André Manhães Machado

Matheurística com Abordagem Hierárquica Aplicada ao Problema de Roteamento de Veículos Capacitados e ao Problema de Roteamento de Helicópteros

Vitória, ES

2021

André Manhães Machado

Matheurística com Abordagem Hierárquica Aplicada ao Problema de Roteamento de Veículos Capacitados e ao Problema de Roteamento de Helicópteros

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Informática (PPGI) da UFES como parte dos requisitos para obtenção do Grau de Doutor em Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. Geraldo Regis Mauri

Coorientador: Profª. Drª. Maria Claudia Silva Boeres

Vitória, ES

2021

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Machado, André Manhães, 1985-
M149m Matheurística com abordagem hierárquica aplicada ao problema de roteamento de veículos capacitados e ao problema de roteamento de helicópteros / André Manhães Machado. - 2021.
164 f. : il.

Orientador: Geraldo Regis Mauri.
Coorientadora: Maria Claudia Silva Boeres.
Tese (Doutorado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Otimização combinatória. 2. Programação linear. 3. Modelos matemáticos. 4. Heurística. 5. Transporte - Planejamento. 6. Helicópteros. I. Mauri, Geraldo Regis. II. Boeres, Maria Claudia Silva. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Matheurística com Abordagem Hierárquica Aplicada ao Problema de Roteamento de Veículos Capacitados e ao Problema de Roteamento de Helicópteros

André Manhães Machado

Tese submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 27 de outubro de 2021:

Prof. Dr. Geraldo Regis Mauri
Orientador, participação remota

Profª. Drª. Maria Claudia Silva Boeres
Coorientadora, participação remota

Prof. Dr. André Renato Sales Amaral
Membro Interno, participação remota

Prof. Dr. Isaac Pinheiro dos Santos
Membro Interno, participação remota

Prof. Dr. Glaydston Mattos Ribeiro
Membro Externo, participação remota

Prof. Dr. Luciano Lessa Lorenzoni
Membro Externo, participação remota

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória/ES, 27 de outubro de 2021.

amar é um elo
entre o azul
e o amarelo

Paulo Leminski

Agradecimentos

Aos meus pais que sacrificaram muitas vezes seus sonhos para que os meus fossem realizados e, especialmente, a minha mãe que superou seus próprios desafios.

À minha esposa, Patricia Stelzer da Cruz, pelo apoio, paciência e incentivo constantes.

Aos professores Geraldo Regis Mauri, Maria Claudia Silva Boeres, Rodrigo de Alvarenga Rosa pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos professores do Programa de Pós-Graduação em Informática da UFES pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

Resumo

Esta tese propõe uma matheurística baseada na abordagem *Route-First-Cluster-Second* (RFCS) usando *Greedy Randomized Adaptive Search Procedure* (GRASP), modelos matemáticos e *Variable Neighborhood Search* (VNS) para resolver dois tipos de problemas de roteamento de veículos: o *Capacitated Vehicle Routing Problem* (CVRP) e o *Helicopter Routing Problem* (HRP). Inicialmente, no método proposto, o roteamento é realizado usando heurísticas construtivas e um Problema de Cobertura de Conjuntos (PCC). O PCC emprega as soluções localmente ótimas encontradas em iterações prévias do VNS para criar um *tour* parcial, o qual é completado por heurísticas construtivas se necessário. Em seguida, a solução construída passa por uma busca local pelo VNS. Esse processo é repetido no laço principal do GRASP. Após a finalização deste, um Problema de Particionamento de Conjuntos (PPC) gera a solução final da matheurística usando as soluções localmente ótimas encontradas no laço principal do GRASP. Em relação aos problemas estudados, o CVRP consiste em gerar um conjunto de rotas para atender um conjunto de requisições de transporte com o apoio de uma frota de veículos idênticos. O objetivo é gerar rotas cuja soma das distâncias seja mínima. Já o HRP visa atender um conjunto de requisições de transporte, definidas como um par de locais de embarque e de desembarque, usando helicópteros como meio de transporte. O objetivo é atender todas as requisições com o menor custo possível. Além disso, esta tese também propõe um novo modelo matemático para o HRP que inclui novas características no atendimento de plataformas marítimas, as quais resolvem um problema até o momento em aberto: plataformas marítimas só possuem capacidade de atender um helicóptero por vez. Nesse novo modelo permite-se, também, que os helicópteros executem várias viagens por dia e que as requisições de transporte possuam janelas de tempo. Além disso, o modelo contém restrições relacionadas ao tempo, ao consumo de combustível, ao peso transportado e ao uso de assentos nas aeronaves. A matheurística é proposta em duas versões. A primeira resolve o CVRP e é avaliada em sete *benchmarks* do problema usando outras heurísticas da literatura como comparação. A segunda versão é aplicada em dois modelos do HRP, nos quais o método é testado em 37 instâncias com até 1000 requisições. Os experimentos computacionais mostram que o algoritmo proposto para o CVRP é competitivo em qualidade com as soluções publicadas em trabalhos recentes. Além disso, nas aplicações no HRP, a matheurística conseguiu resultados iguais ou superiores para 36 instâncias quando comparada a outros métodos da literatura.

Palavras-chaves: Roteamento de Veículos Capacitados. Roteamento de Helicópteros. *Greedy Randomized Adaptive Search Procedure* (GRASP). *Variable Neighborhood Search* (VNS). Problema de Cobertura de Conjuntos (PCC). Problema de Particionamento de Conjuntos (PPC).

Abstract

This work develops a hybrid matheuristic based on the approach *Route-First-Cluster-Second* (RFCS) by applying *Greedy Randomized Adaptive Search Procedure* (GRASP), mathematical models and *Variable Neighborhood Search* (VNS) to tackle two types of vehicle routing problems: the *Capacitated Vehicle Routing Problem* (CVRP) and the *Helicopter Routing Problem* (HRP). At first, in the proposed method, a routing is performed using constructive heuristics and the *Set Covering Problem* (SCP). SCP employs local optima solutions found in previous iterations of VNS to create a partial tour which is filled by a constructive heuristic if needed. Then, the built solution undergoes a local search phase by VNS. This process is repeated as the main loop of the GRASP. As last step of the method, the *Set Partitioning Problem* (SPP) provides a new improved solution with regard to solutions found in the GRASP. In relation to the study problems, the CVRP consists of designing a set of routes for a fleet of identical vehicles to attend a set of customers at shortest distance travelled, while the HRP aims of serving a set of transportation requests, defined as a pair of boarding and landing locations, using helicopters as the mode of transportation to minimize the cost of meeting the set of transportation requests. Besides, we propose a new HRP model to address unique characteristics of offshore platforms through a novel constraint to solve an issue that remains unnoticed until now: offshore platforms can be visited by helicopters one at a time. We also add the possibility to make multiple trips inside each route and enforce time window to attend passengers. Besides, the model has restrictions regarding to the total time of the flight, the fuel consumption, the total weight during the flight, the number of seats used by passengers. The matheuristic is proposed in two versions. The first one solves the CVRP, and it is tested in seven benchmarks using other heuristics in the literature as a comparison. The second version is applied in two models of the HRP, where the method is tested in 37 instances with up to 1000 requests. Computational experiments showed that the proposed matheuristic for CVRP is competitive in terms of the quality for solutions reported in recent works. Moreover, in relation to the HRP, the matheuristic achieves results equal to or greater than other heuristics in 36 instances.

