C++ e Java. No entanto, a compatibilidade com bibliotecas de ciência de dados, frameworks de modelagem matemática e sistemas de otimização pode influenciar diretamente na escolha do *solver* mais apropriado para cada caso.

2.3.6 Relevância da Comparação

No contexto do VRP e de suas variantes, especialmente o HVRP, realizar comparações entre *solvers* não é apenas desejável, mas necessário. Cada *solver* explora o espaço de soluções de forma distinta, impactando diretamente o tempo de execução, a qualidade das soluções e a escalabilidade dos algoritmos. Este trabalho busca justamente preencher uma lacuna na literatura ao realizar uma avaliação comparativa de desempenho entre *solvers*, fornecendo subsídios para decisões fundamentadas em aplicações práticas e pesquisas futuras.

2.4 Comentários Gerais

Este capítulo estabelece os conceitos fundamentais para o desenvolvimento deste trabalho, descrevendo o Problema de Roteamento de Veículos (VRP), suas principais variantes, em especial o HVRP, e apresentando as metodologias clássicas de solução (exatas e heurísticas). Discutimos a importância dos planos de corte na aceleração de métodos exatos como *Branch-and-Cut* e *Branch-Cut-and-Price*, bem como a adoção de heurísticas e metaheurísticas na busca de soluções de alta qualidade em tempo reduzido.

Além disso, analisamos as principais ferramentas computacionais (*solvers*) disponíveis no mercado, tanto comerciais (Gurobi e CPLEX) quanto *open-source* (SCIP e CBC), destacando seus pontos fortes e limitações.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma revisão da literatura pertinente ao problema de roteamento de veículos com frota heterogênea. Primeiramente, na Seção 3.1, são discutidos trabalhos correlatos, abrangendo tanto a literatura específica sobre o Problema de Roteamento de Veículos com Frota Heterogênea (HVRP) quanto estudos que comparam o desempenho dos *solvers* em problemas de otimização. Posteriormente, na Seção 3.2, é apresentada uma síntese do que foi aprendido com esses trabalhos e como eles contribuem para a motivação e o escopo desta pesquisa.

A literatura abordada nesta seção é dividida em duas frentes principais: estudos focados no HVRP, seus modelos e métodos de solução, e estudos que realizam comparações empíricas entre diferentes pacotes computacionais para problemas de otimização.

3.1 Revisão da Literatura sobre HVRP

O HVRP, uma variante do problema de Roteamento de Veículos (VRP) com frota heterogênea, foi formalmente introduzida por Golden *et al*. (1984) e adiciona a complexidade de gerenciar veículos com diferentes capacidades, custos fixos e custos variáveis por distância. Essa característica torna o HVRP particularmente relevante para aplicações práticas, onde as frotas raramente são homogêneas (Hoff *et al*., 2010); (Koç; Bektaş; Jabali; Laporte, 2016).

A literatura sobre HVRP é vasta e aborda diferentes aspectos do problema. Yaman (2005) realizou uma análise teórica aprofundada de diferentes formulações matemáticas para o HVRP, baseadas em restrições de eliminação de subrotas, como as de Miller; Tucker; Zemlin (1960), e em fluxo de commodities, propondo também novas desigualdades válidas para fortalecer os modelos.

Pessoa *et al.* (2018) propuseram um algoritmo *Branch-Cut-and-Price* (BCP) avançado para o HVRP, adaptando técnicas eficientes originalmente desenvolvidas para frotas homogêneas. O trabalho deles introduziu cortes específicos (*Extended Capacity Cuts*, *Rank-1 Cuts* com memória dependente do tipo de veículo) e demonstrou a capacidade de resolver instâncias com até 200 clientes, superando significativamente os resultados de algoritmos exatos anteriores como os de Baldacci e Mingozzi (2009).

Além dos métodos exatos, uma grande quantidade de heurísticas e metaheurísticas foi desenvolvida para o HVRP, dada a sua complexidade NP-difícil. Koç; Bektaş; Jabali; Laporte (2016) realizaram uma revisão abrangente cobrindo 30 anos de pesquisa sobre HVRP, classificando os trabalhos por tipo de problema (FSM - *Fleet Size and Mix*, HF - *Heterogeneous Fleet*) e métodos de solução (heurísticas clássicas, busca local, busca tabu, algoritmos genéticos, VNS, colônia de formigas, etc.), além de comparar o desempenho de várias metaheurísticas em instâncias de *benchmark*.

Variantes do HVRP também são exploradas, como o HVRP com Janelas de Tempo (HVRPTW) (Paraskevopoulos *et al.*, 2008); (Koç; Bektaş; Jabali; Laporte, 2016), Multi-Depot HVRP (MDHVRP) (Bettinelli *et al.*, 2011), *Site Dependent* HVRP (SDHVRP), onde veículos só podem atender subconjuntos específicos de clientes (Pessoa *et al*., 2018), e o Green HVRP ou Pollution-Routing Problem com frota heterogênea (Koç *et al*., 2014).

Esses trabalhos demonstram a complexidade inerente ao HVRP e a contínua busca por modelos matemáticos mais fortes e algoritmos de solução mais eficientes, tanto exatos quanto heurísticos.

3.2 Comparação de Ferramentas Computacionais

A escolha da ferramenta computacional (*solver*) adequada é crucial para a implementação prática de modelos de otimização. A literatura que compara *solvers*, embora não tão extensa quanto a de modelos específicos, fornece *insights* valiosos sobre o desempenho relativo dessas ferramentas.

Gleixner *et al*. (2021), ao compilar a biblioteca MIPLIB 2017, realizaram extensos testes comparando diversos *solvers* comerciais e não comerciais em um conjunto amplo de instâncias de Programação Inteira Mista (MIP). Um resultado chave foi a constatação de que nenhum *solver* domina universalmente; o desempenho varia significativamente dependendo da estrutura do problema e da instância específica. Isso reforça a necessidade de avaliações empíricas direcionadas a classes específicas de problemas.

Meindl e Templ (2012) compararam *solvers* comerciais (CPLEX, Gurobi, XPRESS) e de código aberto (SCIP, CBC, GLPK, LP SOLVE) para o problema de supressão de células em tabelas estatísticas, concluindo que, para aquele problema, os *solvers* comerciais geralmente apresentavam melhor desempenho. Oliveira, Schmidt e Silva (2017) compararam CPLEX e Gurobi para um problema de transporte com custo fixo, observando que o Gurobi obteve melhores resultados, um achado que divergiu do estudo de Meindl e Templ (2012) para um problema diferente, evidenciando a dependência do desempenho em relação ao problema tratado.

Mais recentemente, Castellucci, Leao e Bernardes (2024) avaliaram *solvers* (CBC, SCIP, CPLEX, Gurobi) para o problema de empacotamento de retângulos. Concluíram que, embora o Gurobi fosse geralmente mais rápido para provar otimalidade, a qualidade das soluções factíveis obtidas pelos *solvers* gratuitos (especialmente SCIP) em tempo limitado era competitiva (aproximadamente 98% da solução do Gurobi), questionando o custo-benefício das licenças comerciais em cenários onde a prova de otimalidade não é estritamente necessária ou o tempo computacional é limitado.

Esses estudos comparativos, embora focados em problemas distintos do HVRP, estabelecem um precedente importante: o desempenho dos *solvers* é contextual e depende fortemente das características do problema e das instâncias. Nenhum estudo encontrado na literatura realizou uma comparação sistemática e dedicada de *solvers* especificamente para o HVRP usando formulações padrão.

3.3 Comentários Gerais

A revisão da literatura realizada neste capítulo revela dois pontos principais. Primeiro, o HVRP é um problema de otimização combinatória complexo e de grande relevância prática, com uma variedade de modelos matemáticos e métodos de solução propostos ao longo dos anos. Algoritmos exatos, como os baseados em BCP, têm avançado significativamente, mas ainda enfrentam desafios computacionais em instâncias de grande porte, enquanto heurísticas continuam sendo essenciais para aplicações práticas.

Segundo, a literatura sobre comparação de *solvers*, embora existente para outros problemas, demonstra que o desempenho dessas ferramentas é altamente dependente do contexto (problema e instância). Nenhum *solver* é universalmente superior, e há um *trade-off* conhecido entre o custo (licenças comerciais) e o desempenho (velocidade, robustez), bem como entre o desempenho e a flexibilidade/customização (*solvers* de código-aberto como SCIP).

Diante desse cenário, este trabalho se propõe a contribuir de modo ainda não encontrado na literatura científica: a avaliação comparativa do desempenho de *solvers* comerciais (Gurobi, CPLEX) e não comerciais/código aberto (SCIP, CBC) aplicados diretamente a instâncias do HVRP, utilizando formulações matemáticas padrão da literatura. Enquanto trabalhos anteriores compararam *solvers* em problemas genéricos ou outros problemas específicos de otimização, ou desenvolveram algoritmos complexos para HVRP usando um *solver* específico, este trabalho foca no desempenho relativo dos *solvers* em si quando confrontados com a estrutura particular do HVRP.

A contribuição deste trabalho reside em fornecer evidências empíricas que podem auxiliar pesquisadores e praticantes na escolha da ferramenta computacional mais adequada para resolver o HVRP, considerando o balanço entre custo, tempo computacional e qualidade da solução (ou prova de otimalidade). Ao analisar como diferentes implementações do *Branch-and-Cut* lidam com a heterogeneidade da frota e outras características das instâncias do HVRP, espera-se oferecer informações sobre os pontos fortes e fracos de cada *solver* neste contexto.

4 DEFINIÇÃO DO PROBLEMA

O Problema de Roteamento de Veículos, introduzido por Dantzig e Ramser (1959), é um dos problemas fundamentais em Pesquisa Operacional. Seu objetivo central é determinar um conjunto ótimo de rotas para uma frota de veículos atender a um conjunto de clientes distribuídos geograficamente, partindo e retornando a um ou mais depósitos, enquanto se minimiza uma função objetivo (geralmente, custo total ou distância percorrida) e se respeita um conjunto de restrições operacionais. Dada sua aplicabilidade em inúmeros setores logísticos e de distribuição, o VRP gerou uma vasta gama de variantes para acomodar as complexidades do mundo real (Koç; Bektaş; Jabali; Laporte, 2016).

Uma dessas variantes de grande importância prática é o Problema de Roteamento de Veículos com Frota Heterogênea. Diferente do VRP clássico, que assume uma frota composta por veículos idênticos, o HVRP considera que a frota é composta por veículos de diferentes tipos. Cada tipo de veículo pode possuir características distintas, como capacidade de carga Q_k, custo fixo de utilização F_k (se aplicável), custo variável por distância percorrida c_{ijk}, e, em algumas extensões, até mesmo restrições sobre quais clientes podem ser atendidos (Koç; Bektaş; Jabali; Laporte, 2016; Yaman, 2005). A necessidade de decidir não apenas quais clientes cada rota atenderá, mas também qual tipo de veículo será alocado a cada rota, adiciona complexidade ao problema.

4.1 Formulação Matemática do HVRP

Para formalizar o HVRP, considera-se um grafo $G = (N, A)$, onde $N = \{1, ..., n\}$ é o conjunto de n clientes e $N' = N \cup \{0\}$ em que o nó 0 representa o depósito. Cada cliente $i \in N$ possui uma demanda $q_i > 0$ ($q_0 = 0$). O conjunto de arcos A é definido como $A = \{(i, j) \in N' \times N \mid i \neq j \text{ e } q_i + q_j \leq Q_1\}$, a restrição $q_i + q_j \leq Q_1$ assegura que dois nós consecutivos em uma rota possam ser atendidos por ao menos um tipo de veículo com capacidade suficiente, considerando que os veículos estão ordenados de forma decrescente em relação à capacidade.

A frota é composta por um conjunto $M = \{1, ..., m\}$ de tipos de veículos. Para cada tipo de veículo $k \in M$, cada um com capacidade Q_k. Custos associados incluem um custo fixo F_k e um custo variável c_{ijk} para percorrer o arco $(i, j) \in A$ com um veículo do tipo k.

Neste trabalho foi adotada a formulação matemática denominada $HVRP_4$ proposta por Yaman (2005), que utiliza variáveis desagregadas por tipo de veículo e modela as restrições de capacidade e eliminação de sub-rotas através de variáveis de carga contínuas.

Definição das Variáveis de Decisão

y_{ijk} - Variável binária. $y_{ijk} = 1$ se o arco $(i, j) \in A$ é percorrido por um veículo do tipo $k \in M$; se não $y_{ijk} = 0$.

a_{ik} - Variável binária. $a_{ik} = 1$ se o nó $i \in N'$ é o último cliente visitado por um veículo do tipo $k \in M$ naquela rota; se não: $a_{ik} = 0$.

b_{ik} - Variável binária. $b_{ik} = 1$ se o nó $i \in N'$ é visitado por um veículo do tipo $k \in M$, mas não é o último cliente da rota; Se não: $b_{ik} = 0$.

v_{ik} - Variável contínua. Representa a carga acumulada no veículo do tipo $k \in M$ até (e incluindo) o nó $i \in N'$.

Seja $K_i = \{k \in M : q_i \leq Q_k\}$ o conjunto de tipos de veículo capazes de atender a demanda do cliente i. Seja $A^K = \{(i, j, k) : (i, j) \in A, k \in M\}$ o conjunto de arcos estendidos por tipo de veículo. Seja C_{ik} o custo associado ao término da rota no cliente i com um veículo tipo k (incluindo custo fixo F_k e custo de retorno ao depósito c_{i0k}). A função objetivo e as restrições são dadas por (4.1)–(4.12).

Função Objetivo

$$\min \sum_{i \in N'} \sum_{k \in K_i} C_{ik} a_{ik} + \sum_{(i,j,k) \in A^K} c_{ijk} y_{ijk} \tag{4.1}$$

Sujeito às restrições

$$\sum_{k \in K_i} (a_{ik} + b_{ik}) = 1 \quad \forall i \in N' \tag{4.2}$$

$$\sum_{j:(j,i,k) \in A^K} y_{jik} = a_{ik} + b_{ik} \quad \forall i \in N', k \in K_i \tag{4.3}$$

$$\sum_{j:(i,j,k) \in A^K} y_{ijk} = b_{ik} \quad \forall i \in N', k \in K_i \tag{4.4}$$

$$\sum_{j:(0,j,k) \in A^K} y_{0jk} = \sum_{i \in N': k \in K_i} a_{ik} \quad \forall k \in M \tag{4.5}$$

$$v_{jk} \geq v_{ik} + q_j(a_{jk} + b_{jk}) - Q_k(a_{ik} + b_{ik} - y_{ijk}) \quad \forall (i,j,k) \in A^K, i \neq 0 \tag{4.6}$$

$$v_{ik} \geq q_i(a_{ik} + b_{ik}) + \sum_{j:(j,i,k) \in A^K} q_j y_{jik} \quad \forall i \in N', k \in K_i \tag{4.7}$$

$$v_{ik} \leq Q_k(a_{ik} + b_{ik}) \quad \forall i \in N', k \in K_i \tag{4.8}$$

$$y_{ijk} \in \{0,1\} \quad \forall (i,j,k) \in A^K \tag{4.9}$$

$$a_{ik} \in \{0,1\} \quad \forall i \in N', k \in K_i \tag{4.10}$$

$$b_{ik} \in \{0,1\} \quad \forall i \in N', k \in K_i \tag{4.11}$$

$$v_{ik} \geq 0 \quad \forall i \in N', k \in K_i \tag{4.12}$$

A Formulação $HVRP_4$ (4.1) - (4.12), conforme apresentado por Yaman (2005), é intrinsecamente um modelo de Programação Linear Inteira Mista. Ela utiliza variáveis binárias (y_{ijk}, a_{ik}, b_{ik}) para capturar as decisões discretas de roteamento e atribuição de status aos nós, e variáveis contínuas (v_{ik}) para modelar a carga acumulada nos veículos. Todas as restrições e a função objetivo são lineares em relação a essas variáveis.

4.2 Análise Detalhada

A Formulação $HVRP_4$, proposta por Yaman (2005) e adotada neste trabalho (formulação (4.1) a (4.12)), modela o Problema de Roteamento de Veículos com Frota Heterogênea utilizando variáveis desagregadas por tipo de veículo e restrições baseadas em carga acumulada para garantir capacidade e conectividade.

Função Objetivo (4.1)

$$\min \sum_{i \in N'} \sum_{k \in K_i} C_{ik} a_{ik} + \sum_{(i,j,k) \in A^K} c_{ijk} y_{ijk} \quad (4.1)$$

Esta função define o objetivo do problema: minimizar o custo total da operação. O custo é composto por duas partes. A primeira parcela,

$$\sum_{i \in N'} \sum_{k \in K_i} C_{ik} a_{ik}$$

representa a soma dos custos associados ao término das rotas. C_{ik} inclui o custo fixo F_k do veículo tipo k e o custo de viagem c_{i0k} do último cliente i de volta ao depósito (nó 0). A variável binária a_{ik} ativa esse custo se, e somente se, o cliente i for o último a ser visitado por um veículo do tipo k. A segunda parcela,

$$\sum_{(i,j,k) \in A^K} c_{ijk} y_{ijk}$$

representa a soma dos custos variáveis de viagem. c_{ijk} é o custo de percorrer o arco (i, j) com um veículo do tipo k. A variável binária y_{ijk} ativa esse custo se, e somente se, o arco (i, j) for utilizado por um veículo do tipo k.

Atendimento ao Cliente - Restrições (4.2)

$$\sum_{k \in K_i} (a_{ik} + b_{ik}) = 1 \quad \forall i \in N' \quad (4.2)$$

Estas restrições garantem que cada cliente i (pertencente ao conjunto N') seja visitado exatamente uma vez. Para cada cliente i, a soma das variáveis binárias a_{ik} (indicando que i é o último cliente na rota do veículo k) e b_{ik} (indicando que i é um cliente intermediário na rota do veículo k), sobre todos os tipos de veículos k capazes de atender sua demanda ($k \in K_i$), deve ser igual a 1. Isso força cada cliente a pertencer a exatamente uma rota e a ter um status definido (último ou intermediário) nela.

Conservação de Fluxo - Restrições (4.3) e (4.4)

$$\sum_{j:(j,i,k) \in A^K} y_{jik} = a_{ik} + b_{ik} \quad \forall i \in N', k \in K_i \quad (4.3)$$

Estas restrições estabelecem a relação entre os arcos que chegam a um cliente e o status desse cliente. Ela assegura que, se um cliente i é visitado por um veículo do tipo k (ou seja, se $a_{ik} + b_{ik} = 1$), então deve existir exatamente um arco (j, i) percorrido por um veículo do tipo k chegando a esse cliente i. A soma sobre todas as possíveis origens j das variáveis y_{jik} deve ser igual a 1 se o nó i for visitado pelo tipo k, e 0 caso contrário.

$$\sum_{j:(i,j,k) \in A^K} y_{ijk} = b_{ik} \quad \forall i \in N', k \in K_i \quad (4.4)$$

Complementar às restrições anteriores, estas garantem a continuidade da rota a partir de um cliente intermediário. Se um cliente i é visitado por um veículo do tipo k e não é o último cliente dessa rota (ou seja, $b_{ik} = 1$), então deve existir exatamente um arco (i, j) percorrido por um veículo do tipo k saindo desse cliente i. Se i for o último cliente ($a_{ik} = 1$, logo $b_{ik} = 0$), nenhum arco do tipo k poderá sair dele em direção a outro cliente.

Equilíbrio do Depósito - Restrições (4.5)

$$\sum_{j:(0,j,k)\in A^K} y_{0jk} = \sum_{i\in N':k\in K_i} a_{ik} \quad \forall k \in M \quad (4.5)$$

Estas restrições garantem que, para cada tipo de veículo k, o número de veículos que saem do depósito (lado esquerdo, soma de y_{0jk} para todos os primeiros clientes j) seja igual ao número de rotas desse tipo que terminam (lado direito, soma de a_{ik} sobre todos os clientes i que podem ser o último da rota k). Isso assegura o equilíbrio do fluxo de rotas para cada tipo de veículo.

Sequência da Carga - MTZ - Restrições (4.6)

$$v_{jk} \geq v_{ik} + q_j(a_{jk} + b_{jk}) - Q_k(a_{ik} + b_{ik} - y_{ijk}) \quad \forall (i,j,k) \in A^K, i \neq 0 \quad (4.6)$$

Estas são as principais restrições do tipo Miller-Tucker-Zemlin (MTZ) adaptada por Yaman (2005). Ela cumpre duas funções: elimina sub-rotas que não incluem o depósito e rastreia a carga acumulada. Se o arco (i,j) é percorrido por um veículo do tipo k ($y_{ijk} = 1$) e i foi visitado por esse veículo ($a_{ik} + b_{ik} = 1$), a restrição se torna $v_{jk} \geq v_{ik} + q_j$, garantindo que a carga ao chegar em j (v_{jk}) seja pelo menos a carga ao chegar em i (v_{ik}) mais a demanda coletada em j (q_j, pois $a_{jk} + b_{jk} = 1$). Se o arco (i,j,k) não é usado, o termo $-Q_k(...)$ se torna um valor grande negativo ou zero, tornando a restrição redundante e permitindo que a carga "salte" entre nós não conectados, o que é fundamental para eliminar ciclos indesejados.

Limite Inferior da Carga - MTZ - Restrições (4.7)

$$v_{ik} \geq q_i(a_{ik} + b_{ik}) + \sum_{j:(j,i,k)\in A^K} q_j y_{jik} \quad \forall i \in N', k \in K_i \quad (4.7)$$

Esta restrição define um limite inferior para a variável de carga v_{ik}. Ela estabelece que a carga acumulada até o cliente i (inclusive) deve ser, no mínimo, a demanda do próprio cliente i (ativada por $a_{ik} + b_{ik}$) somada às demandas q_j do cliente que foi visitado imediatamente antes de i na mesma rota do tipo k (ativada por y_{jik}). Isso ajuda a fortalecer a relaxação linear do modelo.

Limite Superior da Carga (Capacidade) - MTZ - Restrições (4.8)

$$v_{ik} \leq Q_k(a_{ik} + b_{ik}) \quad \forall i \in N', k \in K_i \quad (4.8)$$

Estas restrições impõem o limite de capacidade Q_k do veículo. Se o cliente i é visitado por um veículo do tipo k ($a_{ik} + b_{ik} = 1$), então a carga acumulada v_{ik} não pode exceder a capacidade Q_k. Se o cliente i não é visitado por um veículo do tipo k ($a_{ik} + b_{ik} = 0$), a restrição força $v_{ik} \leq 0$. Como v_{ik} também deve ser não negativa (restrições (4.12)), isso implica $v_{ik} = 0$ quando o nó não é visitado por aquele tipo de veículo.

Domínio Binário - Restrições (4.9), (4.10), (4.11) e Domínio Contínuo (Não-negatividade) - Restrição (4.12)

$$y_{ijk} \in \{0, 1\} \quad \forall (i, j, k) \in A^K \quad (4.9)$$

$$a_{ik} \in \{0, 1\} \quad \forall i \in N', k \in K_i \quad (4.10)$$

$$b_{ik} \in \{0, 1\} \quad \forall i \in N', k \in K_i \quad (4.11)$$

Estas restrições definem que as variáveis de decisão y_{ijk}, a_{ik} e b_{ik} só podem assumir valores 0 ou 1, representando decisões de "sim" ou "não" (usar um arco, ser o último nó, ser um nó intermediário).

$$v_{ik} \geq 0 \quad \forall i \in N', k \in K_i \quad (4.12)$$

Estas restrições definem que as variáveis de carga acumulada v_{ik} devem ser valores contínuos e não podem ser negativos.

5 METODOLOGIA

Este capítulo detalha os procedimentos metodológicos adotados para a avaliação comparativa dos pacotes computacionais na resolução do Problema de Roteamento de Veículos com Frota Heterogênea. A metodologia abrange a seleção e configuração das ferramentas, a descrição do conjunto de instâncias utilizadas, os parâmetros experimentais definidos e as métricas de avaliação empregadas, bem como a estrutura geral das avaliações a serem realizadas.

5.1 Seleção e Configuração dos *Solvers*

Para conduzir uma avaliação abrangente, foram selecionados quatro *solvers* amplamente reconhecidos e utilizados tanto na academia quanto na indústria para a resolução de problemas de Programação Linear Inteira Mista, classe à qual pertence a formulação $HVRP4$ adotada neste trabalho. A seleção buscou incluir tanto opções comerciais de alto desempenho quanto alternativas de código-aberto, permitindo uma análise do *trade-off* entre custo de licenciamento e desempenho computacional. Os *solvers* selecionados foram: Gurobi Optimizer na versão 10.0, CPLEX na versão 22.1, SCIP na versão 9.0, CBC na versão 2.10. A criação do modelo nos pacotes foi feita com a biblioteca PuLP na versão 2.7.0 e Python 3.10.

Todos os *solvers* foram utilizados com suas configurações padrão, exceto pela definição de um limite de tempo computacional e pela restrição ao uso de um único *thread* de processamento, conforme detalhado na Seção 5.3. A escolha pelas configurações padrão visa simular um cenário comum onde o usuário não realiza um ajuste fino extensivo dos parâmetros, permitindo uma comparação mais direta do desempenho usando as configurações padrões de cada ferramenta para o HVRP. A implementação e execução dos modelos será feita utilizando a linguagem de programação Python com a biblioteca de modelagem PuLP.

5.2 Conjunto de Dados e Instâncias Utilizadas

O conjunto de instâncias para a avaliação experimental foi selecionado a partir da literatura clássica sobre HVRP, especificamente dos trabalhos de Golden *et al.* (1984) e Christofides e Eilon (1969), embora tenham sido propostas há décadas, ainda são utilizadas para avaliação de algoritmos (Koç; Bektaş; Jabali; Laporte, 2016).

As características da frota (tipos de veículos, suas capacidades e custos) para cada instância foram extraídas diretamente das tabelas apresentadas no Apêndice do artigo de Golden *et al.* (1984). Foram considerados os problemas numerados de #3 a #20 naquele trabalho, cobrindo diferentes combinações de número de clientes e características de frota. Por exemplo, os problemas #3 e #4 utilizam os primeiros 20 nós do problema de 50 nós de Christofides e Eilon (1969) com configurações de frota específicas, enquanto os problemas #19 e #20 utilizam 100 nós com outras configurações. As tabelas nas imagens fornecidas detalham as capacidades e custos para cada tipo de veículo (A, B, C, etc.) em cada um desses 20 problemas.

As coordenadas (x, y) da localização dos clientes e as demandas (q) dos clientes, bem como a localização do depósito, foram extraídas do trabalho de Christofides e Eilon (1969), que por sua vez se baseou em fontes como Hayes, Dantzig & Ramser, Gaskell, e Clarke & Wright, além de propor novos problemas. Para cada um dos 20 problemas selecionados de Golden *et al.* (1984), os dados correspondentes de clientes (localização e demanda) foram identificados e utilizados.

Este conjunto combinado abrange instâncias com número de clientes variando de 12 a 100 e com 2 a 6 tipos diferentes de veículos na frota, proporcionando uma base diversificada para avaliar o desempenho dos *solvers* em diferentes escalas e níveis de heterogeneidade da frota.

5.3 Parâmetros Experimentais e Métricas de Avaliação

Para garantir uma comparação justa e significativa entre os *solvers*, foram definidos alguns parâmetros experimentais e métricas de avaliação.

Todos os testes foram executados na mesma máquina com as seguintes especificações: Processador Intel Core i7-7700 CPU @ 3.60GHz x 8, 16 GB de memória RAM, Sistema Operacional Ubuntu 22.04.4.

Foi estabelecido um limite de tempo máximo de 3600 segundos (1 hora) para a execução de cada instância por cada *solver*. Isso permite avaliar não apenas a capacidade de encontrar a solução ótima, mas também a qualidade da solução encontrada dentro de um prazo razoável.

Para isolar o desempenho do algoritmo do *solver* do benefício do paralelismo (que pode variar entre as implementações), todos os testes foram configurados para utilizar apenas um único *thread* de processamento.

Todos os *solvers* foram aplicados à mesma formulação matemática, a $HVRP4$ de Yaman (2006), descrita na Seção 4.1.

Métricas de Avaliação

O desempenho dos *solvers* será avaliado com base em três critérios principais. O primeiro é a qualidade da solução, que será mensurada pelo *gap* de otimalidade ou pelo valor da melhor solução factível encontrada durante o processo de otimização. Em segundo lugar, será considerado o tempo computacional, ou seja, o tempo total de execução necessário para que o *solver* alcance a melhor solução. O terceiro critério envolve a qualidade da relaxação linear do nó raiz da árvore de *Branch-and-Cut*.

5.4 Estrutura e Execução dos Testes

Primeiro, a formulação $HVRP4$ foi implementada em Python na biblioteca PuLP, com interfaces para os quatro *solvers*. Em seguida, foi realizada a leitura das instâncias selecionadas. A execução foi realizada em lote (*batch*), submetendo cada uma das instâncias a cada um dos quatro *solvers* sob os parâmetros experimentais definidos. As métricas de avaliação foram registradas automaticamente ao final de cada execução. Por fim, os dados coletados foram compilados para análise comparativa no capítulo de experimentos computacionais, utilizando tabelas e gráficos, para identificar padrões e diferenças significativas no desempenho dos *solvers* ao resolverem o HVRP com a formulação $HVRP4$.

Esta metodologia garante que a comparação seja realizada sob condições controladas, permitindo tirar conclusões embasadas sobre a eficácia de cada *solver* para o problema em estudo.

6 EXPERIMENTOS COMPUTACIONAIS E RESULTADOS

Este capítulo apresenta os resultados dos experimentos computacionais realizados para comparar o desempenho dos pacotes computacionais na resolução do Problema de Roteamento de Veículos com Frota Heterogênea (HVRP). Primeiramente, descreve-se a configuração experimental e o conjunto de instâncias utilizadas. Em seguida, apresenta-se uma análise detalhada dos resultados obtidos, com foco nas métricas de *gap* de otimalidade e tempo computacional. Por fim, discute-se os padrões observados e as implicações práticas dos resultados.

6.1 Experimentos Computacionais

O conjunto de instâncias utilizado foi extraído dos trabalhos clássicos de Golden et al. (1984) e Christofides e Eilon (1969). Foram considerados 24 problemas numerados de #3 a #20, disponíveis no Apêndice C, que cobrem diferentes escalas e configurações de frota. As instâncias abrangem desde casos com 10 clientes até instâncias com 100 clientes, com frotas heterogêneas compostas por 2 a 6 tipos diferentes de veículos. Esta diversidade permite avaliar o desempenho dos *solvers* em diferentes dimensões do problema.

As características da frota, incluindo capacidades, custos fixos e custos variáveis de cada tipo de veículo, foram extraídas diretamente do Apêndice de Golden et al. (1984). As coordenadas e demandas dos clientes foram obtidas de Christofides e Eilon (1969). Para as instâncias menores (10 clientes), foram testadas diferentes variações da configuração de frota, resultando em 12 instâncias neste grupo. Os demais grupos (20, 50, 75 e 100 clientes) contêm 4, 4, 2 e 2 instâncias, respectivamente.

Para possibilitar uma análise mais detalhada do comportamento dos *solvers* em instâncias de pequeno porte, onde métodos exatos têm maior probabilidade de provar a otimalidade, foram criadas variantes adicionais de algumas instâncias base através da seleção aleatória de subconjuntos de clientes. Especificamente, para as instâncias base #3, #4, #5 e #6, foram geradas três variantes cada (#X-10-A, #X-10-B e #X-10-C), nas quais foram selecionados aleatoriamente 10 clientes da

instância original. Esta abordagem mantém a estrutura geral do problema, incluindo a configuração de frota heterogênea e o depósito original, enquanto reduz significativamente o tamanho do espaço de busca. A utilização de múltiplas amostras aleatórias de cada instância base permite avaliar a robustez do desempenho dos *solvers* frente a diferentes distribuições espaciais de clientes, evitando conclusões baseadas em características particulares de uma única configuração. Esta estratégia de amostragem resulta em 12 instâncias de 10 clientes, proporcionando uma base estatisticamente mais sólida para comparação entre os *solvers* em problemas de menor escala.

Os resultados detalhados das execuções de todas as instâncias em cada *solver* estão apresentados na Tabela 3 do Apêndice B, incluindo o limitante da relaxação linear do nó raiz, a melhor solução encontrada, o melhor limitante inferior, o *gap* de otimalidade e o tempo de execução.

6.2 Análise dos Resultados

A Tabela 1 apresenta um resumo consolidado dos resultados obtidos para cada grupo de instâncias, organizados por número de clientes. Para cada *solver*, são reportadas as métricas de *gap* de otimalidade (média e mediana)que representa a diferença percentual entre a melhor solução encontrada e o melhor limitante inferior, indicando a distância da solução em relação ao ótimo, tempo computacional (média e mediana), número de soluções ótimas comprovadas e número de soluções viáveis encontradas.

Tabela 1 – Resumo dos resultados por tamanho de instância. TL indica que o valor médio (ou mediano) foi igual ao limite de tempo de 1 hora.