Key-words: *Capacitated Vehicle Routing Problem* (CVRP). *Helicopter Routing Problem* (HRP). *Greedy Randomized Adaptive Search Procedure* (GRASP). *Variable Neighborhood Search* (VNS). *Set Covering Problem* (SCP). *Set Partitioning Problem* (SPP)

Lista de tabelas

Lista de ilustrações

Índice de algoritmos

Lista de abreviaturas e siglas

ACO *Ant Colony Optimization*

AEVRP *Anticipatory Emergency Vehicle Routing Problem*

AG Algoritmo Genético

ALNS *Adaptive Large Neighborhood Search*

AM Algoritmo Memético

BB *Branch-and-bound*

BC *Branch-and-cut*

BLFCP *Base Location and Fleet Composition Problem*

BLG Busca Local Gulosa

BP *Branch-and-price*

BPP-1 *One-Dimensional Bin Packing Problem*

BT Busca Tabu

CFRS *Cluster-First-Route-Second*

CHRP *Capacitated Helicopter Routing Problem*

CS *Clustering Search*

CVRP *Capacitated Vehicle Routing Problem*

CW *Clark and Wright*

DARP *Dial-A-Ride Problem*

GC Geração de Colunas

GES *Grouping Evolution Strategy*

GPDP *General Pickup and Delivery Problems*

GRASP *Greedy Randomized Adaptive Search Procedure*

HRP *Helicopter Routing Problem*

HRPM *Helicopter Routing Problem with Multiple Trips*

HRPS *Helicopter Routing Problem with Single Trip*

HS Heurística

ILS *Iterated Local Search*

Irace *Iterated Racing Package*

LCA *League Championship Algorithm*

LCP *Location Convering Proplem*

LNS *Large Neighborhood Search*

MGV-C Matheurística GRASP+VNS para o CVRP

MGV-H Matheurística GRASP+VNS para o HRP

OPTP *Oil Platform Transport Problem*

PCC Problema de Cobertura de Conjuntos

PD Programação Dinâmica

PDVRP *Pickup and Delivery Vehicle Routing Problem*

PEO *Penguin Optimization*

PIB Programação Inteira Binária

PI-PCC Programação Inteira para o Problema de Cobertura de Conjuntos

PI-PPC Programação Inteira para o Problema de Particionamento de Conjuntos

PI Programação Inteira

PLIM Programação Linear Inteira Mista

PL Programação Linear

PPC Problema de Particionamento de Conjuntos

PPI Problema de Programação Inteira

PQB Programação Quadrática Binária

PSO *Particle Swarming Optimization*

RFCS *Route-First-Cluster-Second*

RL-PCC Relaxação Linear do Problema de Corbetura de Conjuntos

SA *Simulated Annealing*

SCP *Set Covering Problem*

SOP *Sequential Ordering Problem*

SPP *Set Partitioning Problem*

SWP *Sweep*

TSP *Traveling Salesman Problem*

VND *Variable Neighborhood Descent*

VNS *Variable Neighborhood Search*

VRP *Vehicle Routing Problem*

VRPTW *Vehicle Routing Problem with Time Windows*

Sumário

1 Introdução

O *Vehicle Routing Problem* (VRP) é um conhecido problema combinatorial formulado, inicialmente, pelos trabalhos de Dantzig e Ramser (1959) e Clarke e Wright (1964). Nele, uma frota de veículos deve ser roteada para atender um conjunto de requisições com o menor custo possível. Dada sua aplicabilidade no mundo real, existem diversos tipos de VRP segundo as exigências operacionais da área de interesse (LAPORTE, 1992; BRAEKERS; RAMAEKERS; VAN NIEUWENHUYSE, 2016). O CVRP, por exemplo, é um caso da classe VRP que tem atraído considerável atenção na comunidade científica nos últimos anos (PRINS, 2009; HOSSEINABADI et al., 2017). Ele tem aplicação direta em várias atividades econômicas, visto que o transporte é fundamental para que agentes econômicos executem as suas atividades ao menor custo possível (MUTAR et al., 2020). Nesse sentido, um caso especial do CVRP com aplicação na indústria de exploração de petróleo surge quando os veículos utilizados no roteamento são helicópteros.

Helicópteros são um meio de transporte utilizado em várias atividades civis em que o deslocamento deve ser flexível e rápido. A principal área de aplicação é no negócio de exploração de petróleo em alto-mar, onde a opção por traslado aéreo implica numa redução no tempo de comutação e uma melhoria do conforto de funcionários, quando comparado ao transporte por mar (QIAN; GRIBKOVSKAIA; HALSKAU SR, 2011). Além disso, no Brasil e na Noruega, novos campos de exploração recentemente descobertos distam mais de 300 km da costa, aumentando a necessidade desta categoria de transporte (BRACHNER; HVATTUM, 2017). Uma área com crescentes aplicações é a logística aplicada a operações humanitárias e desastres naturais em que o gerenciamento de uma crise impõe um conjunto de problemas de decisão relacionado ao bem-estar das pessoas afetadas. Uma questão importante neste caso é projetar a transferência de bens e pessoas a áreas geograficamente dispersas, geralmente executada por helicópteros (BARBAROSOĞLU; ÖZDAMAR; CEVIK, 2002).

Na indústria de petróleo, o transporte de funcionários para plataformas em alto-mar é o principal utilizado há décadas (QIAN; GRIBKOVSKAIA; HALSKAU SR, 2011). Apesar disso, o primeiro trabalho a analisar o problema de transporte por helicópteros para este ramo somente foi publicado no começo da década de 1990 (GALVÃO; GUIMARÃES, 1990). Desde então, diversos trabalhos usando modelos matemáticos, meta-heurísticas e hibridizações foram sugeridos. Em geral, neste tipo de problema, o objetivo é minimizar o custo operacional da utilização de aeronaves ao atender um conjunto de requisições de transporte de funcionários, as quais têm origem ou destino em plataformas marítimas (BRACHNER; HVATTUM, 2017). Mais recentemente, devido aos riscos inerentes do transporte aéreo (QIAN; GRIBKOVSKAIA; HALSKAU SR, 2011), há um esforço em incluir nos modelos restrições relacionadas à segurança (HALSKAU, 2014; GRIBKOVSKAIA; HALSKAU; KOVALYOV, 2015; BRACHNER; HVATTUM, 2017), principalmente as relativas ao número de pousos e decolagens (HALSKAU, 2014).

O problema de roteamento de helicópteros é designado por Rosero e Torres (2006) como *Helicopter Routing Problem* (HRP), o qual consiste na construção de rotas factíveis a serem percorridas por helicópteros de modo a satisfazer requisições de transporte. De acordo com esta definição, o HRP pode ser classificado como um caso particular dos *General Pickup and Delivery Problems* (GPDP). Na classificação original de Savelsbergh e Sol (1995), o GPDP subdivide-se em três tipos de problemas: o *Pickup and Delivery Vehicle Routing Problem* (PDVRP), o *Dial-A-Ride Problem* (DARP) e o VRP. Por outro lado, Díaz-Parra et al. (2017) definem a classe *Oil Platform Transport Problem* (OPTP) como uma combinação do HRP, representando o roteamento de helicópteros, e o *One-Dimensional Bin Packing Problem* (BPP-1), correspondendo à designação dos passageiros aos assentos das aeronaves. Em outros trabalhos, este tipo de problema é designado como *Capacitated Helicopter Routing Problem* (CHRP) (FIALA TIMLIN; PULLEYBLANK, 1992). Nesta tese, denomina-se como HRP todo problema de roteamento de helicópteros, com ou sem restrições relacionadas à capacidade de passageiros.

Diversos métodos foram propostos para resolver o HRP. Modelos matemáticos existem para distintas formulações (QIAN; GRIBKOVSKAIA; HALSKAU SR, 2011; OZDAMAR, 2011; HALSKAU, 2014; GRIBKOVSKAIA; HALSKAU; KOVALYOV, 2015; ABBASI-POOYA; KASHAN, 2017; BRACHNER; HVATTUM, 2017). Entretanto, estes paradigmas exatos são capazes de resolver somente instâncias pequenas, pois o problema HRP é NP-difícil (DÍAZ-PARRA et al., 2017). Por isso, normalmente estas formulações vêm acompanhadas de heurísticas ou meta-heurísticas (ARMSTRONG-CREWS; MOCK, 2005; VELASCO et al., 2009; MOTTA; VIEIRA; SOLETTI, 2011; QIAN et al., 2012; ANDREEVA-MORI; KOBAYASHI; SHINDO, 2015; ABBASI-POOYA; KASHAN, 2017).

As abordagens de resolução utilizadas por heurísticas ou meta-heurísticas para problemas de roteamento, incluindo o HRP, podem ser classificadas em dois grandes grupos (PRINS; LACOMME; PRODHON, 2014). O primeiro representa o paradigma *Cluster-First-Route-Second* (CFRS). Nesse enfoque, as requisições são designadas a distintos grupamentos e, posteriormente, cada grupamento é designado a um único veículo. Na segunda abordagem, chamada *Route-First-Cluster-Second* (RFCS), as requisições dos clientes são agrupadas numa sequência de atendimento. Em seguida, essa sequência é dividida em pequenos trechos, os quais representam o atendimento realizado por cada veículo disponível. O paradigma RFCS também é denominado como abordagem hierárquica.

1.1 Motivação

Atualmente, uma das principais preocupações está na operação segura de helicópteros em plataformas marítimas, pois é onde há o maior risco de acidente (BRACHNER; HVATTUM, 2017). Segundo Gribkovskaia, Halskau e Kovalyov (2015), a maioria desses eventos ocorre no pouso ou na decolagem. Nesse sentido, um aspecto ainda não presente nos modelos do HRP,

até a publicação do trabalho de Machado et al. (2019) oriundo desta tese, é a exigência das aeronaves executarem o atendimento em plataformas sem a concorrência de outros helicópteros. A incorporação dessa exigência aumenta o nível de segurança ao planejar o atendimento das requisições.

Além disso, segundo a pesquisa bibliográfica realizada para HRP, os trabalhos de Velasco et al. (2009) e Abbasi-Pooya e Kashan (2017) são as únicas aplicações do paradigma RFCS para o HRP, mas com algumas restrições. Em Velasco et al. (2009), a criação de rotas via *splitting* pode gerar soluções inviáveis, as quais são descartadas pelo método. O motivo é que se usa um algoritmo adequado para o CVRP, mas inadequado para problemas com restrições de *pickup* e *delivery*. Em Abbasi-Pooya e Kashan (2017), a abordagem é usada somente para permitir a criação de uma solução inicial. Após isso, todas as operações do algoritmo trabalham com o mesmo tipo de solução.

Assim, nesta tese, propõe-se uma nova formulação matemática do HRP, na qual sejam incluídas novas restrições para os pousos e decolagens em plataformas marítimas, múltiplas viagens por helicóptero e janelas de tempo. Esse modelo também possui restrições relacionadas ao tempo de execução da rota, à capacidade de ocupação nos veículos, ao consumo de combustível, ao peso transportado e ao combustível reserva. Além disso, propõe-se, também, uma matheurística conforme a abordagem hierárquica RFCS como método de resolução. Essa nova matheurística é avaliada nos problemas do tipo CVRP e, posteriormente, adaptada para os modelos HRP definidos por Rosa et al. (2016) e por esta tese.

1.2 Objetivos

Esta seção é dividida em objetivo geral (Subseção 1.2.1) e objetivos específicos (Subseção 1.2.2)

1.2.1 Objetivo Geral

Esta tese tem como objetivo geral desenvolver uma nova formulação matemática para o HRP e um novo método de resolução, o qual é validado no CVRP e aplicado no HRP proposto.

1.2.2 Objetivos Específicos

Considerando o objetivo geral deste trabalho, os objetivos específicos são apresentados a seguir:

- Desenvolver um novo modelo matemático, alternativo aos do estado da arte do problema, que englobe restrições operacionais inéditas para o HRP;
- Desenvolver uma nova matheurística com as meta-heurísticas GRASP e VNS;

- Validar a nova abordagem na classe de problemas CVRP;
- Adaptar e aplicar a nova abordagem na classe de problemas HRP.

1.3 Publicações

No decorrer do desenvolvimento dessa tese, as seguintes publicações foram produzidas:

- A MILP Model and a GRASP Algorithm for the Helicopter Routing Problem with Multi-Trips and Time Windows (2019) publicado no congresso *ICCL 2019 - International Conference on Computational Logistics*.
- A New Hybridization of Evolutionary Algorithms, GRASP and Set-Partitioning Formulation for the Capacitated Vehicle Routing Problem (2020) publicado no congresso *BRACIS 2020 - Brazilian Conference on Intelligent Systems*;
- A new hybrid matheuristic of GRASP and VNS based on constructive heuristics, set-covering and set-partitioning formulations applied to the capacitated vehicle routing problem (2021) publicado na revista *Expert Systems with Applications*.

1.4 Estrutura do Trabalho

Além deste Capítulo 1, esta tese possui mais sete capítulos descritos a seguir. No Capítulo 2 é apresentada a revisão bibliográfica dos problemas VRP e HRP. No Capítulo 3 é apresentado o modelo clássico do CVRP, assim como os dois modelos propostos para o HRP. No Capítulo 4 apresentam-se os métodos básicos para resolver os problemas de roteamento. No Capítulo 5 descreve-se o método proposto, chamado MGV-C, para resolver o problema CVRP através do uso de matheurística, abordagem hierárquica e hibridizações. No Capítulo 6, descreve-se a adaptação da matheurística MGV-C a problemas HRP. No Capítulo 7, os resultados computacionais da aplicação do MGV-C na resolução dos problemas CVRP e HRP são apresentados e discutidos. O Capítulo 8 apresenta as considerações finais do trabalho e trabalhos futuros.

A tese também possui quatro apêndices. No Apêndice A são apresentados as definições e os conceitos gerais que são utilizados no restante do trabalho. No Apêndice B exemplifica-se a aplicação da matheurística MGV-C num problema CVRP. No Apêndice C analisa-se a contribuição dos principais componentes do MGV-C. No Apêndice D apresentam-se os detalhes de como executar os movimentos da busca local aplicada à classe de problemas HRP.

2 Trabalhos Correlatos

Neste capítulo são analisadas as classes de problemas do *Vehicle Routing Problem* (VRP) e algumas das suas variações. A seguir, o *Capacitated Vehicle Routing Problem* (CVRP) é apresentado com uma revisão bibliográfica dos principais trabalhos. Na seção subsequente são apresentados os trabalhos relacionados ao *Helicopter Routing Problem* (HRP) identificados na pesquisa bibliográfica executada. A partir dessa revisão, são discutidos os principais pontos e propriedades das propostas para o HRP da literatura.

2.1 Introdução

O *Vehicle Routing Problem* (VRP) é um conhecido problema combinatorial que atrai o interesse da comunidade científica desde os trabalhos seminais de Dantzig e Ramser (1959) e Clarke e Wright (1964). De maneira geral, o VRP é definido como o problema de roteamento de veículos com um depósito, um conjunto de clientes ou requisições, uma frota de veículos e o objetivo de minimizar o custo total de atendimento das requisições usando os veículos disponíveis. Além disso, outras restrições ou características podem ser adicionadas ao problema. Por isso, centenas de novos modelos e algoritmos foram propostos para encontrar soluções ótimas ou aproximadas para cada tipo de VRP (LAPORTE, 1992; BRAEKERS; RAMAEKERS; VAN NIEUWENHUYSE, 2016).