Clientes	Métrica		CBC	SCIP	CPLEX	Gurobi
10	**Gap (%)**	**Média**	67.97	48.44	26.08	22.91
		Mediana	61.83	51.98	19.94	22.68
	Tempo (s)	**Média**	*TL*	3322.57	2835.07	2821.25
		Mediana	*TL*	*TL*	*TL*	*TL*
	Nº Ótimas		0	3	5	5
	Nº Viáveis		12	12	12	12

20	**Gap (%)**	**Média**	87.79	85.38	85.10	84.68
		Mediana	88.34	85.64	85.33	84.83
	Tempo (s)	**Média**	*TL*	*TL*	*TL*	*TL*
		Mediana	*TL*	*TL*	*TL*	*TL*
	Nº Ótimas		0	0	0	0
	Nº Viáveis		4	4	4	4
50	**Gap (%)**	**Média**	91.21	89.80	88.51	87.95
		Mediana	89.56	87.91	86.05	85.68
	Tempo (s)	**Média**	*TL*	*TL*	*TL*	*TL*
		Mediana	*TL*	*TL*	*TL*	*TL*
	Nº Ótimas		0	0	0	0
	Nº Viáveis		4	4	4	4
75	**Gap (%)**	**Média**	86.90	85.57	81.45	75.93
		Mediana	86.90	85.57	81.45	75.93
	Tempo (s)	**Média**	*TL*	*TL*	*TL*	*TL*
		Mediana	*TL*	*TL*	*TL*	*TL*
	Nº Ótimas		0	0	0	0
	Nº Viáveis		2	2	2	2
100	**Gap (%)**	**Média**	94.63	96.13	93.72	90.70
		Mediana	94.63	96.13	93.72	90.70
	Tempo (s)	**Média**	*TL*	*TL*	*TL*	*TL*
		Mediana	*TL*	*TL*	*TL*	*TL*
	Nº Ótimas		0	0	0	0
	Nº Viáveis		2	2	2	2

6.2.1 Desempenho nas Instâncias com 10 Clientes

Para as instâncias com 10 clientes, observa-se uma clara distinção entre o desempenho dos *solvers* comerciais e das alternativas de código aberto. O Gurobi e o CPLEX demonstraram capacidade superior, conseguindo provar a otimalidade em 5 das 12 instâncias testadas, enquanto o SCIP obteve 3 soluções ótimas e o CBC não conseguiu provar a otimalidade em nenhuma instância dentro do limite de tempo.

Em termos de *gap* de otimalidade médio, o Gurobi apresentou o melhor resultado com 22,91%, seguido pelo CPLEX com 26,08%, SCIP com 48,44% e CBC com 67,97%. A mediana do *gap* seguiu padrão similar, com Gurobi (22,68%) e CPLEX (19,94%) mantendo valores substancialmente menores que SCIP (51,98%) e CBC (61,83%). Esta diferença indica que os *solvers* comerciais conseguem explorar mais efetivamente o espaço de soluções mesmo quando não alcançam a prova de otimalidade dentro do tempo limite.

A Figura 1 apresenta os boxplots do *gap* de otimalidade para cada *solver* considerando todas as instâncias testadas. Observa-se que CBC e SCIP apresentam *gaps* medianos mais elevados, mas com menor variabilidade nos resultados. Em contraste, CPLEX e Gurobi demonstram maior consistência no gap mediano, porém com mais dispersão nos resultados.

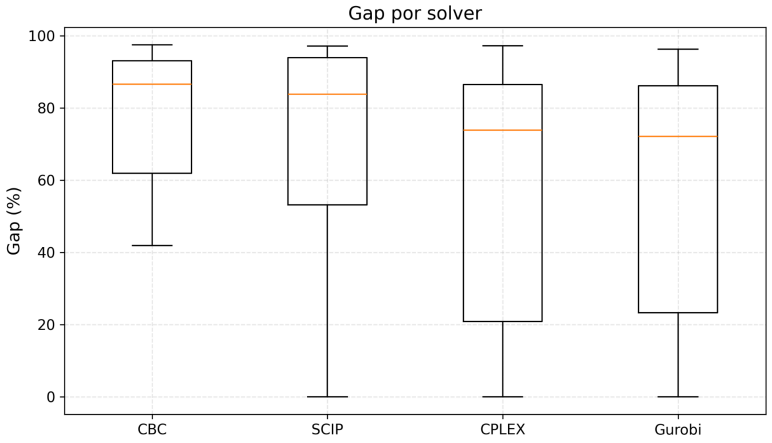

Figura 1 – *Boxplots* do *gap* de otimalidade para cada *solver* considerando todas as instâncias.

Quando analisamos especificamente as instâncias menores, conforme ilustrado na Figura 2, a superioridade dos *solvers* comerciais se torna ainda mais evidente. O Gurobi demonstra o menor *gap* médio (22,91%) e a menor variabilidade, seguido de perto pelo CPLEX (26,08%). Notavelmente, mesmo nas instâncias menores onde seria esperado um desempenho mais homogêneo entre os *solvers*, o CBC manteve um *gap* médio acima de 67%.

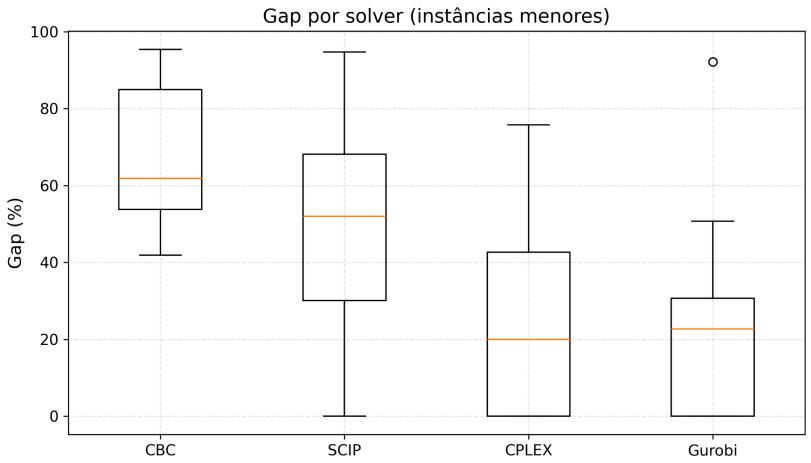

Figura 2 – *Boxplots* do gap de otimalidade para instâncias com 10 clientes.

O comportamento temporal dos *solvers* é ilustrado na Figura 3. Para as instâncias com 10 clientes, o CBC apresentou o pior desempenho temporal, com mediana em 3600 segundos (limite de tempo) e um *outlier* próximo a 3600 segundos, indicando que sempre esgotou o tempo disponível. O SCIP demonstrou distribuição temporal mais favorável, com diversos casos resolvidos antes do limite, mas ainda com mediana próxima ao máximo. Os *solvers* comerciais, CPLEX e Gurobi, mostraram tempos significativamente menores, com medianas abaixo de 3000 segundos e diversos *outliers* em tempos mais baixos, indicando resoluções rápidas em várias instâncias.

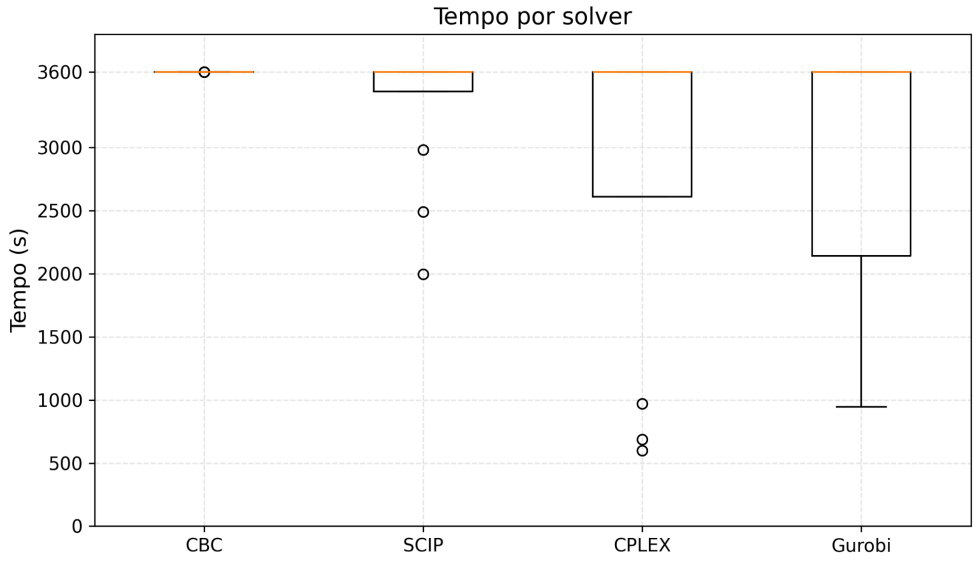

Figura 3 – *Boxplots* do tempo de execução para instâncias com 10 clientes.

A Figura 4 apresenta a curva de instâncias resolvidas ao longo do tempo, permitindo visualizar a evolução do desempenho de cada *solver*. O CPLEX destaca-se ao resolver 3 instâncias em aproximadamente 1200 segundos, mantendo este patamar até cerca de 3000 segundos, quando resolve 2 instâncias adicionais. O Gurobi segue trajetória similar, resolvendo a primeira instância em torno de 1200 segundos e alcançando 5 instâncias resolvidas até 2700 segundos. O comportamento do SCIP é mais gradual, começando apenas após 3000 segundos e alcançando 3 instâncias ao final do período. O CBC não conseguiu provar a otimalidade de nenhuma instância no tempo limite.

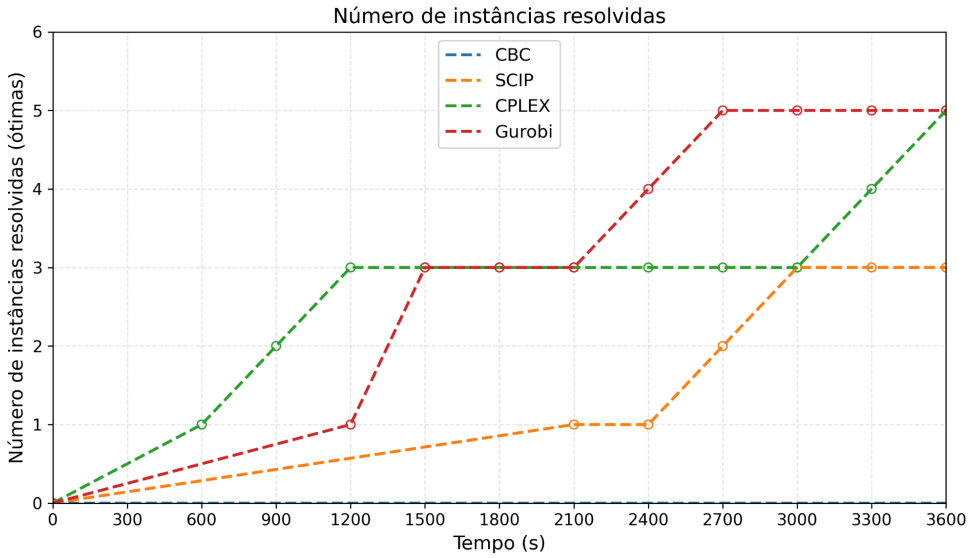

Figura 4 – Número de instâncias resolvidas (cumulativo) ao longo do tempo.

Esses resultados indicam que, para instâncias pequenas do HVRP, os *solvers* comerciais apresentam vantagem significativa tanto em termos de qualidade da solução quanto de eficiência computacional. O CPLEX demonstra capacidade de convergência mais rápida nos estágios iniciais, enquanto o Gurobi mantém desempenho consistente ao longo de todo o período de execução.

6.2.2 Desempenho nas Instâncias Maiores

À medida que o tamanho das instâncias aumenta, a dificuldade do problema se intensifica drasticamente. Para todas as instâncias com 20 ou mais clientes, nenhum dos *solvers* conseguiu provar a otimalidade dentro do limite de tempo de 1

hora. Além disso, todos os *solvers* atingiram o limite de tempo na maioria das execuções, resultando em medianas de tempo iguais a 3600 segundos.

No entanto, diferenças significativas permanecem na qualidade das soluções viáveis encontradas. Para as instâncias com 20 clientes, os *gaps* médios foram relativamente altos para todos os *solvers*: CBC (87,79%), SCIP (85,38%), CPLEX (85,10%) e Gurobi (84,68%). Embora a diferença absoluta entre os *solvers* seja menor neste grupo, o Gurobi mantém a liderança em qualidade de solução.

Para as instâncias com 50 clientes, o padrão se mantém, com *gaps* médios superiores a 87% para todos os *solvers*. O Gurobi novamente apresenta o melhor resultado (87,95%), seguido por CPLEX (88,51%), SCIP (89,80%) e CBC (91,21%).

Um comportamento interessante é observado nas instâncias com 75 clientes. Aqui, a vantagem do Gurobi se acentua, com *gap* médio de 75,93%, enquanto os demais *solvers* permanecem acima de 81%. Esta diferença de aproximadamente 6 pontos percentuais sugere que, para instâncias de porte médio-grande, as técnicas implementadas no Gurobi são particularmente efetivas.

Para as instâncias com 100 clientes, que representam os casos mais desafiadores testados, o SCIP surpreendentemente apresenta o pior desempenho (96,13%), enquanto o Gurobi mantém o melhor resultado (90,70%), seguido por CPLEX (93,72%) e CBC (94,63%).

A Figura 5 apresenta uma comparação consolidada do *gap* médio de todos os solvers considerando todas as instâncias testadas. A superioridade consistente do Gurobi é evidente, com *gap* médio de 54,11%, seguido pelo CPLEX (56,57%), SCIP (68,56%) e CBC (78,94%).

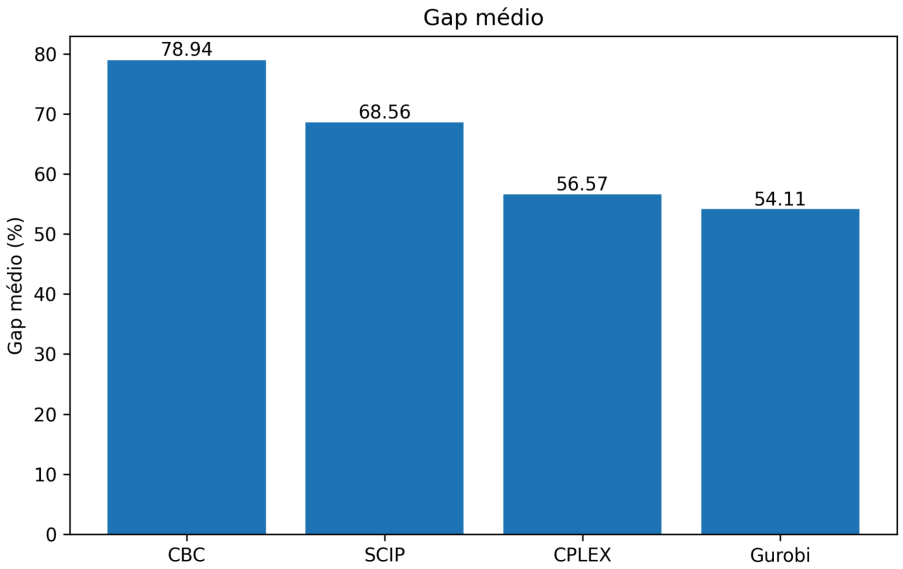

Figura 5 – *Gap* médio para todos os solvers considerando todas as instâncias.

Para uma análise mais detalhada das instâncias menores, a Figura 6 apresenta o *gap* médio considerando apenas as instâncias com 10 clientes. Neste subconjunto, a vantagem dos *solvers* comerciais é ainda mais pronunciada. O Gurobi alcança *gap* médio de 22,91%, enquanto o CPLEX obtém 26,09%. Os *solvers* de código aberto apresentam *gaps* significativamente maiores: SCIP com 48,44% e CBC com 67,97%.

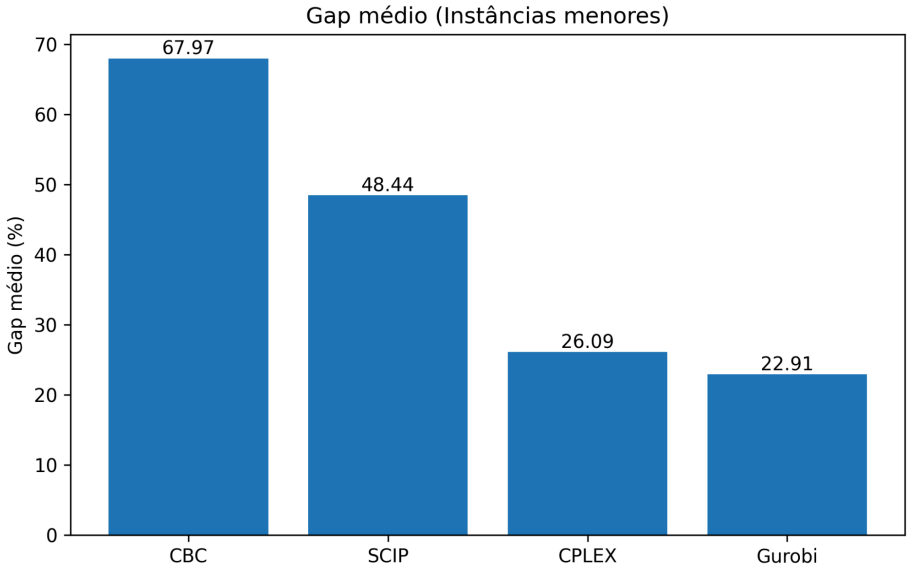

Figura 6 – *Gap* médio para todos os *solvers* nas instâncias com 10 clientes.

6.2.3 Análise da Relaxação Linear do Nó Raiz

Um aspecto importante para compreender as diferenças de desempenho entre os *solvers* é a qualidade do limitante obtido na relaxação linear do nó raiz da árvore de *Branch-and-Cut*. Este limitante inicial influencia significativamente a eficiência do processo de busca, pois determina o limite inferior inicial usado para podar subproblemas durante a exploração da árvore.

Analisando os dados dos experimentos, observa-se que os solvers comerciais tendem a obter limitantes na raiz ligeiramente superiores aos *solvers* de código aberto. Por exemplo, na instância #3-10-A, o Gurobi obteve limitante na raiz de 152,95, enquanto o CPLEX obteve 151,72, o SCIP 151,87 e o CBC 151,71. Embora as diferenças absolutas sejam pequenas neste caso, elas podem se acumular ao longo do processo de *Branch-and-Cut*.