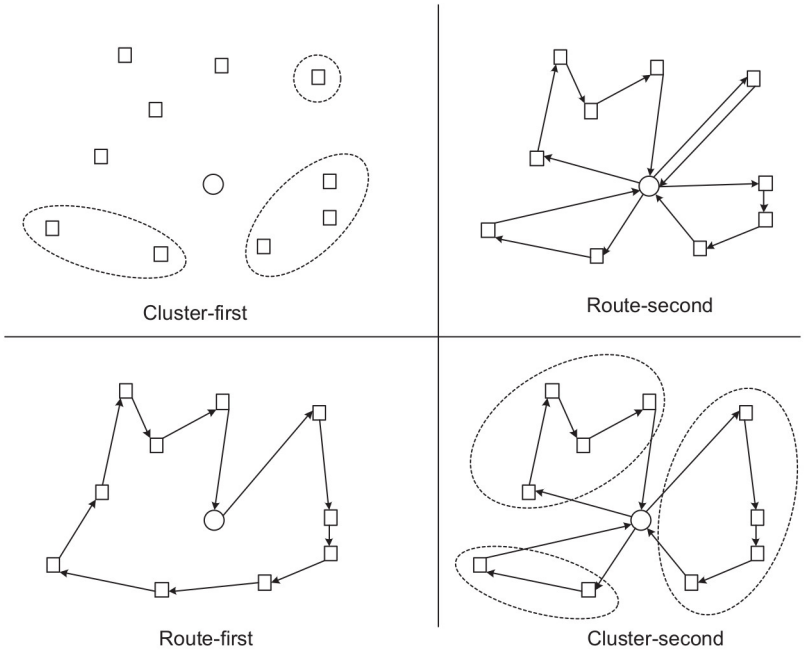
As atuais variantes do VRP surgem da necessidade de incorporar as novas características que nascem no mundo real (BRAEKERS; RAMAEKERS; VAN NIEUWENHUYSE, 2016). De acordo com Braekers, Ramaekers e Van Nieuwenhuyse (2016), a literatura do VRP cresce exponencialmente a uma taxa de 6% ao ano. Como exemplo dessa progressão, podem ser citadas como variações do VRP: i. a adição de restrições para exigir o atendimento de requisições dentro de janelas de tempo; ii. capacidade máxima de veículos e iii. se é obrigatório o retorno do veículo ao depósito no fim da rota (CACERES-CRUZ et al., 2014). A inclusão dessas novas restrições implica em alterações no modelo matemático e isso criou diversos ramos de pesquisas do VRP com suas próprias abreviações.

Algumas classes de VRP podem ser vistas como a composição de distintas instâncias em VRP (TOTH; VIGO, 2002b). Duas delas que, geralmente, são modelos para criação de novos tipos são *Pickup and Delivery Vehicle Routing Problem* (PDVRP) e *Vehicle Routing Problem with Time Windows* (VRPTW). No PDVRP um conjunto de requisições R deve ser atendido por um conjunto fixo de veículos K. Cada requisição $r \in R$ está associada com dois nós na rede: um representando uma demanda a ser carregada no embarque (*pickup*) do local r_i e entregue no desembarque (*delivery*) no local r_j (BRAEKERS; RAMAEKERS; VAN NIEUWENHUYSE, 2016). As componentes de *pickup* e *delivery* devem ser realizadas pelo mesmo veículo e o *pickup* deve

ser executado antes do *delivery*. Além disso, a capacidade dos veículos não pode ser violada durante o atendimento. O objetivo é encontrar um conjunto de rotas que minimize a distância total e atenda a todas as requisições (PISINGER; ROPKE, 2007; CACERES-CRUZ et al., 2014). No VRPTW um conjunto de requisições R deve ser atendido por uma frota K de veículos. Cada requisição $r \in R$ deve ser atendida na janela de tempo $[L_r, U_r]$. O objetivo é minimizar o custo total de atender todas as requisições R. Além disso, em alguns problemas, as janelas de tempo podem ser relaxadas com inclusão de uma penalização na função objetivo (CACERES-CRUZ et al., 2014).

As técnicas de solução para o VRP podem ser divididas em duas classes (PRINS; LACOMME; PRODHON, 2014): *Cluster-First-Route-Second* (CFRS) e *Route-First-Cluster-Second* (RFCS). Na abordagem *Cluster-First-Route-Second* (CFRS), os clientes $i \in N$ são designados a grupos distintos e então cada grupo é resolvido como um TSP. No processo de designação é garantido que um veículo executando o atendimento das requisições nesse grupo não viola nenhuma das restrições do problema. Na abordagem hierárquica ou *Route-First-Cluster-Second* (RFCS), primeiro as requisições $i \in N$ são designadas a um percurso fechado único, ou seja, uma solução do TSP. Para isso, as restrições relacionadas aos clientes ou veículos são relaxadas. Formado esse primeiro percurso, um procedimento chamado *splitting* o transforma em rotas viáveis, restabelecendo as restrições do problema. A Figura 1 representa as duas abordagens CFRS e RFCS.

Figura 1 – Técnicas de solução para os problemas VRP.



Fonte: Adaptado de Prins, Lacomme e Prodhon (2014).

No *survey* de Prins, Lacomme e Prodhon (2014) são revisados os artigos de RFCS publicados entre 1984 e 2014. Segundo os autores, 43% das publicações usam versões básicas do

procedimento *splitting*, em que somente a capacidade do veículo é analisada no processo de transformação do percurso fechado em várias rotas viáveis. Por volta de 35% dos trabalhos adicionam outras exigências como janelas de tempo ou veículos heterogêneos. Adaptações mais exigentes ou utilizando outras formas de *splitting* ainda são escassas, apesar do potencial de aplicações a diversos tipos de problemas VRP. Isso se reflete no caso de problemas PDVRP, no qual o *survey* somente cita os trabalhos de Mosheiov (1998) e Velasco et al. (2009) como casos de aplicação. A técnica de RFCS teve sua aplicação inicial em problemas CVRP. Na próxima seção, essa classe de VRP é analisada em detalhes.

2.2 O *Capacitated Vehicle Routing Problem* (CVRP)

O CVRP é um dos problemas mais investigados da classe VRP e, por isso, tem atraído considerável atenção na comunidade científica nos últimos anos (PRINS, 2009; HOSSEINABADI et al., 2017). Ele tem aplicação direta em várias atividades como distribuição de bens, coleta de lixo e logística aplicada a cidades, visto que o transporte é fundamental para que agentes econômicos executem as suas atividades ao menor custo possível (MUTAR et al., 2020).

Esse problema consiste em, dada uma frota de veículos com capacidade fixa, planejar um conjunto de rotas para atender a um conjunto de clientes na menor distância possível. O CVRP é classificado como NP-difícil (TOTH; VIGO, 2014). As aplicações no mundo real do CVRP são extensas e, por isso, muitas heurísticas e meta-heurísticas foram propostas para essa classe de VRP. Essas abordagens envolvem, entre outros, algoritmos evolucionários (WANG; LU, 2009), *Simulated Annealing* (SA) (KAMPKE; ARROYO; SANTOS, 2009), Busca Tabu (BT) (ZHU et al., 2012) e *Large Neighborhood Search* (LNS) (AKPINAR, 2016). No restante dessa seção são mostrados alguns trabalhos relacionados ao CVRP.

Toth e Vigo (2002a) realizaram uma revisão da literatura de algoritmos exatos baseados no método *Branch-and-bound* (BB) para o CVRP, todos propostos entre 1980 e 2000. Os resultados computacionais compararam o desempenho de diferentes relaxações e algoritmos num conjunto de instâncias clássicas. Os autores ressaltaram que os novos métodos analisados permitem que instâncias de até 100 requisições sejam resolvidas otimamente, enquanto os métodos da geração anterior são capazes de resolver instâncias com até 25 requisições.