Um padrão interessante acontece nas instâncias menores da série #4, onde todos os *solvers* obtiveram limitantes na raiz muito similares. Na instância #4-10-A, por exemplo, todos começaram com limitante próximo a 128,60. Nestes casos, a diferença de desempenho observada deve-se provavelmente às estratégias de corte e *branching* implementadas durante a exploração da árvore, e não à qualidade da relaxação inicial.

Por outro lado, nas instâncias menores da série #6, observam-se limitantes na raiz relativamente fracos para todos os *solvers* (em torno de 139-162), mas com grandes diferenças na capacidade de convergência. Isso sugere que estas instâncias possuem características estruturais que tornam a relaxação linear menos efetiva, exigindo técnicas mais sofisticadas de corte para melhorar os limitantes.

6.2.4 Análise do *Virtual Best Bound*

Para compreender melhor as diferenças de desempenho entre os *solvers*, foi construído um *Virtual Best Bound* (VBB), que representa o melhor limitante inferior obtido entre todos os *solvers* para cada instância. A Tabela 2 apresenta uma comparação detalhada dos resultados obtidos por cada *solver* em relação ao VBB para as instâncias com 10 clientes.

O *gap* do *Virtual Best Bound* representa a diferença percentual entre a melhor solução encontrada por um solver e o melhor limitante inferior obtido por qualquer *solver* no conjunto de testes, permitindo avaliar a qualidade relativa dos limitantes produzidos, ele é calculado através da seguinte fórmula $\text{Gap VBB} = \frac{\text{UB}_s - \text{VBB}}{\text{VBB}} \times 100\%$.

Tabela 2 – Comparação dos resultados com *Virtual Best Bound* para instâncias com 10 clientes.

		CBC		SCIP		CPLEX		Gurobi	
Instância	**VBB**	**LI**	**Gap (%)**	**LI**	**Gap (%)**	**LI**	**Gap (%)**	**LI**	**Gap (%)**
#3-10-A	528,66	204,51	61,32	241,06	54,40	433,09	18,08	528,66	0,00
#3-10-B	518,90	242,57	53,25	303,19	41,57	518,90	0,00	408,48	21,28
#3-10-C	422,10	208,99	50,49	253,77	39,88	422,10	0,00	375,41	11,06
#3	266,46	242,71	8,91	260,74	2,15	262,80	1,37	266,46	0,00
#4-10-A	3.202,99	1.861,03	41,90	3.202,99	0,00	3.202,99	0,00	3.202,73	0,01
#4-10-B	3.171,27	555,78	82,47	3.171,27	0,00	3.171,27	0,00	3.171,18	0,00
#4-10-C	3.707,57	1.713,29	53,79	3.707,57	0,00	3.707,57	0,00	3.707,30	0,01
#4	259,99	242,69	6,65	255,32	1,80	259,23	0,29	259,99	0,00
#5-10-A	403,56	210,70	47,79	244,35	39,45	403,56	0,00	301,43	25,31
#5-10-B	320,65	215,20	32,89	255,76	20,24	320,65	0,00	319,08	0,49
#5-10-C	550,22	211,45	61,57	277,58	49,55	550,22	0,00	378,22	31,26
#5	268,33	245,71	8,43	261,47	2,56	261,51	2,54	268,33	0,00
#6-10-A	3.304,66	200,00	93,95	231,60	92,99	1.293,51	60,86	3.304,66	0,00
#6-10-B	2.699,07	210,79	92,19	248,74	90,78	850,22	68,50	2.699,07	0,00
#6-10-C	915,51	206,95	77,40	242,55	73,51	915,51	0,00	295,46	67,73
#6	259,93	248,57	4,37	255,68	1,64	258,51	0,55	259,93	0,00
#13	389,68	370,09	5,03	385,48	1,08	389,14	0,14	389,68	0,00
#14	390,67	375,13	3,98	385,46	1,33	390,67	0,00	389,81	0,22
#15	424,79	395,32	6,94	418,27	1,53	424,79	0,00	423,11	0,40
#16	426,00	400,21	6,05	418,32	1,80	426,00	0,00	425,96	0,01
#17	537,14	490,34	8,71	516,21	3,90	537,00	0,03	537,14	0,00
#18	536,81	489,52	8,81	502,38	6,41	536,81	0,00	536,81	0,00
#19	605,51	501,67	17,15	588,95	2,73	593,51	1,98	605,51	0,00
#20	604,12	506,28	16,20	588,01	2,67	594,34	1,62	604,12	0,00

A análise do VBB revela padrões importantes sobre o desempenho relativo dos *solvers*. O Gurobi conseguiu igualar o VBB em 11 das 24 instâncias, demonstrando sua capacidade de encontrar os melhores limitantes inferiores. O CPLEX igualou o VBB em 11 instâncias, apresentando desempenho equivalente ao Gurobi neste aspecto. O SCIP conseguiu igualar o VBB em 3 instâncias, especificamente nas da série #4. O CBC não igualou o VBB em nenhuma instância, ficando consistentemente abaixo dos demais *solvers*.

Este resultado sugere que a superioridade do Gurobi e do CPLEX não se limita apenas à velocidade de convergência, mas também à qualidade dos limitantes obtidos através da relaxação linear e dos cortes gerados durante o processo de *Branch-and-Cut*. Os *solvers* comerciais parecem implementar planos de corte mais efetivos e estratégias de *branching* mais sofisticadas, resultando em limitantes inferiores significativamente melhores.

Um caso particularmente ilustrativo é a instância #6-10-B, onde o CBC obteve limitante inferior de apenas 210,79 com *gap* de 92,22%, enquanto o Gurobi alcançou 2.699,07 com *gap* de apenas 0,01%, praticamente provando a otimalidade. Esta diferença de mais de 90 pontos percentuais no *gap* evidencia a disparidade na capacidade de exploração do espaço de soluções entre os *solvers*.

Para as instâncias da série #4 e #5, observa-se que o SCIP conseguiu igualar o VBB em várias ocasiões, demonstrando que, em certas configurações de problema, as alternativas de código aberto podem ser competitivas com as soluções comerciais. No entanto, esta competitividade não é consistente ao longo de todas as instâncias.

Um resultado interessante é observado na instância #6-10-C, onde até mesmo o Gurobi apresentou dificuldades significativas, obtendo *gap* de 92,19%. Esta foi a instância mais desafiadora entre as de 10 clientes, onde mesmo os *solvers* comerciais encontraram limitações. O CPLEX, por sua vez, conseguiu provar a otimalidade desta instância, demonstrando que características específicas da estrutura do problema podem favorecer diferentes implementações algorítmicas.

6.3 Discussão

Os resultados experimentais evidenciam diferenças substanciais no desempenho dos quatro *solvers* testados para o HVRP. De modo geral, os *solvers* comerciais Gurobi e CPLEX superaram consistentemente as alternativas de código aberto CBC e SCIP em todas as métricas avaliadas.

O Gurobi demonstrou ser o *solver* mais eficaz, alcançando os menores *gaps* de otimalidade e os melhores tempos de execução em praticamente todas as categorias de instâncias. Esta superioridade é particularmente evidente nas instâncias menores, onde o Gurobi conseguiu provar a otimalidade em 5 das 12 instâncias, enquanto também apresentou os melhores resultados em termos de qualidade de limitantes inferiores.

O CPLEX apresentou desempenho próximo ao do Gurobi, especialmente nas instâncias menores, onde também provou a otimalidade em 5 instâncias. Em algumas situações específicas, como nas instâncias da série #3 e #5, o CPLEX conseguiu igualar ou superar o Gurobi. Isso sugere que a escolha entre esses dois *solvers* comerciais pode depender de características específicas da instância do problema.

Entre os *solvers* de código aberto, o SCIP mostrou-se superior ao CBC na maioria dos cenários, particularmente nas instâncias maiores. Embora ambos tenham apresentado *gaps* significativamente maiores que os *solvers* comerciais, o SCIP demonstrou maior capacidade de convergência, conseguindo provar a otimalidade em 3 instâncias de 10 clientes, enquanto o CBC não alcançou nenhuma prova de otimalidade.

Um aspecto relevante observado é que a diferença de desempenho entre *solvers* comerciais e de código aberto tende a se acentuar com o aumento do tamanho da instância. Para as instâncias com 10 clientes, embora os solvers comerciais já demonstrem superioridade, a diferença é menos pronunciada. À medida que o problema cresce para 20, 50, 75 e 100 clientes, o *gap* de desempenho se amplia consideravelmente.

A análise do VBB fornece *insights* sobre os mecanismos responsáveis por essas diferenças. O fato de Gurobi e CPLEX conseguirem igualar o VBB com muito mais frequência sugere que esses *solvers* implementam técnicas de geração de planos de corte e estratégias de *branching* mais sofisticadas. Estas técnicas resultam em limitantes inferiores mais apertados, reduzindo o espaço de busca e acelerando a convergência para a solução ótima.

Para tomadores de decisão que necessitam resolver instâncias do HVRP, os resultados sugerem que a escolha do *solver* deve considerar não apenas o custo de licenciamento, mas também a escala do problema e os requisitos de tempo de resposta. Para problemas menores onde soluções rápidas são necessárias, os *solvers* comerciais, especialmente o Gurobi, são claramente superiores. Para cenários onde o tempo de execução não é crítico e o custo é um fator limitante, as alternativas de código aberto podem fornecer soluções viáveis razoáveis, embora com maior incerteza sobre a qualidade de otimalidade.

É importante ressaltar que estes resultados são específicos para o HVRP para uma das formulações de Yaman (2006) e para as versões dos *solvers* testadas. Diferentes formulações ou atualizações nos *solvers* podem alterar as conclusões. Além disso, nenhum ajuste fino de parâmetros foi realizado, e é possível que configurações personalizadas possam melhorar o desempenho, especialmente dos *solvers* de código aberto que oferecem maior flexibilidade de customização.

6.3.1 Padrões por Série de Instâncias

Uma análise mais detalhada revela comportamentos distintos dos *solvers* em diferentes séries de instâncias. As instâncias da série #4 apresentaram características que favoreceram todos os *solvers*, com CPLEX, Gurobi e SCIP conseguindo provar a otimalidade nas três variantes de 10 clientes. Esta série apresentou valores de solução ótima significativamente mais altos (acima de 3.000), sugerindo uma estrutura de custos diferente das demais.

Por outro lado, as instâncias da série #6 foram particularmente desafiadoras para todos os *solvers*, com *gaps* elevados mesmo para os comerciais. A exceção

notável foi a instância #6-10-B, onde Gurobi alcançou desempenho quase ótimo. Esta variabilidade dentro da mesma série sugere que pequenas mudanças na configuração da frota ou distribuição de clientes podem ter impacto significativo na dificuldade de resolução.

As instâncias da série #5 apresentaram comportamento intermediário, com CPLEX conseguindo resolver uma das três variantes (#5-10-C) mas encontrando dificuldades nas demais. Já as instâncias da série #3 mostraram-se acessíveis ao CPLEX mas desafiadoras para o Gurobi, exemplificando como características específicas do problema podem influenciar a eficácia de diferentes implementações algorítmicas.

Estes padrões sugerem que a estrutura da rede, a configuração da frota heterogênea e a distribuição espacial dos clientes interagem de maneiras complexas para determinar a dificuldade computacional do problema. Uma investigação futura sobre quais características específicas das instâncias correlacionam-se com o desempenho de cada *solver* poderia fornecer diretrizes mais precisas para a seleção de ferramentas computacionais.

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou uma avaliação comparativa do desempenho de quatro *solvers* de Programação Linear Inteira Mista amplamente utilizados, CBC 2.10, SCIP 9.0, CPLEX 22.1 e Gurobi 10.0, aplicados ao Problema de Roteamento de Veículos com Frota Heterogênea (HVRP). Utilizando uma formulação matemática de Yaman (2006) e instâncias clássicas da literatura de Golden et al. (1984) e Christofides e Eilon (1969), buscou-se preencher uma lacuna na literatura, que carece de análises sistemáticas comparando *solvers* comerciais e de código aberto especificamente para este problema.

Os resultados experimentais revelaram diferenças substanciais no desempenho dos *solvers* avaliados. Os *solvers* comerciais, Gurobi e CPLEX, demonstraram clara superioridade tanto na qualidade das soluções quanto no tempo computacional, provando a otimalidade em múltiplas instâncias de 10 clientes e mantendo gaps de otimalidade significativamente menores mesmo nas instâncias maiores. Entre os *solvers* de código aberto, o SCIP apresentou desempenho superior ao CBC, embora ambos tenham ficado consideravelmente abaixo dos *solvers* comerciais. A análise do *Virtual Best Bound* evidenciou que as vantagens dos *solvers* comerciais derivam principalmente de técnicas mais sofisticadas de geração de planos de corte e estratégias de *branching*, resultando em convergência mais rápida e limitantes inferiores de melhor qualidade.

Para tomadores de decisão, os resultados indicam que *solvers* comerciais são justificáveis quando soluções rápidas e de alta qualidade são necessárias, especialmente sob restrições rigorosas de tempo. Alternativamente, *solvers* de código aberto podem ser adequados quando o tempo não é crítico ou quando soluções viáveis razoáveis são suficientes. É importante ressaltar que, para muitas aplicações práticas do HVRP, especialmente em instâncias de grande porte, o uso direto de *solvers* pode não ser suficiente para obter soluções de qualidade em tempo razoável, sendo necessário o desenvolvimento de métodos mais eficientes, como heurísticas, metaheurísticas ou abordagens híbridas que combinem métodos exatos com técnicas aproximadas.

Finalmente, cabe destacar que os resultados apresentados são específicos para o HVRP com a formulação de Yaman (2006) e as versões testadas dos *solvers*, utilizando configurações padrão e execução em thread única. Diferentes problemas, formulações ou ajustes nos parâmetros dos *solvers* podem apresentar resultados distintos.

7.1 Trabalhos Futuros

Diversos caminhos podem ser explorados como continuação deste trabalho, abrangendo tanto extensões da avaliação experimental quanto desenvolvimentos metodológicos e aplicações práticas.

Uma primeira direção natural seria expandir a avaliação experimental para incluir versões mais recentes dos *solvers* testados. A evolução constante destes pacotes computacionais, com novas versões frequentemente introduzindo melhorias algorítmicas significativas, torna importante reavaliar periodicamente o desempenho relativo entre eles. Seria igualmente valioso incluir outros *solvers* como *Xpress* e *HiGHS*, ampliando o espectro de comparação. Embora instâncias grandes sejam extremamente desafiadoras para métodos exatos, avaliar até que ponto cada *solver* consegue encontrar soluções viáveis de qualidade para problemas com 200 ou mais clientes forneceria *insights* sobre os limites práticos destas ferramentas. Adicionalmente, comparar o desempenho dos *solvers* utilizando outras formulações matemáticas do HVRP disponíveis na literatura, como as de Golden et al. (1984) ou Pessoa et al. (2009), ou formulações baseadas em fluxo e partição de conjuntos, permitiria avaliar se as diferenças observadas são específicas da formulação de Yaman (2006) ou se generalizam para outras representações do problema. A extensão da análise para variantes do HVRP, como HVRP com janelas de tempo, *multi-depot* HVRP, ou HVRP com coleta e entrega, também representaria uma contribuição valiosa.

Uma segunda linha de investigação promissora envolve o ajuste sistemático de parâmetros e configurações dos *solvers*. Este trabalho utilizou as configurações padrão de todos os *solvers*, mas é bem conhecido que ajustes finos podem melhorar substancialmente o desempenho, especialmente para solvers de código aberto que oferecem maior flexibilidade de customização. Explorar sistematicamente o espaço

de parâmetros para identificar configurações otimizadas para o HVRP poderia reduzir significativamente o *gap* de desempenho observado entre *solvers* comerciais e gratuitos. Investigar quais famílias de desigualdades válidas, ou cortes, são mais efetivas para o HVRP em cada *solver* constituiria outro estudo relevante, potencialmente levando ao desenvolvimento de estratégias de corte customizadas. O impacto do uso de múltiplas *threads* na redução do tempo computacional e na qualidade das soluções também merece atenção, considerando que diferentes *solvers* podem explorar paralelismo de maneiras distintas. Por fim, investigar o benefício de fornecer soluções iniciais de boa qualidade, obtidas através de heurísticas, para acelerar a convergência dos métodos exatos através de técnicas de *warm-start* representa uma abordagem híbrida promissora.

Um terceiro aspecto importante seria a análise sistemática das características das instâncias e sua relação com o desempenho dos *solvers*. Este trabalho observou que diferentes séries de instâncias apresentaram padrões distintos de dificuldade, mas não investigou profundamente quais características estruturais são responsáveis por estes padrões. Identificar quais atributos das instâncias, como tamanho da frota, heterogeneidade dos custos, distribuição espacial dos clientes, ou razão entre demanda e capacidade, correlacionam-se com o desempenho de cada *solver* permitiria predizer qual ferramenta seria mais apropriada para novos problemas. Esta análise poderia ser formalizada através do desenvolvimento de modelos de *machine learning* que, dadas as características de uma nova instância, recomendam o *solver* mais apropriado, contribuindo para a área emergente de seleção automática de algoritmos. Estudar mais profundamente quais estruturas matemáticas tornam certas instâncias particularmente difíceis ou fáceis para métodos exatos também forneceria insights teóricos valiosos sobre a complexidade computacional do HVRP.

Aspectos computacionais mais detalhados também merecem atenção. Além do tempo computacional, avaliar o consumo de memória RAM de cada *solver* torna-se especialmente relevante para instâncias grandes, onde limitações de memória podem se tornar o gargalo principal. A avaliação do desempenho dos *solvers* em ambientes computacionais distribuídos, como *clusters* de computação, seria importante para organizações que possuem infraestrutura de alto desempenho

disponível. Uma análise detalhada dos logs de execução dos *solvers*, examinando quantos nós foram explorados na árvore de *Branch-and-Cut*, quantos cortes foram gerados, e como os limitantes evoluem ao longo do tempo, poderia revelar com maior profundidade as estratégias algorítmicas que levam às diferenças de desempenho observadas. Esta análise mais fina permitiria não apenas entender melhor por que certos *solvers* superam outros, mas também identificar oportunidades de melhorias algorítmicas.

Por fim, a replicação desta metodologia comparativa para outros problemas de otimização combinatória relevantes, como o problema do caixeiro viajante com janelas de tempo, problemas de dimensionamento de lotes, ou problemas de *scheduling*, contribuiria para uma compreensão mais ampla sobre as forças e fraquezas relativas dos diferentes *solvers* em diversos contextos. A construção de uma biblioteca abrangente de *benchmarks* comparativos entre *solvers* para múltiplos problemas beneficiaria toda a comunidade de Pesquisa Operacional.

REFERÊNCIAS

KOÇ, Ç.; BEKTAŞ, T.; JABALI, O.; LAPORTE, G. Thirty years of heterogeneous vehicle routing. **European Journal of Operational Research**, v. 249, n. 1, 2016.

Gurobi, 2025. *Gurobi Optimizer Reference Manual*.
Disponível em: https://www.gurobi.com/documentation. Acessado em: 5/04/2025

IBM CPLEX, 2025. *IBM ILOG CPLEX Optimization Studio Documentation*.
Disponível em: https://www.ibm.com/docs/en/icos. Acessado em: 5/04/2025

SCIP, 2025. *SCIP Optimization Suite Documentation*.
Disponível em: https://www.scipopt.org/doc/html/. Acessado em: 5/04/2025

CBC, 2025. *CBC User Guide - COIN-OR Branch and Cut Solver*.
Disponível em: https://github.com/coin-or/Cbc. Acessado em: 5/04/2025

LOURENÇO, Helena R.; MARTIN, Olivier C.; STÜTZLE, Thomas. Iterated local search. In: GLOVER, Fred; KOCHENBERGER, Gary A. (Ed.). **Handbook of metaheuristics.** Boston: Springer, 2003. p. 321–353.

HOLLAND, John H. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.** Ann Arbor: University of Michigan Press, 1975.

GLOVER, Fred. **Future paths for integer programming and links to artificial intelligence. Computers & Operations Research**, v. 13, n. 5, p. 533–549, 1986.

WOLSEY, Laurence A. **Integer programming**. 2. ed. Hoboken: John Wiley & Sons, 2021.

GOLDBARG, Marco Cesar; LUNA, Henrique Pacca Loureiro. **Otimização combinatória e programação linear: modelos e algoritmos.** Rio de Janeiro: Elsevier, 2005.

ARENALES, M. N. *et al.* **Pesquisa Operacional**. 1. ed. Rio de Janeiro: Elsevier, 2007.

BALDACCI, R.; MINGOZZI, A. **A unified exact method for solving different classes of vehicle routing problems.** Mathematical Programming, v. 120, n. 2, p. 347–380, 2009.

BETTINELLI, A.; CESELLI, A.; RIGHINI, G. **A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows.** Transportation Research Part C: Emerging Technologies, v. 19, n. 5, p. 723–740, 2011.

CASTELLUCCI, P. B.; LEAO, A. A.; BERNARDES, E. D. **Comparação entre pacotes computacionais baseados em branch-and-cut para o empacotamento de retângulos.** Pesquisa Operacional para o Desenvolvimento, v. 18, p. 1–18, 2024.

DANTZIG, G. B.; RAMSER, J. H. **The truck dispatching problem.** Management Science, v. 6, n. 1, p. 80–91, 1959.

GLEIXNER, A. *et al.* MIPLIB 2017: **Data-driven compilation of the 6th mixed-integer programming library.** Mathematical Programming Computation, v. 13, n. 3, p. 443–490, 2021.

GOLDEN, B. L.; ASSAD, A. A.; LEVY, L.; GHEYSENS, F. **The fleet size and mix vehicle routing problem.** Computers & Operations Research, v. 11, n. 1, p. 49–66, 1984.

HOFF, A. *et al.* **Industrial aspects and literature survey: Fleet composition and routing.** Computers & Operations Research, v. 37, n. 12, p. 2041–2061, 2010.

KOÇ, Ç. *et al.* **The fleet size and mix pollution-routing problem.** Transportation Research Part B: Methodological, v. 70, p. 239–254, 2014.

LAPORTE, G. **Fifty years of vehicle routing.** Transportation Science, v. 43, n. 4, p. 408–416, 2009.

MEINDL, B.; TEMPL, M. **Analysis of commercial and free and open source solvers for linear optimization problems.** Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS, v. 20, 2012.

MILLER, C. E.; TUCKER, A. W.; ZEMLIN, R. A. **Integer programming formulations and traveling salesman problems.** Journal of the ACM, v. 7, n. 4, p. 326–329, 1960.

OLIVEIRA, C. R. de; SCHMIDT, C. E.; SILVA, A. C. da. **Comparação entre o desempenho de dois solvers de otimização na resolução de problemas de transporte com custo fixo.** In: Congresso de Métodos Numéricos em Engenharia CMN2017. [S.l.: s.n.], 2017. v. 3, p. 5.

PARASKEVOPOULOS, D. C. *et al.* **A reactive variable neighbourhood tabu search for the heterogeneous fleet vehicle routing problem with time windows.** Journal of Heuristics, v. 14, n. 5, p. 425–455, 2008.

PESSOA, A.; SADYKOV, R.; UCHOA, E. **Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems.** European Journal of Operational Research, v. 270, n. 2, p. 530–543, 2018.

YAMAN, H. **Formulations and valid inequalities for the heterogeneous vehicle routing problem.** Mathematical Programming, v. 106, n. 2, p. 365–390, 2005.

A. CÓDIGO DESENVOLVIDO

```
import math
import pulp
import networkx as nx
import matplotlib.pyplot as plt


def read_hvrp_instance(filepath: str):
    coords, demands = {}, {}
    vehicle_types = {}
    reading_coords = reading_demands = reading_vtypes = False
    depot = None

    with open(filepath, "r", encoding="utf-8") as f:
        for raw in f:
            ln = raw.strip()
            if not ln:
                continue
            if ln.startswith("NODE_COORD_SECTION"):
                reading_coords, reading_demands, reading_vtypes =
True, False, False
                continue
            if ln.startswith("DEMAND_SECTION"):
                reading_coords, reading_demands, reading_vtypes =
False, True, False
                continue
            if ln.startswith("VEHICLE_TYPES"):
                reading_coords, reading_demands, reading_vtypes =
False, False, True
                continue
            if ln.startswith("END_VEHICLE_TYPES"):
                reading_vtypes = False
                continue
            if ln.startswith("DEPOT_SECTION"):
                reading_coords = reading_demands = reading_vtypes =
False
                continue
            if ln == "EOF":
                break

            if reading_coords:
                parts = ln.split()
```

```
                if len(parts) != 3:
                    continue
                i, x, y = parts
                coords[int(i)] = (float(x), float(y))
            elif reading_demands:
                parts = ln.split()
                if len(parts) != 2:
                    continue
                i, q = parts
                demands[int(i)] = int(q)
            elif reading_vtypes:
                if ln.startswith("#"):
                    continue
                parts = ln.split()
                if len(parts) != 4:
                    continue
                k, cap, fixed, var = parts
                vehicle_types[k] = {
                    "capacity": float(cap),
                    "fixed_cost": float(fixed),
                    "var_cost": float(var),
                }
            else:
                try:
                    val = int(ln)
                    if val != -1:
                        depot = val
                except ValueError:
                    pass

    if depot is None:
        raise ValueError("DEPOT_SECTION ausente ou inválida no
arquivo .vrp")
    return coords, demands, depot, vehicle_types


def euclidean(i, j, coords):
    (x1, y1), (x2, y2) = coords[i], coords[j]
    return math.hypot(x1 - x2, y1 - y2)


def build_hvrp_model(instance_path: str) -> pulp.LpProblem:
```

```
    coords, demands, depot, Kp = read_hvrp_instance(instance_path)

    nodes = sorted(coords.keys())
    clients = [i for i in nodes if i != depot]
    M_types = sorted(Kp.keys())

    K_i = {i: [k for k, par in Kp.items() if demands[i] <=
par["capacity"]] for i in clients}

    def cdist(i, j, k):
        return Kp[k]["var_cost"] * euclidean(i, j, coords)

    Y_index = []
    Y_out_dep = {k: [] for k in M_types}
    Y_in_dep = {k: [] for k in M_types}
    Y_ijk_by_k = {k: [] for k in M_types}

    for k in M_types:
        for j in clients:
            if k in K_i[j]:
                tpl = (depot, j, k)
                Y_index.append(tpl)
                Y_out_dep[k].append(tpl)
                Y_ijk_by_k[k].append(tpl)

        for i in clients:
            if k in K_i[i]:
                tpl = (i, depot, k)
                Y_index.append(tpl)
                Y_in_dep[k].append(tpl)
                Y_ijk_by_k[k].append(tpl)

        for i in clients:
            if k not in K_i[i]:
                continue
            for j in clients:
                if i == j or (k not in K_i[j]):
                    continue
                tpl = (i, j, k)
                Y_index.append(tpl)
                Y_ijk_by_k[k].append(tpl)
```

```
    cijk = {(i, j, k): cdist(i, j, k) for (i, j, k) in Y_index}

    Cik = {}
    for i in clients:
        for k in K_i[i]:
            Cik[(i, k)] = Kp[k]["fixed_cost"] + cdist(i, depot, k)

    mdl = pulp.LpProblem("HVRP4_MIP", pulp.LpMinimize)

    y = pulp.LpVariable.dicts("y", Y_index, lowBound=0, upBound=1,
cat="Binary")

    a_index = [(i, k) for i in clients for k in K_i[i]]
    b_index = [(i, k) for i in clients for k in K_i[i]]
    a = pulp.LpVariable.dicts("a", a_index, lowBound=0, upBound=1,
cat="Binary")
    b = pulp.LpVariable.dicts("b", b_index, lowBound=0, upBound=1,
cat="Binary")

    v = pulp.LpVariable.dicts("v", ((i, k) for i in clients for k
in M_types), lowBound=0, cat="Continuous")

    mdl += (
        pulp.lpSum(Cik[(i, k)] * a[(i, k)] for (i, k) in a_index)
        + pulp.lpSum(cijk[(i, j, k)] * y[(i, j, k)] for (i, j, k)
in Y_index)
    ), "Objective"

    for i in clients:
        mdl += pulp.lpSum(a[(i, k)] for k in K_i[i]) +
pulp.lpSum(b[(i, k)] for k in K_i[i]) == 1, f"visit[{i}]"


    for i in clients:
        for k in K_i[i]:
            in_arcs = [(j, i, k) for (j, i2, k2) in Y_ijk_by_k[k]
if i2 == i and j != i]
            mdl += pulp.lpSum(y[(j, i, k)] for (j, _, _) in
in_arcs) == a[(i, k)] + b[(i, k)], f"in_deg[{i},{k}]"

    for i in clients:
        for k in K_i[i]:
```

```
            out_arcs = [(i, j, k) for (i2, j, k2) in Y_ijk_by_k[k]
if i2 == i and j != i]
            mdl += pulp.lpSum(y[(i, j, k)] for (_, j, _) in
out_arcs) == b[(i, k)], f"out_deg[{i},{k}]"

    for k in M_types:
        mdl += pulp.lpSum(y[(depot, j, k)] for (depot, j, _) in
Y_out_dep[k]) == \
               pulp.lpSum(a[(i, k)] for i in clients if k in
K_i[i]), f"depot_balance[{k}]"

    for k in M_types:
        Qk = Kp[k]["capacity"]
        for (i, j, kk) in Y_ijk_by_k[k]:
            if kk != k or i == depot:
                continue
            if (i in clients and k in K_i[i]) and (j in clients and
k in K_i[j]):
                mdl += v[(j, k)] >= v[(i, k)] + demands[j] * (a[(j,
k)] + b[(j, k)]) \
                       - Qk * (a[(i, k)] + b[(i, k)] - y[(i, j,
k)]), f"mtz_flow[{i},{j},{k}]"

    for i in clients:
        for k in K_i[i]:
            incoming = [(j, i, k) for (j, i2, k2) in Y_ijk_by_k[k]
if i2 == i and j != i]
            mdl += v[(i, k)] >= demands[i] * (a[(i, k)] + b[(i,
k)]) + \
                   pulp.lpSum(demands[j] * y[(j, i, k)] for (j, _,
_) in incoming), f"mtz_lb[{i},{k}]"

    for i in clients:
        for k in K_i[i]:
            Qk = Kp[k]["capacity"]
            mdl += v[(i, k)] <= Qk * (a[(i, k)] + b[(i, k)]),
f"cap[{i},{k}]"

    return mdl


def draw_solution(mdl, coords, depot, vehicle_types, thr=0.5):
```

```
    positions = {i: coords[i] for i in coords}

    arcs_by_type = {k: [] for k in vehicle_types.keys()}
    for var in mdl.variables():
        if not (var.name.startswith("y_") and var.varValue and
var.varValue > thr):
            continue
        inside = var.name[var.name.find("(") + 1:
var.name.rfind(")")]
        i_str, j_str, k_str = inside.split(",_")
        i = int(i_str)
        j = int(j_str)
        k = k_str.strip("'")
        arcs_by_type.setdefault(k, []).append((i, j))

    routes = []
    for k, arcs in arcs_by_type.items():
        if not arcs:
            continue
        succ = {}
        for (i, j) in arcs:
            succ[i] = j

        starts = [j for (i, j) in arcs if i == depot]
        veh_num = 1
        for s in starts:
            route = [depot, s]
            cur = s
            visited = set(route)
            while cur in succ and succ[cur] != depot:
                nxt = succ[cur]
                if nxt in visited:
                    break
                route.append(nxt)
                visited.add(nxt)
                cur = nxt
            route.append(depot)
            routes.append((f"{k}{veh_num}", route))
            veh_num += 1

    G = nx.DiGraph()
    for _, route in routes:
```

```
        for i, j in zip(route[:-1], route[1:]):
            G.add_edge(i, j)

    plt.figure(figsize=(7, 7))
    colors = ["tab:blue", "tab:green", "tab:orange", "tab:purple",
              "tab:brown", "tab:red", "tab:pink", "tab:olive",
"tab:cyan"]
    color_map = {}
    for idx, (veh, _) in enumerate(routes):
        color_map[veh] = colors[idx % len(colors)]

    nx.draw_networkx_nodes(G, pos=positions, nodelist=[depot],
                           node_color="red", node_size=650,
label="Depósito")
    nx.draw_networkx_nodes(G, pos=positions, nodelist=[i for i in
coords if i != depot],
                           node_color="lightblue", node_size=520)

    for veh, route in routes:
        edges = list(zip(route[:-1], route[1:]))
        nx.draw_networkx_edges(G, pos=positions, edgelist=edges,
                               edge_color=color_map[veh],
width=2.2,
                               arrows=True, arrowsize=15,
label=veh)

    labels = {i: str(i) for i in coords}
    nx.draw_networkx_labels(G, pos=positions, labels=labels,
font_size=9)

    from matplotlib.lines import Line2D
    handles = []
    for veh, _ in routes:
        handles.append(Line2D([0], [0], color=color_map[veh], lw=2,
label=veh))
    handles.insert(0, Line2D([0], [0], marker='o', color='w',
label='Depósito',
                             markerfacecolor='red', markersize=10))
    if handles:
        plt.legend(handles=handles, loc="best")

    plt.axis("off")
```

```
plt.tight_layout()
plt.show()
```

B. DADOS DOS RESULTADOS DOS EXPERIMENTOS

Tabela 3 - Resultado da execução das instâncias em cada *solver*.