Berger e Barkaoui (2003) aplicaram um Algoritmo Genético (AG) híbrido para resolver instâncias do CVRP. A abordagem proposta implica em evoluir dois conjuntos de populações que minimizam a distância total percorrida. Os melhores indivíduos de cada população são permutados a cada geração. Para evoluí-las, operadores genéticos foram construídos utilizando estratégias chaves usadas para resolver problemas de roteamento e técnicas de busca no espaço de solução. Segundo os testes realizados, a proposta é competitiva com os resultados dos *benchmarks* clássicos em 2003.

Toth e Vigo (2003) propuseram uma nova variante do BT chamada BT Granular e a

aplicaram a diversos problemas VRP, inclusive o CVRP. Nesse novo método, utilizam-se diversas estratégias de diversificação e intensificação que podem ser aplicadas a diversas classes de problemas que utilizam grafos. A ideia subjacente é restringir o número de vizinhos numa solução ao impedir movimentos que contêm elementos que, geralmente, não podem pertencer a boas soluções. Essa nova vizinhança, chamada de granular, é uma implementação específica da estratégia de lista de candidatos utilizada por outras heurísticas BT. Além disso, o trabalho mostra como encontrar vizinhos granulares a partir de vizinhanças clássicas para problemas de roteamento. Nos testes computacionais, o algoritmo proposto conseguiu encontrar boas soluções com pouco tempo computacional. Segundo os autores, sem a utilização da vizinhança granular, o tempo dispendido para encontrar a mesma solução é excessivo.

Lysgaard, Letchford e Eglese (2004) apresentaram um novo algoritmo *Branch-and-cut* (BC) para o CVRP. O método proposto usa diversos planos de corte, incluindo vários tipos relacionadas à capacidade, desigualdades e cortes de Gomory. Para cada um desses cortes, um algoritmo é proposto em detalhes. Além disso, os autores também apresentaram as regras de ramificação do BC, as estratégias de seleção de nós e o gerenciamento dos cortes. O resultado dos testes computacionais mostrou que o algoritmo proposto é competitivo com os *benchmarks* em 2004. Em especial, ele foi capaz de encontrar o ótimo pela primeira vez em três instâncias do *dataset* de Augerat (AUGERAT et al., 1995).

Altınel e Öncan (2005) modificaram a heurística *Clark and Wright* (CW) para incorporar as demandas dos clientes no algoritmo. Segundo os autores, as meta-heurísticas, geralmente, são superiores às heurísticas para resolver problemas de roteamento. Entretanto, métodos como o CW são rápidos e fáceis de implementar, o que explica a sua popularidade para resolver diversos VRP. A partir dos testes computacionais da versão modificada do CW, os autores afirmaram que o novo método é, além de rápido, bastante satisfatório em relação a outras heurísticas construtivas disponíveis em 2005.

Tavakkoli-Moghaddam, Safaei e Gholipour (2006) apresentaram um novo modelo de Programação Linear Inteira Mista (PLIM) para o CVRP para minimizar o custo de uma frota heterogênea de veículos e maximizar a utilização da capacidade dos veículos. No modelo proposto, o custo da frota é independente do tamanho da rota e há janelas de tempo para o depósito. Além disso, mostra-se que o modelo permite a geração de rotas que servem todos os clientes com o mínimo de veículos usados. Para resolver o PLIM para instâncias grandes, a meta-heurística SA é aplicada gerando bons resultados.

Christiansen e Lysgaard (2007) desenvolveram um novo algoritmo exato para o CVRP com demandas estocásticas usando o *Branch-and-price* (BP). O CVRP é formulado como o Problema de Particionamento de Conjuntos (PPC) e o novo método aplica Geração de Colunas (GC) para resolvê-lo. O subproblema do GC é resolvido usando programação dinâmica. Nos testes computacionais, o algoritmo proposto obtém melhores soluções quando as restrições de capacidade são mais rígidas.

Kwon et al. (2007) propuseram uma heurística BT que incorpora um método de redução de vizinhança usando a informação de proximidade do diagrama de Voronoi. O método proposto é chamado VTS e, conforme os autores, foi a primeira vez que esse tipo de abordagem foi aplicada ao CVRP. Os experimentos computacionais foram realizados em *benchmarks* clássicos e os resultados mostraram que a proposta é competitiva com os algoritmos existentes na época da publicação.

Borgulya (2008) apresentou um algoritmo evolucionário multiobjetivo para o CVRP com balanceamento de rotas. Nessa versão do problema, além de minimizar o custo total das rotas, um objetivo adicional é manter as rotas com tamanhos similares. O algoritmo proposto é baseado num método de memória coletiva explícita conhecido como *extended virtual loser*. Essa abordagem é adaptada e aprimorada para resolver o CVRP com rotas balanceadas usando novos operadores de mutação e novas funções de seleção. Nos testes computacionais, bons resultados foram encontrados para instâncias clássicas e foram similares aos algoritmos evolutivos disponíveis em 2009.

Ai e Kachitvichyanukul (2009) aplicaram o algoritmo evolutivo *Particle Swarming Optimization* (PSO) ao CVRP. Para isso, os autores desenvolveram duas representações alternativas de soluções, chamadas SR-1 e SR-2, e suas correspondentes funções de decodificação. Na primeira representação, S-1, as partículas do PSO são definidas como vetores de dimensões $(n + 2m)$, em que n é o número de clientes e m o número de veículos do problema. A função de decodificação inicia com a transformação de uma partícula numa lista de prioridades dos clientes para entrar na rota e uma matriz de prioridades dos veículos que podem servir os clientes. A partir dessas listas e matrizes, as rotas são construídas. Na segunda representação, SR-2, a partícula é definida como uma vetor de tamanho $3m$. Para essa representação, a função decodificadora projeta as partículas no plano cartesiano munido de um raio de atendimento. As rotas são construídas segundo o raio e a localização de cada veículo. Os resultados computacionais mostraram que ambas representações são efetivas para o CVRP e superam, em 2009, outras abordagens utilizando o PSO.

Prins (2009) foi o primeiro a ressaltar a aplicabilidade das abordagens hierárquicas como método eficaz para problemas de roteamento. Nesse trabalho, chamado GRELS, a principal característica é a alternância entre *tours* TSP e soluções do CVRP. A proposta realizada é a aplicação de uma busca local evolucionária e procedimentos de busca local inseridos num GRASP. Os testes computacionais mostraram que o algoritmo supera, em qualidade de solução, as heurísticas disponíveis em 2009 na literatura.

Brito et al. (2009) propuseram um GRASP hibridizado com VNS com janelas de tempo, em que o tempo de viagem são números *fuzzy* triangulares para problema de roteamento de veículos. Os testes computacionais compararam a abordagem híbrida com as soluções obtidas, separadamente, pelo GRASP ou pelo VNS usando instâncias da literatura. Em ambos casos, os melhores resultados foram alcançados quando o GRASP hibridizado com VNS é usado.

Lee et al. (2010) aplicaram uma nova versão do *Ant Colony Optimization* (ACO) ao CVRP. O objetivo do trabalho foi propor um *framework* de resolução, no qual a solução inicial é construída pelo SA e a meta-heurística ACO é usada para encontrar uma solução localmente ótima. O motivo de usar SA e ACO, segundo os autores, é que o primeiro, geralmente, encontra soluções iniciais de boa qualidade que podem ser aprimoradas pelo segundo. Para validar o *framework*, 34 instâncias de teste foram utilizadas, sendo 14 instâncias pequenas e 20 instâncias grandes. Para as instâncias pequenas, o método sempre encontrou a melhor solução conhecida para oito execuções, enquanto para as instâncias grandes ele encontrou, pelo menos uma vez, a melhor solução.

Szeto, Wu e Ho (2011) propuseram duas heurísticas ACO para resolver o CVRP. Nele, uma versão com adaptação mais simples do ACO é proposta e outra, mais específica para o CVRP, a qual considera propriedades do problema, também é desenvolvida para encontrar soluções mais robustas. Nos testes computacionais, dois *datasets* clássicos foram utilizados para avaliação das heurísticas. Os resultados encontrados mostraram que a versão específica do ACO é superior à primeira versão do ACO. Além disso, a específica encontrou boas soluções quando comparada com as heurísticas disponíveis em 2011.

Groër, Golden e Wasil (2011) desenvolveram um algoritmo, chamado RRTR, que incorpora as novas funcionalidades dos ambientes de alto desempenho de multiprocessadores para solucionar diversos tipos de problemas VRP. O algoritmo paralelo proposto designa alguns processadores para heurísticas locais, enquanto outros tentam combinar rotas de diferentes soluções em novas soluções. Os testes computacionais mostraram que o método desenvolvido gera soluções competitivas com os melhores valores conhecidos na literatura em 2011.

Babu, Poojary e Renuka (2012) desenvolveram uma hibridização do SA e do ACO que obtém resultados superiores aos obtidos isoladamente pelo SA ou ACO. No algoritmo proposto, o SA é usado para gerar soluções iniciais viáveis, enquanto o ACO é usado para realizar a exploração do espaço de busca e evitar a convergência prematura para uma solução localmente ótima.

Vidal et al. (2012) propuseram uma meta-heurística chamada HGSADC que combina a exploração em largura da busca evolucionária, as potencialidades das meta-heurísticas baseadas em vizinhança e esquemas para diversificar populações. Os experimentos computacionais mostraram que o procedimento proposto supera, em qualidade das soluções, algoritmos conhecidos para o CVRP na data da publicação.

Xiao et al. (2012) apresentaram um novo modelo matemático do CVRP que incorpora o consumo de combustível na função objetivo do problema. Segundo os autores, o consumo de combustível representa uma fatia grande dos custos relacionados ao transporte no mundo real. Para resolver o modelo, a heurística SA é aplicada tanto no modelo tradicional quanto no modelo estendido. Além disso, os autores analisam os principais motivos para a variação de consumo de combustível no roteamento de veículos. Nos testes computacionais, o SA foi

capaz de encontrar rotas que usam, em média, 5% a menos de combustível.

Zhou, Xie e Zheng (2013) apresentaram o algoritmo evolutivo *bat* com *path-relinking* chamado HBA-PR para o CVRP. O *bat* é uma meta-heurística proposta por Mishra, Shaw e Mishra (2012), na qual a inspiração é o comportamento de ecolocalização de morcegos quando estes estão em busca de alimentos. Na abordagem proposta, o método *bat* é hibridizado com o GRASP e utiliza *path-relinking*. Esse último é utilizado como uma estratégia de intensificação para explorar o espaço de solução que conecta duas soluções obtidas pelo método *bat*. Nos testes computacionais, diversos *benchmarks* foram usados. Segundo os autores, o método HBA-PR é efetivo para resolver o CVRP e compete com heurísticas e métodos exatos presentes em 2013.

Jin, Crainic e Løkketangen (2014) desenvolveram uma meta-heurística paralela e cooperativa para o CVRP usando BT. Nela, múltiplas execuções da BT são mantidas em paralelo com cooperação assíncrona, na qual as melhores soluções são permutadas através de um conjunto comum de soluções. Nesse conjunto, soluções similares são agrupadas segundo as suas características estruturais. Além disso, informações históricas geradas a partir da construção do conjunto de soluções é utilizado como estratégia para intensificar ou diversificar as *threads* da BT. Os autores também definiram um novo modo de inserir uma requisição numa solução ao incluir a informação de distância entre o depósito e a requisição como parâmetro do método. Isso permitiu melhorar o processo de exploração das vizinhanças. Nos experimentos computacionais, dois *benchmarks* com o total de 32 instâncias grandes foram usados para avaliar a meta-heurística e o método proposto foi capaz de encontrar dez instâncias com novos mínimos. Além disso, para o restante das instâncias, o algoritmo foi competitivo com as demais heurísticas da literatura em 2014.

Teoh, Ponnambalam e Kanagaraj (2015) definiram um método chamado DELS que combina algoritmos de evolução diferencial com busca local. A ideia é que este último explore novas áreas do espaço de busca e refine as soluções encontradas. A proposta dos autores gerou soluções comparáveis com os melhores resultados para os *benchmarks* da literatura em 2015.

Niu et al. (2015) aplicaram a heurística membrana com o ACO para resolver o CVRP. O algoritmo membrana é inspirado na estrutura e função das células vivas em organismos pluricelulares. A abordagem utilizada interpreta o sistema de membranas como um *framework* distribuído, paralelo e não determinístico. Cada membrana corresponde a aplicação de um ACO. As soluções encontradas por estes são intercambiadas em diferentes níveis das membranas. Nos experimentos computacionais, 13 instâncias do CVRP foram usadas para avaliar a heurística e para 7 instâncias foram encontrados novos limites inferiores.

Akpinar (2016) apresentou um novo algoritmo, chamado LNS-ACO, que hibridiza as heurísticas LNS e ACO. Este último é usado para diversificar soluções iniciais do CVRP geradas pelo LNS. Os testes computacionais foram realizados em *benchmarks* da literatura e o algoritmo proposto foi comparado a outras soluções usando o LNS como estratégia. Os re-

sultados mostraram que o método híbrido obteve desempenho superior às outras heurísticas disponíveis na época de publicação.

Haddadene, Labadie e Prodhon (2016) hibridizaram o GRASP com o *Iterated Local Search* (ILS) para o CVRP com janelas de tempo e restrições de tempo. Nesse problema, o requisitante pode solicitar mais de uma visita simultânea ou uma ordem de prioridade para sua requisição. Essas restrições de tempo tornam o problema mais realístico e, por consequência, mais difícil de resolver. Além disso, os experimentos foram executados em instâncias inéditas da literatura e os resultados mostraram, a partir de testes estatísticos, os benefícios da meta-heurística hibridizada.

Ammi e Chikhi (2016) propuseram uma nova meta-heurística chamada *Penguin Optimization* (PEO), a qual gerencia outras heurísticas como a AG e o ACO para resolver o CVRP. Na PEO, cada meta-heurística opera de maneira independente e soluções são permutadas, periodicamente, entre as diferentes meta-heurísticas. Para isso, um método chamado operador de migração é definido. Nos testes computacionais, vários estudos comparativos em instâncias do *benchmark* foram realizados para mostrar a eficiência da meta-heurística proposta.

Amous et al. (2017) hibridizaram o VNS com o *Variable Neighborhood Descent* (VND) para resolver o CVRP. O VND possui diferentes vizinhanças para intensificar e diversificar a exploração do espaço de busca. Ele é aplicado na fase de busca local do VNS para fazer uma busca em profundidade. Nos testes computacionais, a meta-heurística híbrida foi aplicada a vários *benchmarks* da literatura e obteve resultados superiores para distintas instâncias quando comparada a outros algoritmos disponíveis em 2017. Além disso, para algumas instâncias, encontraram-se novas soluções ótimas.

Kır, Yazgan e Tüncel (2017) hibridizaram a BT e o *Adaptive Large Neighborhood Search* (ALNS) com diversas técnicas para resolver o CVRP. Essas novas características incluem a possibilidade de soluções inviáveis, operadores de destruição e reparação no ALNS, distintas estratégias de diversificação e intensificação, além do uso de memória adaptativa. Nos experimentos computacionais, o método proposto foi avaliado em *benchmarks* da literatura. Ele foi capaz de encontrar novos ótimos para 3 instâncias e o ótimo conhecido para 11 instâncias. Em média, o método converge para a solução ótima com uma diferença de 0,01%. Além disso, os autores avaliaram o algoritmo num problema real com resultados satisfatórios.