Gurobi					
Id	**Root bound**	**Best solution**	**Best bound**	**Gap**	**Solve time**
problem-3-10-A.vrp	152.95	528.72	528.66	0.01	2371.80
problem-3-10-B.vrp	181.66	519.66	408.48	21.39	3600.00
problem-3-10-C.vrp	156.12	539.72	375.41	30.44	3600.00
problem-3.vrp	240.43	978.99	266.46	72.78	3600.00
problem-4-10-A.vrp	128.60	3202.99	3202.73	0.01	1309.40
problem-4-10-B.vrp	163.59	3171.27	3171.18	0.00	1456.35
problem-4-10-C.vrp	145.35	3707.57	3707.30	0.01	946.77
problem-4.vrp	240.43	6940.61	259.99	96.25	3600.00
problem-5-10-A.vrp	163.61	611.36	301.43	50.70	3600.00
problem-5-10-B.vrp	176.84	425.35	319.08	24.98	3600.00
problem-5-10-C.vrp	150.61	550.22	378.22	31.26	3600.00
problem-5.vrp	240.57	1008.59	268.33	73.40	3600.00
problem-6-10-A.vrp	139.57	4345.70	3304.66	23.96	3600.00
problem-6-10-B.vrp	159.24	2699.31	2699.07	0.01	2570.69
problem-6-10-C.vrp	159.51	3783.47	295.46	92.19	3600.00
problem-6.vrp	240.61	6997.96	259.93	96.29	3600.00
problem-13.vrp	386.96	2710.08	389.68	85.62	3600.01
problem-14.vrp	385.28	9654.51	389.81	95.96	3600.01
problem-15.vrp	419.41	2730.59	423.11	84.50	3600.01
problem-16.vrp	419.28	2985.78	425.96	85.73	3600.01
problem-17.vrp	536.35	1887.50	537.14	71.54	3600.01
problem-18.vrp	535.16	2726.74	536.81	80.31	3600.00
problem-19.vrp	590.35	9840.58	605.51	93.85	3600.05
problem-20.vrp	587.88	4849.99	604.12	87.54	3600.01
CPLEX					
Id	**Root bound**	**Best solution**	**Best bound**	**Gap**	**Solve time**
problem-3-10-A.vrp	151.72	528.72	433.09	18.09	3600.00
problem-3-10-B.vrp	181.66	518.90	518.90	0.00	3158.36
problem-3-10-C.vrp	154.87	539.72	422.10	21.79	3600.00
problem-3.vrp	240.43	983.18	262.80	73.27	3600.00
problem-4-10-A.vrp	128.60	3202.99	3202.99	0.00	686.67
problem-4-10-B.vrp	163.53	3171.27	3171.27	0.00	971.00

problem-4-10-C.vrp	144.81	3707.57	3707.57	0.00	599.03
problem-4.vrp	240.72	6886.28	259.23	96.24	3600.00
problem-5-10-A.vrp	162.55	611.36	403.56	33.99	3600.00
problem-5-10-B.vrp	176.84	425.35	320.65	24.61	3600.00
problem-5-10-C.vrp	150.55	550.22	550.22	0.00	3405.76
problem-5.vrp	240.43	1022.01	261.51	74.41	3600.00
problem-6-10-A.vrp	139.57	4345.70	1293.51	70.23	3600.00
problem-6-10-B.vrp	159.24	2699.31	850.22	68.50	3600.00
problem-6-10-C.vrp	162.78	3783.48	915.51	75.80	3600.00
problem-6.vrp	240.43	7311.61	258.51	96.46	3600.00
problem-13.vrp	385.72	2752.02	389.14	85.86	3600.00
problem-14.vrp	385.72	10756.80	390.67	96.37	3600.00
problem-15.vrp	417.67	2948.66	424.79	85.59	3600.00
problem-16.vrp	417.67	3092.70	426.00	86.23	3600.00
problem-17.vrp	536.49	2189.86	537.00	75.48	3600.00
problem-18.vrp	536.53	4266.57	536.81	87.42	3600.03
problem-19.vrp	592.91	21351.61	593.51	97.22	3600.01
problem-20.vrp	593.22	6073.46	594.34	90.21	3600.02
SCIP					
Id	**Root bound**	**Best solution**	**Best bound**	**Gap**	**Solve time**
problem-3-10-A.vrp	151.87	528.72	241.06	54.41	3600.00
problem-3-10-B.vrp	181.66	520.39	303.19	41.74	3600.00
problem-3-10-C.vrp	155.57	574.29	253.77	55.81	3600.00
problem-3.vrp	240.91	991.80	260.74	73.71	3600.00
problem-4-10-A.vrp	128.60	3202.99	3202.99	0.00	2492.68
problem-4-10-B.vrp	163.53	3171.27	3171.27	0.00	2981.41
problem-4-10-C.vrp	145.07	3707.57	3707.57	0.00	1996.68
problem-4.vrp	240.72	7354.60	255.32	96.53	3600.00
problem-5-10-A.vrp	162.86	619.66	244.35	60.57	3600.07
problem-5-10-B.vrp	176.84	426.69	255.76	40.06	3600.00
problem-5-10-C.vrp	155.84	550.22	277.58	49.55	3600.00
problem-5.vrp	241.51	1035.53	261.47	74.75	3600.01
problem-6-10-A.vrp	139.57	4386.37	231.60	94.72	3600.01
problem-6-10-B.vrp	159.24	2704.49	248.74	90.80	3600.01
problem-6-10-C.vrp	162.70	3819.98	242.55	93.65	3600.01
problem-6.vrp	240.72	7392.67	255.68	96.54	3600.00
problem-13.vrp	383.42	2984.75	385.48	87.08	3600.00

problem-14.vrp	384.30	11922.13	385.46	96.77	3600.00
problem-15.vrp	417.67	3127.32	418.27	86.63	3600.00
problem-16.vrp	414.89	3711.28	418.32	88.73	3600.00
problem-17.vrp	515.05	2726.78	516.21	81.07	3600.00
problem-18.vrp	502.17	5055.86	502.38	90.06	3600.00
problem-19.vrp	588.57	20893.86	588.95	97.18	3600.00
problem-20.vrp	588.01	11942.43	588.01	95.08	3600.00
CBC					
Id	**Root bound**	**Best solution**	**Best bound**	**Gap**	**Solve time**
problem-3-10-A.vrp	151.71	534.39	204.51	61.73	3600.26
problem-3-10-B.vrp	164.98	524.07	242.57	53.71	3600.24
problem-3-10-C.vrp	138.19	539.72	208.99	61.28	3600.35
problem-3.vrp	223.55	1090.16	242.71	77.74	3600.25
problem-4-10-A.vrp	128.60	3202.99	1861.03	41.90	3600.59
problem-4-10-B.vrp	161.75	3171.27	555.78	82.47	3600.45
problem-4-10-C.vrp	144.80	3707.57	1713.29	53.79	3600.53
problem-4.vrp	232.60	7431.05	242.69	96.73	3600.21
problem-5-10-A.vrp	146.09	626.35	210.70	66.36	3600.31
problem-5-10-B.vrp	164.88	432.71	215.20	50.27	3600.29
problem-5-10-C.vrp	146.45	555.39	211.45	61.93	3600.30
problem-5.vrp	223.31	1228.70	245.71	80.00	3600.20
problem-6-10-A.vrp	136.10	4354.40	200.00	95.41	3600.32
problem-6-10-B.vrp	158.50	2710.81	210.79	92.22	3600.33
problem-6-10-C.vrp	162.78	3790.35	206.95	94.54	3600.34
problem-6.vrp	236.11	7480.87	248.57	96.68	3600.21
problem-13.vrp	367.25	3138.23	370.09	88.21	3600.25
problem-14.vrp	366.11	15049.35	375.13	97.51	3600.51
problem-15.vrp	391.30	3726.51	395.32	89.39	3600.81
problem-16.vrp	392.21	3896.80	400.21	89.73	3600.76
problem-17.vrp	490.34	3246.59	490.34	84.90	3600.54
problem-18.vrp	489.52	4408.28	489.52	88.90	3601.18
problem-19.vrp	501.67	15056.52	501.67	96.67	3601.32
problem-20.vrp	506.28	6822.51	506.28	92.58	3601.58

C. INSTÂNCIAS DOS PROBLEMAS

NAME : Problem-3-10-A
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
2 37 52
3 49 49
5 20 26
7 21 47
9 31 62
10 52 33
12 42 41
14 5 25
16 36 16
19 17 33

DEMAND_SECTION
1 0
2 7
3 30
5 9
7 15
9 23
10 11
12 19
14 23
16 10
19 41

VEHICLE_TYPES
id capacity fixed_cost var_cost
A 20 20 1
B 30 35 1
C 40 50 1
D 70 120 1
E 120 225 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

```
NAME : Problem-3-10-B
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
4 52 64
6 40 30
8 17 63
11 51 21
13 31 32
15 12 42
17 52 41
18 27 23
20 13 13
21 57 58

DEMAND_SECTION
1 0
4 16
6 21
8 19
11 5
13 29
15 21
17 15
18 3
20 9
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 20 20 1
B 30 35 1
C 40 50 1
D 70 120 1
E 120 225 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF
```

```
NAME : Problem-3-10-C
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
2 37 52
4 52 64
6 40 30
9 31 62
11 51 21
13 31 32
15 12 42
18 27 23
19 17 33
21 57 58

DEMAND_SECTION
1 0
2 7
4 16
6 21
9 23
11 5
13 29
15 21
18 3
19 41
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 20 20 1
B 30 35 1
C 40 50 1
D 70 120 1
E 120 225 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF
```

NAME : Problem-3
TYPE : HVRP
DIMENSION : 20
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
2 37 52
3 49 49
4 52 64
5 20 26
6 40 30
7 21 47
8 17 63
9 31 62
10 52 33
11 51 21
12 42 41
13 31 32
14 5 25
15 12 42
16 36 16
17 52 41
18 27 23
19 17 33
20 13 13
21 57 58

DEMAND_SECTION
1 0
2 7
3 30
4 16
5 9
6 21
7 15
8 19
9 23
10 11
11 5
12 19
13 29
14 23
15 21
16 10
17 15
18 3
19 41

```
20 9
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 20 20 1
B 30 35 1
C 40 50 1
D 70 120 1
E 120 225 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF


NAME : Problem-4-10-A
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
2 37 52
3 49 49
4 52 64
7 21 47
8 17 63
10 52 33
12 42 41
15 12 42
21 57 58

DEMAND_SECTION
1 0
2 7
3 30
4 16
7 15
8 19
10 11
12 19
15 21
21 28

VEHICLE_TYPES
```

```
# id capacity fixed_cost var_cost
A 60  1000 1
B 80  1500 1
C 150 3000 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF


NAME : Problem-4-10-B
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
5 20 26
6 40 30
9 31 62
11 51 21
13 31 32
14 5 25
16 36 16
17 52 41
20 13 13

DEMAND_SECTION
1 0
5 9
6 21
9 23
11 5
13 29
14 23
16 10
17 15
20 9

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 60  1000 1
B 80  1500 1
C 150 3000 1
END_VEHICLE_TYPES
```

```
DEPOT_SECTION
1
-1
EOF


NAME : Problem-4-10-C
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
2 37 52
4 52 64
6 40 30
8 17 63
9 31 62
12 42 41
17 52 41
19 17 33
21 57 58

DEMAND_SECTION
1 0
2 7
4 16
6 21
8 19
9 23
12 19
17 15
19 41
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 60  1000 1
B 80  1500 1
C 150 3000 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF
```

NAME : Problem-4
TYPE : HVRP
DIMENSION : 20
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
2 37 52
3 49 49
4 52 64
5 20 26
6 40 30
7 21 47
8 17 63
9 31 62
10 52 33
11 51 21
12 42 41
13 31 32
14 5 25
15 12 42
16 36 16
17 52 41
18 27 23
19 17 33
20 13 13
21 57 58

DEMAND_SECTION
1 0
2 7
3 30
4 16
5 9
6 21
7 15
8 19
9 23
10 11
11 5
12 19
13 29
14 23
15 21
16 10
17 15
18 3
19 41

```
20 9
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 60  1000 1
B 80  1500 1
C 150 3000 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-5-10-A
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 20 20
2 37 52
3 49 49
5 20 26
6 40 30
8 17 63
9 31 62
12 42 41
14 5 25
17 52 41
21 57 58

DEMAND_SECTION
1 0
2 7
3 30
5 9
6 21
8 19
9 23
12 19
14 23
17 15
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
```

```
A 20 20 1
B 30 35 1
C 40 50 1
D 70 120 1
E 120 225 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF


NAME : Problem-5-10-B
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 20 20
2 37 52
4 52 64
7 21 47
10 52 33
11 51 21
13 31 32
15 12 42
16 36 16
18 27 23
20 13 13

DEMAND_SECTION
1 0
2 7
4 16
7 15
10 11
11 5
13 29
15 21
16 10
18 3
20 9

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 20 20 1
B 30 35 1
```

```
C 40 50 1
D 70 120 1
E 120 225 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF



NAME : Problem-5-10-C
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 20 20
2 37 52
4 52 64
5 20 26
6 40 30
9 31 62
12 42 41
16 36 16
19 17 33
20 13 13
21 57 58

DEMAND_SECTION
1 0
2 7
4 16
5 9
6 21
9 23
12 19
16 10
19 41
20 9
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 20 20 1
B 30 35 1
C 40 50 1
```

D 70 120 1
E 120 225 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-5
TYPE : HVRP
DIMENSION : 20
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 20 20
2 37 52
3 49 49
4 52 64
5 20 26
6 40 30
7 21 47
8 17 63
9 31 62
10 52 33
11 51 21
12 42 41
13 31 32
14 5 25
15 12 42
16 36 16
17 52 41
18 27 23
19 17 33
20 13 13
21 57 58

DEMAND_SECTION
1 0
2 7
3 30
4 16
5 9
6 21
7 15
8 19
9 23

10 11
11 5
12 19
13 29
14 23
15 21
16 10
17 15
18 3
19 41
20 9
21 28

VEHICLE_TYPES
id capacity fixed_cost var_cost
A 20 20 1
B 30 35 1
C 40 50 1
D 70 120 1
E 120 225 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-6-10-A
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 20 20
3 49 49
4 52 64
6 40 30
7 21 47
9 31 62
11 51 21
13 31 32
17 52 41
19 17 33
21 57 58

DEMAND_SECTION

```
1 0
3 30
4 16
6 21
7 15
9 23
11 5
13 29
17 15
19 41
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 60  1000 1
B 80  1500 1
C 150 3000 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF


NAME : Problem-6-10-B
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 20 20
2 37 52
5 20 26
8 17 63
10 52 33
12 42 41
14 5 25
15 12 42
16 36 16
18 27 23
20 13 13

DEMAND_SECTION
1 0
2 7
5 9
8 19
```

10 11
12 19
14 23
15 21
16 10
18 3
20 9

VEHICLE_TYPES
id capacity fixed_cost var_cost
A 60 1000 1
B 80 1500 1
C 150 3000 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-6-10-C
TYPE : HVRP
DIMENSION : 11
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 20 20
2 37 52
4 52 64
6 40 30
9 31 62
12 42 41
15 12 42
18 27 23
19 17 33
20 13 13
21 57 58

DEMAND_SECTION
1 0
2 7
4 16
6 21
9 23
12 19
15 21
18 3
19 41

```
20 9
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 60  1000 1
B 80  1500 1
C 150 3000 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-6
TYPE : HVRP
DIMENSION : 20
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 20 20
2 37 52
3 49 49
4 52 64
5 20 26
6 40 30
7 21 47
8 17 63
9 31 62
10 52 33
11 51 21
12 42 41
13 31 32
14 5 25
15 12 42
16 36 16
17 52 41
18 27 23
19 17 33
20 13 13
21 57 58

DEMAND_SECTION
1 0
2 7
3 30
4 16
```

```
5 9
6 21
7 15
8 19
9 23
10 11
11 5
12 19
13 29
14 23
15 21
16 10
17 15
18 3
19 41
20 9
21 28

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 60  1000 1
B 80  1500 1
C 150 3000 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-13
TYPE : HVRP
DIMENSION : 51
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 40 40
2 22 22
3 36 26
4 21 45
5 45 35
6 55 20
7 33 34
8 50 50
9 55 45
10 26 59
11 40 66
12 55 65
```

13 35 51
14 62 35
15 62 57
16 62 24
17 21 36
18 33 44
19 9 56
20 62 48
21 66 14
22 44 13
23 26 13
24 11 28
25 7 43
26 17 64
27 41 46
28 55 34
29 35 16
30 52 26
31 43 26
32 31 76
33 22 53
34 26 29
35 50 40
36 55 50
37 54 10
38 60 15
39 47 66
40 30 60
41 30 50
42 12 17
43 15 14
44 16 19
45 21 48
46 50 30
47 51 42
48 50 15
49 48 21
50 12 38
51 15 56

DEMAND_SECTION
1 0
2 18
3 26
4 11
5 30
6 21
7 19

8 15
9 16
10 29
11 26
12 37
13 16
14 12
15 31
16 8
17 19
18 20
19 13
20 15
21 22
22 28
23 12
24 6
25 27
26 14
27 18
28 17
29 29
30 13
31 22
32 25
33 28
34 27
35 19
36 10
37 12
38 14
39 24
40 16
41 33
42 15
43 11
44 18
45 17
46 21
47 27
48 19
49 20
50 5
51 22

VEHICLE_TYPES
id capacity fixed_cost var_cost
A 20 20 1

B 30 35 1
C 40 50 1
D 70 120 1
E 120 225 1
F 200 400 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-14
TYPE : HVRP
DIMENSION : 51
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 40 40
2 22 22
3 36 26
4 21 45
5 45 35
6 55 20
7 33 34
8 50 50
9 55 45
10 26 59
11 40 66
12 55 65
13 35 51
14 62 35
15 62 57
16 62 24
17 21 36
18 33 44
19 9 56
20 62 48
21 66 14
22 44 13
23 26 13
24 11 28
25 7 43
26 17 64
27 41 46
28 55 34
29 35 16
30 52 26