Alinezhad et al. (2018) hibridizaram o PSO e o SA com diversas técnicas de inserção e de remoção para vários problemas VRP, inclusive o CVRP. Segundo os autores, a hibridização permitiu fazer um *tradeoff* entre exploração e competitividade para a convergência das soluções. Nos testes computacionais, os autores usaram diversos *benchmarks* com instâncias de vários tamanhos e o método foi capaz de encontrar soluções comparáveis às melhores em 2018 para vários tipos de VRP.

Linfati e Escobar (2018) desenvolveram uma heurística para reotimizar soluções do

CVRP. Segundo os autores, a área de roteamento é um contexto dinâmico e existe a necessidade de realizar inclusão/exclusão de requisições numa solução construída, especialmente quando a informação das requisições são disponibilizadas num horizonte de tempo. A heurística proposta usa métricas para diminuir a dispersão da rota e o tamanho total dela. Nela, a solução inicial é construída via o algoritmo CW. O processo de reotimização é feito considerando nós e arestas visitados como fixos. A partir disso, o algoritmo minimiza a dispersão e a distância total percorrida. Finalizada essa fase, uma busca local é realizada para encontrar o mínimo local. Nos testes computacionais, instâncias reais disponíveis em *benchmark* foram utilizadas e, segundo os autores, o método se mostrou efetivo para resolver diversas instâncias.

Altabeeb, Mohsen e Ghallab (2019) propuseram um novo algoritmo híbrido, chamado CVRP-FA, usando a heurística Firefly, busca local e operadores genéticos para resolver o CVRP. Os testes computacionais foram realizados usando 82 instâncias de vários *benchmarks*. Os resultados obtidos indicam a convergência consistente do CVRP-FA para soluções ótimas do CVRP.

Mais recentemente, Toffolo, Vidal e Wauters (2019) desenvolveram uma heurística para o CVRP baseada na decomposição entre designação e sequenciamento de rotas. Dado que o método exige esforço computacional excessivo, um espaço de busca intermediário foi introduzido usando Programação Dinâmica (PD) com o objetivo de fazer um compromisso entre a quantidade de esforço despendido e a quantidade do espaço de busca explorado. Um AG foi hibridizado para operar nesse espaço e, nos testes computacionais, novas solução foram encontradas para *benchmarks* da literatura.

Lin et al. (2019) desenvolveram um AG híbrido para Internet das Coisas formulado como o CVRP. O método proposto, chamado OHGA, usa uma estratégia de inicialização da solução usando o algoritmo *Sweep* (SWP). Além disso, os operadores de *crossover* do AG são combinados com heurísticas de busca local para encontrar soluções viáveis. Inúmeros testes foram realizados em instâncias de diversos *benchmarks* e os resultados mostraram a qualidade do método em relação aos algoritmos da literatura.

Sbai, Krichen e Limam (2020) propuseram uma meta-heurística híbrida, chamada HGA-VNS, que incorpora o VNS como operador de mutação do AG com o intuito de acelerar a convergência de soluções. Dessa maneira, a busca pode aumentar e diversificar a exploração do espaço de soluções. Os resultados do HGA-VNS indicam que o método é competitivo com os algoritmos do estado da arte. Além disso, um estudo de caso real mostra que a abordagem proposta aprimora, consideravelmente, as soluções conhecidas pelo serviço postal turco.

2.3 O *Helicopter Routing Problem* (HRP)

O HRP consiste em transportar pessoas ou bens de um local (aeroporto/plataforma) para outra localidade (aeroporto/plataforma) usando helicópteros como meio de transporte e

respeitando um conjunto de restrições de operação (ROSA et al., 2016). O objetivo é, em geral, minimizar o custo total de operação ou um *trade-off* entre custo e segurança. As restrições são, normalmente, relacionadas a capacidade dos veículos, janela de atendimento das requisições ou consumo de combustível. Por isso, o HRP é visto como um problema que incorpora características de vários outros VRP como o CVRP para a capacidade do veículo, o PDVRP para o atendimento das requisições de embarque (*pickup*) e desembarque (*delivery*) e o VRPTW para o horário de atendimento das requisições. Devido a essa característica, o HRP é um problema NP-difícil (DÍAZ-PARRA et al., 2017).

Helicópteros são um meio de transporte utilizado em várias atividades civis em que o deslocamento deve ser flexível e rápido. Na indústria de petróleo, o transporte de funcionários para plataformas em alto-mar é o principal modo utilizado há décadas (QIAN; GRIBKOVSKAIA; HALSKAU SR, 2011). Nessa área de aplicação, o objetivo é minimizar o custo operacional da utilização de aeronaves ao atender um conjunto de requisições de transporte de funcionários, as quais têm origem ou destino em plataformas marítimas (BRACHNER; HVATTUM, 2017). Mais recentemente, devido aos riscos inerentes do transporte aéreo (QIAN; GRIBKOVSKAIA; HALSKAU SR, 2011), há um esforço em incluir nos modelos restrições relacionadas à segurança (HALSKAU, 2014; GRIBKOVSKAIA; HALSKAU; KOVALYOV, 2015; BRACHNER; HVATTUM, 2017), principalmente as relativas ao número de pousos e decolagens (HALSKAU, 2014).

Na logística aplicada a operações humanitárias e desastres naturais, o objetivo é fornecer assistência de serviços e de bens (comida, água, ajuda médica, abrigo e suprimentos) a áreas afetadas por emergências de grande escala (BEAMON; BALCIK, 2008). Exemplos de calamidades, nas quais é importante um sistema coordenado de socorro, são encontradas nas situações de terremoto, furacão, enchentes, erupções vulcânicas, etc. Por isso, uma diferença fundamental entre o roteamento de helicópteros para usos comerciais e para funções humanitárias é que este último tem que ser capaz de responder com múltiplas intervenções tão rápidas quanto possível e dentro um janela de tempo estreita (VAN WASSENHOVE, 2006).

A seguir, uma revisão bibliográfica dos principais trabalhados relacionados ao HRP é apresentada.

2.3.1 Revisão Bibliográfica

Galvão e Guimarães (1990) desenvolveram e analisaram o impacto de uma heurística para o transporte de funcionários em plataformas marítimas na bacia de Campos, localizada no Rio de Janeiro, Brasil. O problema consiste numa frota heterogênea de aeronaves com o propósito de deslocar funcionários para até 60 bases em alto-mar, as quais estão numa distância entre 100 e 300 quilômetros da costa. O objetivo é reduzir o custo operacional ao atender as demandas de transporte. Há restrições de capacidade de passageiros, quantidade de combustível e peso da aeronave. Ao desenvolver a heurística, os autores mantiveram o esquema

de transporte implantado na empresa alvo do estudo. O modelo em uso consistia de 6 conjuntos de voos planejados por dia. O procedimento de elaboração das rotas foi realizado por um algoritmo construtivo, o qual adiciona os pontos de *pickups* e *deliveries* nas rotas tentando minimizar a distância total percorrida pela frota. Nos testes computacionais efetuados em 65 instâncias, o método proposto conseguiu resultados melhores ou iguais em 72% das instâncias. As comparações foram feitas com as rotas geradas manualmente no dia a dia pela equipe de planejamento de voo. Em geral, uma economia diária de US$ 4.500,00 foi obtida pela proposta dos autores.