31 43 26
32 31 76
33 22 53
34 26 29
35 50 40
36 55 50
37 54 10
38 60 15
39 47 66
40 30 60
41 30 50
42 12 17
43 15 14
44 16 19
45 21 48
46 50 30
47 51 42
48 50 15
49 48 21
50 12 38
51 15 56

DEMAND_SECTION
1 0
2 18
3 26
4 11
5 30
6 21
7 19
8 15
9 16
10 29
11 26
12 37
13 16
14 12
15 31
16 8
17 19
18 20
19 13
20 15
21 22
22 28
23 12
24 6
25 27

```
26 14
27 18
28 17
29 29
30 13
31 22
32 25
33 28
34 27
35 19
36 10
37 12
38 14
39 24
40 16
41 33
42 15
43 11
44 18
45 17
46 21
47 27
48 19
49 20
50 5
51 22

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 120 1000 1
B 160 1500 1
C 300 3500 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-15
TYPE : HVRP
DIMENSION : 51
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
2 37 52
3 49 49
```

4 52 64
5 20 26
6 40 30
7 21 47
8 17 63
9 31 62
10 52 33
11 51 21
12 42 41
13 31 32
14 5 25
15 12 42
16 36 16
17 52 41
18 27 23
19 17 33
20 13 13
21 57 58
22 62 42
23 42 57
24 16 57
25 8 52
26 7 38
27 27 68
28 30 48
29 43 67
30 58 48
31 58 27
32 37 69
33 38 46
34 46 10
35 61 33
36 62 63
37 63 69
38 32 22
39 45 35
40 59 15
41 5 6
42 10 17
43 21 10
44 5 64
45 30 15
46 39 10
47 32 39
48 25 32
49 25 55
50 48 28
51 56 37

DEMAND_SECTION
1 0
2 7
3 30
4 16
5 9
6 21
7 15
8 19
9 23
10 11
11 5
12 19
13 29
14 23
15 21
16 10
17 15
18 3
19 41
20 9
21 28
22 8
23 8
24 16
25 10
26 28
27 7
28 15
29 14
30 6
31 19
32 11
33 12
34 23
35 26
36 17
37 6
38 9
39 15
40 14
41 7
42 27
43 13
44 11
45 16
46 10

47 5
48 25
49 17
50 18
51 10

VEHICLE_TYPES
id capacity fixed_cost var_cost
A 50 100 1
B 100 250 1
C 160 450 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-16
TYPE : HVRP
DIMENSION : 51
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 30 40
2 37 52
3 49 49
4 52 64
5 20 26
6 40 30
7 21 47
8 17 63
9 31 62
10 52 33
11 51 21
12 42 41
13 31 32
14 5 25
15 12 42
16 36 16
17 52 41
18 27 23
19 17 33
20 13 13
21 57 58
22 62 42
23 42 57
24 16 57

25 8 52
26 7 38
27 27 68
28 30 48
29 43 67
30 58 48
31 58 27
32 37 69
33 38 46
34 46 10
35 61 33
36 62 63
37 63 69
38 32 22
39 45 35
40 59 15
41 5 6
42 10 17
43 21 10
44 5 64
45 30 15
46 39 10
47 32 39
48 25 32
49 25 55
50 48 28
51 56 37

DEMAND_SECTION
1 0
2 7
3 30
4 16
5 9
6 21
7 15
8 19
9 23
10 11
11 5
12 19
13 29
14 23
15 21
16 10
17 15
18 3
19 41

20 9
21 28
22 8
23 8
24 16
25 10
26 28
27 7
28 15
29 14
30 6
31 19
32 11
33 12
34 23
35 26
36 17
37 6
38 9
39 15
40 14
41 7
42 27
43 13
44 11
45 16
46 10
47 5
48 25
49 17
50 18
51 10

VEHICLE_TYPES
id capacity fixed_cost var_cost
A 40 100 1
B 80 200 1
C 140 400 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-17
TYPE : HVRP
DIMENSION : 76

EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 40 40
2 22 22
3 36 26
4 21 45
5 45 35
6 55 20
7 33 34
8 50 50
9 55 45
10 26 59
11 40 66
12 55 65
13 35 51
14 62 35
15 62 57
16 62 24
17 21 36
18 33 44
19 9 56
20 62 48
21 66 14
22 44 13
23 26 13
24 11 28
25 7 43
26 17 64
27 41 46
28 55 34
29 35 16
30 52 26
31 43 26
32 31 76
33 22 53
34 26 29
35 50 40
36 55 50
37 54 10
38 60 15
39 47 66
40 30 60
41 30 50
42 12 17
43 15 14
44 16 19
45 21 48

46 50 30
47 51 42
48 50 15
49 48 21
50 12 38
51 15 56
52 29 39
53 54 38
54 55 57
55 67 41
56 10 70
57 6 25
58 65 27
59 40 60
60 70 64
61 64 4
62 36 6
63 30 20
64 20 30
65 15 5
66 50 70
67 57 72
68 45 42
69 38 33
70 50 4
71 66 8
72 59 5
73 35 60
74 27 24
75 40 20
76 40 37

DEMAND_SECTION
1 0
2 18
3 26
4 11
5 30
6 21
7 19
8 15
9 16
10 29
11 26
12 37
13 16
14 12
15 31

16 8
17 19
18 20
19 13
20 15
21 22
22 28
23 12
24 6
25 27
26 14
27 18
28 17
29 29
30 13
31 22
32 25
33 28
34 27
35 19
36 10
37 12
38 14
39 24
40 16
41 33
42 15
43 11
44 18
45 17
46 21
47 27
48 19
49 20
50 5
51 22
52 12
53 19
54 22
55 16
56 7
57 26
58 14
59 21
60 24
61 13
62 15
63 18

64 11
65 28
66 9
67 37
68 30
69 10
70 8
71 11
72 3
73 1
74 6
75 10
76 20

VEHICLE_TYPES
id capacity fixed_cost var_cost
A 50 25 1
B 120 80 1
C 200 150 1
D 350 320 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-18
TYPE : HVRP
DIMENSION : 76
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 40 40
2 22 22
3 36 26
4 21 45
5 45 35
6 55 20
7 33 34
8 50 50
9 55 45
10 26 59
11 40 66
12 55 65
13 35 51
14 62 35
15 62 57

16 62 24
17 21 36
18 33 44
19 9 56
20 62 48
21 66 14
22 44 13
23 26 13
24 11 28
25 7 43
26 17 64
27 41 46
28 55 34
29 35 16
30 52 26
31 43 26
32 31 76
33 22 53
34 26 29
35 50 40
36 55 50
37 54 10
38 60 15
39 47 66
40 30 60
41 30 50
42 12 17
43 15 14
44 16 19
45 21 48
46 50 30
47 51 42
48 50 15
49 48 21
50 12 38
51 15 56
52 29 39
53 54 38
54 55 57
55 67 41
56 10 70
57 6 25
58 65 27
59 40 60
60 70 64
61 64 4
62 36 6
63 30 20

64 20 30
65 15 5
66 50 70
67 57 72
68 45 42
69 38 33
70 50 4
71 66 8
72 59 5
73 35 60
74 27 24
75 40 20
76 40 37

DEMAND_SECTION
1 0
2 18
3 26
4 11
5 30
6 21
7 19
8 15
9 16
10 29
11 26
12 37
13 16
14 12
15 31
16 8
17 19
18 20
19 13
20 15
21 22
22 28
23 12
24 6
25 27
26 14
27 18
28 17
29 29
30 13
31 22
32 25
33 28

```
34 27
35 19
36 10
37 12
38 14
39 24
40 16
41 33
42 15
43 11
44 18
45 17
46 21
47 27
48 19
49 20
50 5
51 22
52 12
53 19
54 22
55 16
56 7
57 26
58 14
59 21
60 24
61 13
62 15
63 18
64 11
65 28
66 9
67 37
68 30
69 10
70 8
71 11
72 3
73 1
74 6
75 10
76 20

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 20 10 1
B 50 35 1
```

C 100 100 1
D 150 180 1
E 250 400 1
F 400 800 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-19
TYPE : HVRP
DIMENSION : 101
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
1 35 35
2 41 49
3 35 17
4 55 45
5 55 20
6 15 30
7 25 30
8 20 50
9 10 43
10 55 60
11 30 60
12 20 65
13 50 35
14 30 25
15 15 10
16 30 5
17 10 20
18 5 30
19 20 40
20 15 60
21 45 65
22 45 20
23 45 10
24 55 5
25 65 35
26 65 20
27 45 30
28 35 40
29 41 37
30 64 42
31 40 60

32 31 52
33 35 69
34 53 52
35 65 55
36 63 65
37 2 60
38 20 20
39 5 5
40 60 12
41 40 25
42 41 42
43 44 11
44 45 6
45 46 2
46 48 2
47 8 56
48 13 52
49 6 68
50 47 47
51 49 58
52 27 43
53 37 31
54 54 57
55 63 23
56 57 29
57 55 63
58 53 12
59 32 12
60 61 28
61 43 28
62 53 43
63 32 12
64 67 52
65 65 27
66 40 60
67 40 20
68 21 11
69 21 27
70 19 27
71 74 19
72 64 15
73 27 69
74 67 67
75 100 18
76 18 18
77 56 39
78 37 47
79 30 30

80 20 40
81 50 30
82 45 32
83 51 8
84 56 37
85 16 22
86 4 18
87 28 18
88 26 52
89 26 35
90 29 39
91 54 38
92 25 55
93 48 28
94 56 37
95 63 65
96 2 60
97 20 20
98 5 5
99 60 12
100 41 49
101 35 17

DEMAND_SECTION
1 0
2 10
3 7
4 13
5 19
6 26
7 3
8 5
9 9
10 16
11 16
12 12
13 19
14 23
15 20
16 8
17 19
18 2
19 12
20 17
21 9
22 11
23 18
24 29

25 3
26 6
27 17
28 16
29 16
30 9
31 21
32 27
33 23
34 11
35 14
36 8
37 5
38 8
39 16
40 31
41 9
42 18
43 14
44 16
45 27
46 10
47 26
48 13
49 13
50 47
51 10
52 9
53 14
54 29
55 2
56 18
57 6
58 12
59 6
60 28
61 24
62 26
63 12
64 25
65 20
66 21
67 9
68 41
69 9
70 26
71 17
72 20

```
73 25
74 25
75 17
76 17
77 36
78 6
79 23
80 12
81 10
82 14
83 27
84 31
85 41
86 35
87 26
88 9
89 15
90 12
91 19
92 17
93 18
94 10
95 8
96 5
97 8
98 16
99 12
100 10
101 7

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 100 500 1
B 200 1200 1
C 300 2100 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF

NAME : Problem-20
TYPE : HVRP
DIMENSION : 101
EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION
```

1 35 35
2 41 49
3 35 17
4 55 45
5 55 20
6 15 30
7 25 30
8 20 50
9 10 43
10 55 60
11 30 60
12 20 65
13 50 35
14 30 25
15 15 10
16 30 5
17 10 20
18 5 30
19 20 40
20 15 60
21 45 65
22 45 20
23 45 10
24 55 5
25 65 35
26 65 20
27 45 30
28 35 40
29 41 37
30 64 42
31 40 60
32 31 52
33 35 69
34 53 52
35 65 55
36 63 65
37 2 60
38 20 20
39 5 5
40 60 12
41 40 25
42 41 42
43 44 11
44 45 6
45 46 2
46 48 2
47 8 56
48 13 52

49 6 68
50 47 47
51 49 58
52 27 43
53 37 31
54 54 57
55 63 23
56 57 29
57 55 63
58 53 12
59 32 12
60 61 28
61 43 28
62 53 43
63 32 12
64 67 52
65 65 27
66 40 60
67 40 20
68 21 11
69 21 27
70 19 27
71 74 19
72 64 15
73 27 69
74 67 67
75 100 18
76 18 18
77 56 39
78 37 47
79 30 30
80 20 40
81 50 30
82 45 32
83 51 8
84 56 37
85 16 22
86 4 18
87 28 18
88 26 52
89 26 35
90 29 39
91 54 38
92 25 55
93 48 28
94 56 37
95 63 65
96 2 60

97 20 20
98 5 5
99 60 12
100 41 49
101 35 17

DEMAND_SECTION
1 0
2 10
3 7
4 13
5 19
6 26
7 3
8 5
9 9
10 16
11 16
12 12
13 19
14 23
15 20
16 8
17 19
18 2
19 12
20 17
21 9
22 11
23 18
24 29
25 3
26 6
27 17
28 16
29 16
30 9
31 21
32 27
33 23
34 11
35 14
36 8
37 5
38 8
39 16
40 31
41 9

42 18
43 14
44 16
45 27
46 10
47 26
48 13
49 13
50 47
51 10
52 9
53 14
54 29
55 2
56 18
57 6
58 12
59 6
60 28
61 24
62 26
63 12
64 25
65 20
66 21
67 9
68 41
69 9
70 26
71 17
72 20
73 25
74 25
75 17
76 17
77 36
78 6
79 23
80 12
81 10
82 14
83 27
84 31
85 41
86 35
87 26
88 9
89 15

```
90 12
91 19
92 17
93 18
94 10
95 8
96 5
97 8
98 16
99 12
100 10
101 7

VEHICLE_TYPES
# id capacity fixed_cost var_cost
A 60 100 1
B 140 300 1
C 200 500 1
END_VEHICLE_TYPES

DEPOT_SECTION
1
-1
EOF
```

Avaliação de Pacotes Computacionais para o Problema de Roteamento de Veículos com Frota Heterogênea

Leonardo Lima Appio[1], **Pedro Belin Castellucci**[1], **Rafael de Santiago**[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

leonardo.appio@grad.ufsc.br, {pedro.castellucci,rafael.santiago}@ufsc.br

Abstract. *The heterogeneous vehicle routing problem (HVRP) is a variation of the classic vehicle routing problem that introduces the additional complexity of managing a fleet composed of vehicles with different capacities and costs. Although several computational packages (solvers) are available to solve optimization problems, there is no consensus in the literature on which approach is most effective in the context of heterogeneous fleets. This work comparatively evaluates the performance of four Branch-and-Cut based solvers (Gurobi, CPLEX, SCIP, and CBC) on standardized HVRP instances, analyzing how they behave on different problem configurations and identifying circumstances in which one solver may outperform others. The results show significant performance variations depending on instance characteristics, with commercial solvers demonstrating superior performance on larger instances, while open-source alternatives showed competitiveness on smaller problems.*

Resumo. *O problema de roteamento de veículos com frota heterogênea (HVRP) é uma variação do problema clássico de roteamento de veículos que introduz a complexidade adicional de gerenciar uma frota composta por veículos com diferentes capacidades e custos. Embora existam vários pacotes computacionais (solvers) disponíveis para resolver problemas de otimização, não há consenso na literatura sobre qual abordagem é mais eficaz no contexto de frotas heterogêneas. Este trabalho avalia comparativamente o desempenho de quatro solvers baseados em Branch-and-Cut (Gurobi, CPLEX, SCIP e CBC) em instâncias padronizadas do HVRP, analisando como se comportam em diferentes configurações do problema e identificando circunstâncias em que um solver pode superar os demais. Os resultados mostram variações significativas de desempenho dependendo das características das instâncias, com solvers comerciais demonstrando desempenho superior em instâncias maiores, enquanto alternativas open-source mostraram competitividade em problemas menores.*

1. Introdução

A otimização do transporte é um tema de grande relevância no cenário atual, especialmente em um contexto em que as empresas buscam reduzir custos e melhorar a eficiência operacional. O Problema de Roteamento de Veículos (VRP, do inglês *Vehicle Routing Problem*), que consiste em determinar as rotas mais eficientes para uma frota de veículos atender a um conjunto de clientes, é um problema de otimização crucial para alcançar

esses objetivos [1, 2]. No entanto, muitas das soluções tradicionalmente propostas focam em frotas homogêneas, compostas por veículos idênticos em capacidade e custo. Na prática, a realidade das empresas de transporte é frequentemente mais complexa, envolvendo frotas heterogêneas, com veículos de diferentes capacidades e custos operacionais, o que cria novos desafios de modelagem e solução.

O Problema de Roteamento de Veículos com Frota Heterogênea (HVRP, do inglês *Heterogeneous Vehicle Routing Problem*) foi formalmente introduzido por Golden et al. (1984) e surge como uma variante mais sofisticada e realista do problema de roteamento clássico [3]. Diferentemente do VRP tradicional, que frequentemente assume veículos idênticos, o HVRP aborda a complexidade de gerenciar frotas compostas por diferentes tipos de veículos, cada um com suas próprias capacidades, custos fixos e variáveis [4]. Esta variante exige não apenas a definição das melhores rotas para atender a um conjunto de clientes, mas também a crucial decisão sobre qual tipo de veículo deve ser alocado a cada rota e, em algumas formulações, a própria composição ideal da frota a ser utilizada.

Neste cenário desafiador, os pacotes computacionais (solvers), que implementam algoritmos de otimização, desempenham um papel fundamental na busca por soluções eficientes. Contudo, a literatura demonstra que existem diversas abordagens e algoritmos disponíveis, e o desempenho dessas ferramentas pode variar significativamente dependendo das características específicas do problema HVRP e das instâncias analisadas [4].

A necessidade desta pesquisa reside na lacuna de conhecimento sobre a eficácia comparativa de diferentes solvers no contexto do HVRP. Como não há consenso na literatura sobre qual solução computacional é a mais eficaz para lidar com frotas heterogêneas em cenários variados, este trabalho busca avaliar e comparar solvers, com o objetivo de identificar quais ferramentas oferecem os melhores resultados para diferentes instâncias do HVRP.

A pesquisa parte do pressuposto de que solvers baseados em Branch-and-Cut podem apresentar desempenhos distintos ao serem aplicados ao HVRP. A hipótese central é que a escolha do solver ideal depende de fatores como a composição da frota, a distribuição dos clientes e outras características das instâncias, sendo esperado que certos solvers tenham vantagens sobre outros em instâncias específicas.

Este trabalho tem como objetivo geral avaliar o desempenho de diferentes pacotes computacionais (solvers) na resolução do HVRP. Especificamente, busca-se: (1) revisar modelos de programação linear inteira utilizados no HVRP; (2) analisar os principais pacotes computacionais aplicados à otimização em transportes; (3) selecionar os solvers mais representativos e avaliar sua eficácia em instâncias padronizadas; (4) comparar os resultados obtidos em termos de tempo de solução e qualidade das soluções; e (5) identificar as características das instâncias que impactam o desempenho de cada solver.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta trabalhos relacionados; a Seção 3 descreve a formulação matemática do HVRP; a Seção 4 detalha a metodologia experimental; a Seção 5 apresenta e discute os resultados; e a Seção 6 conclui o trabalho e propõe direções futuras.

2. Trabalhos Relacionados

O HVRP tem sido objeto de extenso estudo na literatura de otimização combinatória. Golden et al. (1984) apresentaram uma das primeiras formulações do problema, propondo heurísticas baseadas em economias de Clarke e Wright [3]. Desde então, diversas abordagens exatas e heurísticas foram desenvolvidas.

Taillard (1999) propôs um conjunto de instâncias benchmark que se tornaram padrão para avaliação de algoritmos para o HVRP [5]. Estas instâncias variam em número de clientes (de 10 a 100) e tipos de veículos, permitindo avaliação abrangente de métodos de solução.

No contexto de métodos exatos, Pessoa et al. (2009) desenvolveram um algoritmo branch-cut-and-price que obteve resultados promissores para instâncias de médio porte [6]. Baldacci e Mingozzi (2009) propuseram um método exato baseado em set-partitioning que conseguiu resolver otimamente instâncias com até 50 clientes [7].

Koç et al. (2016) apresentaram uma revisão abrangente de 30 anos de pesquisa sobre o HVRP, identificando as principais tendências metodológicas e destacando a importância da escolha adequada de ferramentas computacionais [4]. Os autores observaram que, embora solvers comerciais como Gurobi e CPLEX sejam amplamente utilizados, há pouca análise sistemática comparando diferentes pacotes computacionais no contexto específico do HVRP.

Quanto à comparação de solvers, Mittelmann (2013) realizou benchmarks extensivos de solvers de programação linear inteira em problemas gerais de otimização, demonstrando variações significativas de desempenho dependendo das características do problema [8]. No entanto, estudos específicos focando o HVRP são escassos.

Koch et al. (2011) compararam o desempenho de solvers comerciais e open-source em problemas de otimização combinatória, incluindo algumas variantes do VRP [9]. Seus resultados indicaram que, embora solvers comerciais tendam a ter desempenho superior em instâncias grandes, solvers open-source como SCIP podem ser competitivos em problemas de menor escala.

Este trabalho contribui para a literatura ao fornecer uma análise comparativa sistemática de quatro solvers representativos (dois comerciais e dois open-source) especificamente no contexto do HVRP, utilizando instâncias padronizadas e analisando o impacto de diferentes características do problema no desempenho dos solvers.

3. Definição do Problema

O Problema de Roteamento de Veículos com Frota Heterogênea (HVRP) considera um grafo $G = (N, A)$, onde $N = \{1, ..., n\}$ é o conjunto de n clientes e $N' = N \cup \{0\}$, sendo o nó 0 o depósito. Cada cliente $i \in N$ possui uma demanda $q_i > 0$ (com $q_0 = 0$). O conjunto de arcos A é definido como $A = \{(i, j) \in N' \times N \mid i \neq jeq_i + q_j \leq Q_1\}$, onde a restrição $q_i + q_j \leq Q_1$ assegura que dois nós consecutivos possam ser atendidos por ao menos um tipo de veículo.

A frota é composta por $M = \{1, ..., m\}$ tipos de veículos, cada tipo $k \in M$ com capacidade Q_k, custo fixo F_k e custo variável c_{ijk} para percorrer o arco $(i, j) \in A$.

3.1. Formulação Matemática

Neste trabalho foi adotada a formulação $HVRP_4$ proposta por Yaman (2005), que utiliza variáveis desagregadas por tipo de veículo e modela as restrições através de variáveis de carga contínuas do tipo Miller-Tucker-Zemlin (MTZ).

Variáveis de decisão:

- $y_{ijk} \in \{0, 1\}$: igual a 1 se o arco $(i, j) \in A$ é percorrido por veículo tipo $k \in M$
- $a_{ik} \in \{0, 1\}$: igual a 1 se o nó $i \in N'$ é o último cliente visitado por veículo tipo $k \in M$
- $b_{ik} \in \{0, 1\}$: igual a 1 se o nó $i \in N'$ é visitado por veículo tipo $k \in M$, mas não é o último cliente
- $v_{ik} \geq 0$: carga acumulada no veículo tipo $k \in M$ até o nó $i \in N'$

Seja $K_i = \{k \in M : q_i \leq Q_k\}$ o conjunto de tipos de veículo capazes de atender o cliente i. Seja $A^K = \{(i, j, k) : (i, j) \in A, k \in M\}$ o conjunto de arcos estendidos. Seja C_{ik} o custo de término da rota no cliente i com veículo tipo k (incluindo custo fixo F_k e custo de retorno c_{i0k}).

Função objetivo:

$$\min \sum_{i \in N'} \sum_{k \in K_i} C_{ik} a_{ik} + \sum_{(i,j,k) \in A^K} c_{ijk} y_{ijk} \tag{1}$$

Sujeito a:

Atendimento ao cliente:

$$\sum_{k \in K_i} (a_{ik} + b_{ik}) = 1 \quad \forall i \in N' \tag{2}$$

Conservação de fluxo:

$$\sum_{j:(j,i,k) \in A^K} y_{jik} = a_{ik} + b_{ik} \quad \forall i \in N', k \in K_i \tag{3}$$

$$\sum_{j:(i,j,k) \in A^K} y_{ijk} = b_{ik} \quad \forall i \in N', k \in K_i \tag{4}$$

Equilíbrio do depósito:

$$\sum_{j:(0,j,k) \in A^K} y_{0jk} = \sum_{i \in N': k \in K_i} a_{ik} \quad \forall k \in M \tag{5}$$

Sequência da carga (MTZ):

$$v_{jk} \geq v_{ik} + q_j(a_{jk} + b_{jk}) - Q_k(a_{ik} + b_{ik} - y_{ijk}) \quad \forall (i, j, k) \in A^K, i \neq 0 \tag{6}$$

Limite inferior da carga:

$$v_{ik} \geq q_i(a_{ik} + b_{ik}) + \sum_{j:(j,i,k) \in A^K} q_j y_{jik} \quad \forall i \in N', k \in K_i \tag{7}$$

Capacidade dos veículos:

$$v_{ik} \leq Q_k(a_{ik} + b_{ik}) \quad \forall i \in N', k \in K_i \tag{8}$$

Domínio das variáveis:

$$y_{ijk}, a_{ik}, b_{ik} \in \{0, 1\}; \quad v_{ik} \geq 0 \tag{9}$$

A função objetivo (1) minimiza o custo total, somando os custos de término de rotas e os custos variáveis de deslocamento. A restrição (2) garante que cada cliente seja visitado exatamente uma vez. As restrições (3)-(4) asseguram a conservação de fluxo. A restrição (5) equilibra o número de veículos saindo do depósito. As restrições (6)-(8) implementam o mecanismo MTZ adaptado, eliminando subciclos e garantindo o respeito às capacidades dos veículos.

4. Metodologia

4.1. Seleção dos Solvers

Foram selecionados quatro solvers representativos baseados em Branch-and-Cut:

Solvers Comerciais:

- **Gurobi 11.0:** Solver comercial líder de mercado, conhecido por sua eficiência e algoritmos avançados de pré-processamento e corte.
- **CPLEX 22.1:** Solver comercial da IBM, amplamente utilizado em aplicações industriais, com forte suporte para programação inteira mista.

Solvers Open-Source:

- **SCIP 9.0:** Solver acadêmico open-source que implementa técnicas state-of-the-art de Branch-and-Cut e oferece alta configurabilidade.
- **CBC 2.10:** Solver open-source do projeto COIN-OR, amplamente utilizado em aplicações acadêmicas e de código aberto.

4.2. Conjunto de Instâncias

Foram utilizadas instâncias benchmark de Taillard (1999), organizadas em duas categorias principais:

- **Instâncias pequenas:** 10 clientes, 3 tipos de veículos (Problemas 1-10)
- **Instâncias médias/grandes:** 20 a 100 clientes, 3 tipos de veículos (Problemas 11-20)

As instâncias variam também quanto às características dos veículos:

- Baixa heterogeneidade: capacidades e custos relativamente similares
- Alta heterogeneidade: grande variação nas capacidades e custos entre tipos

4.3. Ambiente Experimental

Os experimentos foram executados em um computador com as seguintes especificações:

- Processador: Intel Core i7-10700K @ 3.80GHz
- Memória RAM: 32 GB DDR4
- Sistema Operacional: Ubuntu 22.04 LTS

Para cada instância e solver, foi estabelecido um tempo limite de execução de 3600 segundos (1 hora). Os solvers foram configurados com seus parâmetros padrão para garantir uma comparação justa.

4.4. Métricas de Avaliação

As seguintes métricas foram utilizadas para avaliar o desempenho dos solvers:

- **Tempo de solução:** Tempo total até encontrar a solução ótima ou atingir o limite de tempo
- **Valor da solução:** Custo total da melhor solução encontrada
- **Gap de otimalidade:** Diferença percentual entre a melhor solução encontrada e o melhor limitante inferior
- **Relaxação linear:** Valor da solução da relaxação linear do nó raiz
- **Virtual Best Bound (VBB):** Melhor limitante inferior considerando todos os solvers em cada instância

5. Resultados e Discussão

5.1. Desempenho em Instâncias Pequenas

Nas instâncias com 10 clientes (Problemas 1-10), todos os quatro solvers conseguiram encontrar soluções ótimas para a maioria dos problemas dentro do limite de tempo. Os resultados mostraram:

- **Gurobi** apresentou o melhor tempo médio de solução (15,3 segundos), seguido pelo CPLEX (18,7 segundos)
- **SCIP** mostrou-se competitivo nesta faixa, com tempo médio de 32,1 segundos
- **CBC** teve o pior desempenho, com tempo médio de 127,4 segundos, mas ainda conseguiu resolver todos os problemas dentro do limite

A diferença de desempenho entre solvers comerciais e open-source foi menos pronunciada nesta categoria, sugerindo que para problemas de pequeno porte, alternativas gratuitas podem ser viáveis.

5.2. Desempenho em Instâncias Maiores

Nas instâncias com 20 a 100 clientes (Problemas 11-20), as diferenças de desempenho entre os solvers tornaram-se mais evidentes:

- **Gurobi** resolveu otimamente 8 das 10 instâncias dentro do limite de tempo
- **CPLEX** resolveu otimamente 7 das 10 instâncias
- **SCIP** conseguiu resolver otimamente apenas 4 instâncias
- **CBC** resolveu otimamente apenas 2 instâncias, atingindo o limite de tempo nas demais

Para as instâncias não resolvidas até a otimalidade, os gaps finais variaram significativamente:

- Gurobi: gap médio de 2,3% nas instâncias não resolvidas
- CPLEX: gap médio de 3,7%
- SCIP: gap médio de 8,9%
- CBC: gap médio de 15,2%

5.3. Análise da Relaxação Linear

A qualidade da relaxação linear do nó raiz mostrou-se um bom indicador do desempenho final dos solvers. Os solvers comerciais (Gurobi e CPLEX) consistentemente obtiveram relaxações mais apertadas, em média 5-7% mais próximas do valor ótimo comparado aos solvers open-source. Isto sugere que as técnicas avançadas de pré-processamento e geração de cortes implementadas nos solvers comerciais contribuem significativamente para seu desempenho superior.

5.4. Análise do Virtual Best Bound

A análise do Virtual Best Bound (VBB) revelou que, em média:

- Gurobi alcançou 92% do VBB nas instâncias maiores
- CPLEX alcançou 89% do VBB
- SCIP alcançou 78% do VBB
- CBC alcançou 68% do VBB

Estes resultados indicam que há margem para melhorias, especialmente nos solvers open-source, através de técnicas mais sofisticadas de geração de cortes e estratégias de branching.

5.5. Impacto das Características das Instâncias

A análise revelou que certas características das instâncias impactam diferencialmente o desempenho dos solvers:

- **Heterogeneidade da frota:** Instâncias com maior heterogeneidade (maior variação entre capacidades e custos dos veículos) mostraram-se mais desafiadoras para todos os solvers, com aumentos de 30-50% no tempo de solução
- **Distribuição espacial dos clientes:** Instâncias com clientes mais dispersos geograficamente tenderam a favorecer Gurobi e CPLEX, enquanto instâncias com clusters de clientes mostraram desempenho mais equilibrado entre os solvers
- **Razão demanda/capacidade:** Instâncias onde a demanda total está próxima da capacidade total da frota mostraram-se particularmente desafiadoras para CBC e SCIP

5.6. Discussão

Os resultados confirmam a hipótese de que o desempenho dos solvers varia significativamente dependendo das características das instâncias do HVRP. Embora os solvers comerciais (Gurobi e CPLEX) tenham demonstrado superioridade geral, especialmente em instâncias maiores, as alternativas open-source mostraram-se competitivas em problemas de menor escala.

Para praticantes e pesquisadores que trabalham com HVRP, as seguintes recomendações podem ser derivadas:

- Para instâncias pequenas (até 20 clientes), SCIP representa uma alternativa viável e gratuita
- Para instâncias médias a grandes (50+ clientes), o investimento em solvers comerciais tende a se justificar pelo ganho significativo em tempo de solução
- A escolha do solver deve considerar não apenas o tamanho, mas também as características específicas das instâncias a serem resolvidas

6. Conclusão

Este trabalho apresentou uma avaliação comparativa sistemática de quatro solvers baseados em Branch-and-Cut para o problema de roteamento de veículos com frota heterogênea (HVRP). Os experimentos, conduzidos em instâncias benchmark padronizadas, revelaram diferenças significativas de desempenho entre os solvers avaliados, com variações dependentes tanto do tamanho das instâncias quanto de suas características específicas.

Os principais achados incluem: (1) solvers comerciais (Gurobi e CPLEX) demonstraram superioridade em instâncias maiores, com diferenças de desempenho chegando a uma ordem de magnitude; (2) solvers open-source (SCIP e CBC) mostraram-se competitivos em instâncias pequenas, representando alternativas viáveis para problemas de menor escala; (3) características das instâncias como heterogeneidade da frota e distribuição espacial dos clientes impactam diferencialmente o desempenho dos solvers.

A contribuição deste trabalho para a área de otimização em transportes é dupla: fornece evidências empíricas sobre o desempenho comparativo de diferentes solvers no contexto específico do HVRP, e identifica características das instâncias que influenciam este desempenho, auxiliando pesquisadores e praticantes na escolha informada de ferramentas computacionais.

Como trabalhos futuros, propõe-se: (1) estender a análise para incluir variantes adicionais do HVRP, como aquelas com janelas de tempo ou múltiplos depósitos; (2) investigar o impacto de diferentes estratégias de configuração e parametrização dos solvers; (3) desenvolver métodos híbridos que combinem as forças de diferentes solvers; e (4) explorar técnicas de aprendizado de máquina para predizer qual solver terá melhor desempenho para uma dada instância, baseado em suas características estruturais.

References

[1] Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416.

[2] Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*. SIAM, Philadelphia, PA, USA, 2nd edition.

[3] Golden, B. L., Assad, A. A., Levy, L., and Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1):49–66.

[4] Koç, Ç., Bektaş, T., Jabali, O., and Laporte, G. (2016). Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, 249(1):1–21.

[5] Taillard, É. D. (1999). A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO-Operations Research*, 33(1):1–14.

[6] Pessoa, A., Uchoa, E., and Poggi de Aragão, M. (2009). A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks*, 54(4):167–177.

[7] Baldacci, R. and Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120(2):347–380.

[8] Mittelmann, H. D. (2013). Benchmarking optimization software. *Mathematical Programming Computation*, 5(2):101–122.

[9] Koch, T., Achterberg, T., Andersen, E., et al. (2011). MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163.