Fiala Timlin e Pulleyblank (1992) desenvolveram uma heurística para o HRP aplicada ao transporte de pessoas em plataformas na Nigéria. As restrições do problema consistem nas precedências de *pickups* e *deliveries*, na capacidade de passageiros na aeronave, rotas começando e terminando no aeroporto. Além disso, as plataformas possuem prioridades de atendimento, por isso, algumas precisam ser atendidas antes de outras. Por este motivo, a heurística proposta subdivide o problema em subproblemas, cada um correspondendo a uma classe de prioridade. O subproblema k consiste em encontrar uma sequência de plataformas com distância mínima e factível (em relação à capacidade da aeronave e a precedências de *pickup* e *delivery*) para cada par de plataformas p_a e p_b que pertence à classe de prioridade k. O procedimento usado para resolver cada subproblema é baseado num algoritmo para resolver o TSP, chamado de inserção mais cara. Em cada iteração deste procedimento, a plataforma p inserida na sub-rota é aquela que possui o maior incremento no custo da rota. Para obter uma solução completa, os autores geraram um grafo ponderado a partir das melhores soluções encontradas para cada subproblema. Usando este grafo, um algoritmo de menor caminho é usado para gerar a solução final. Os testes foram realizados para a empresa Mobil na Nigéria e dois benefícios foram relatados pela companhia: o atendimento de todas as restrições operacionais e a redução de até 15% dos custos de transporte.

Sierksma e Tijssen (1998) desenvolveram um modelo de Programação Linear (PL) com GC e heurísticas de aprimoramento para o problema de atendimento de requisições de transporte para plataformas em alto-mar. Para diminuir a complexidade do problema, os autores executaram um procedimento de agrupamento de plataformas com o objetivo de diminuir o espaço de solução no modelo PL. Os resultados obtidos pelo PL são fracionários, enquanto a solução do problema deve ser inteira. Por isso, é aplicado um procedimento de arredondamento para obter a solução final inteira, o qual gera uma solução factível, mas não necessariamente ótima. Os autores relatam esta questão, mas explicam que os resultados finais são bons suficientes para propósitos práticos. Por fim, no modelo proposto, a solução inteira é melhorada usando as heurísticas 1-opt e 2-opt, as quais executam trocas de plataformas entre aeronaves. As restrições do problema são o número de aeronaves, quantidade de combustível e capacidade de passageiros. Além disso, uma mesma plataforma pode ser visitada por mais de uma aeronave. O objetivo é reduzir a distância viajada pela frota. Os experimentos computacionais foram realizados em 11 instâncias contendo até 51 plataformas e diferentes números de re-

quisições. O método proposto foi comparado com as heurísticas CW e SWP. As comparações realizadas entre os métodos mostram que as melhores soluções são as obtidas pela proposta dos autores, apesar do maior tempo de execução.

Barbarosoğlu, Özdamar e Cevik (2002) definiram uma metodologia multicritério para o planejamento de operações de resgate usando helicópteros, na qual as restrições do modelo são divididas em dois níveis: superior e inferior. No nível superior, é executado um modelo de Programação Inteira (PI) que realiza o transporte de aeronaves e tripulação para uma base B próxima ao local do desastre. A função objetivo é minimizar o número de aeronaves utilizadas. Esta parte do modelo opera com decisões táticas como a composição da frota de helicópteros, a designação da tripulação e o número de voos que cada aeronave deve realizar. O nível inferior é um modelo de PLIM que designa as rotas das aeronaves estacionadas em B a cada local de emergência. Para fazer isso, o nível operacional utiliza as informações obtidas pelo modelo tático. Neste sentido, no nível inferior são atendidas as restrições operacionais como requisições de atendimento e de reabastecimento dos helicópteros. A função objetivo do nível operacional é minimizar o tempo total de operação. Dado que a metodologia proposta pelos autores divide o problema em duas partes, é possível que a solução encontrada no nível tático não seja factível para as restrições presentes no nível operacional. Assim, os autores propuseram uma heurística *top-down* que coordena os níveis táticos e operacionais. Os autores testaram a metodologia proposta em dados da Força Armada Turca.

Hernádvölgyi (2004) analisou diversos métodos para o cálculo de limites inferiores para problemas de otimização. Dentre estes, um caso particular foi feito para o HRP. Neste modelo, o objetivo é criar uma rota que atenda todas as requisições de transporte de funcionários em plataformas marítimas. A proposta foi modelada como um *Sequential Ordering Problem* (SOP) cujo objetivo é encontrar um caminho hamiltoniano de peso mínimo num grafo.

Armstrong-Crews e Mock (2005) apresentaram um AG para a roteirização de helicópteros no Alasca. O problema consiste na criação de uma rota que visite distintos locais de interesse do Departamento de Pesca e Jogos do Alasca, o qual é responsável por diversas estatísticas relacionadas ao meio ambiente. Dado que é usada somente uma aeronave, há diferentes pontos de reabastecimento para helicópteros na região coberta pelos voos. A heurística dos autores define o genoma do indivíduo como uma sequência de locais de visita. A reprodução é feita pela criação de indivíduos para a nova população a partir de trechos aleatórios da população atual. Caso o novo indivíduo seja inviável por falta de combustível para completar a rota, um algoritmo de fixação é executado, o qual insere locais de reabastecimento no genoma do indivíduo. Os testes foram executados e comparados com a rotas geradas pelo Departamento de Pesca e Jogos do Alasca.

Moreno et al. (2005) propuseram dois PLIM e uma heurística para resolver o problema de planejamento de voo no atendimento de requisições de transporte em plataformas *offshore* na bacia de Campos, localizada no Rio de Janeiro, Brasil. O problema descrito consiste numa

frota heterogênea de helicópteros, duas bases (aeroportos), plataformas marítimas e um conjunto de requisições de transporte com horário de partida pré-fixado. O objetivo do problema é construir um conjunto de rotas nas quais cada voo comece e termine numa base, a capacidade do helicóptero não seja excedida na sua rota e que um tempo mínimo de intervalo entre os voos seja respeitado para cada helicóptero. Além disso, há restrições quanto ao número de pousos e decolagens em cada plataforma e deve existir para cada piloto em operação uma pausa para seu almoço. A função objetivo é minimizar o custo das rotas. O primeiro PLIM é baseado num modelo de rede de fluxo multiproduto, tendo um número polinomial de restrições e variáveis. O segundo modelo tem número exponencial de variáveis e é interpretado como uma decomposição do primeiro. Segundo os autores, ambos modelos são difíceis de resolver, por isso, uma heurística de duas fases foi proposta para resolver o problema. A primeira fase, chamada de construtiva, um conjunto grande de voos é gerado. Na segunda fase da heurística, chamada de montagem, os voos são selecionados e designados a cada helicóptero. Os testes computacionais foram executados em instâncias reais, com variações no número de demanda. Os autores relataram que a heurística proposta foi capaz de gerar soluções melhores que as geradas manualmente e, para todas as instâncias, a média de redução no custo e no tempo foram de 15% e 8%, respectivamente.

Rosero e Torres (2006) propuseram um ACO hibridizado com uma heurística de inserção para o transporte de passageiros por helicópteros. A estratégia utilizada é subdividir o problema em duas partes. Na primeira, um ACO é utilizado para sequenciar as requisições de atendimento e, na segunda parte, uma heurística de inserção gulosa é utilizada para colocar as posições de início e fim de um passageiro na rota final. Duas variantes do ACO, chamadas ACO1 e ACO2, foram testadas. A diferença entre elas está na regra utilizada para escolher a próxima requisição que deve ser incluída no sequenciamento do ACO. Os testes foram realizados em 40 instâncias e os resultados comparados com o AG de Torres (2006). Em geral, o ACO1 conseguiu resultados iguais ou melhores ao AG, enquanto o ACO2 obteve soluções melhores somente para algumas instâncias.

Romero, Sheremetov e Soriano (2007) aplicaram a meta-heurística AG e heurísticas de otimização ao PDVRP utilizando helicópteros. O método proposto é dividido em duas fases. Na primeira, é considerado um problema de planejamento. Dado um conjunto de requisições s, o objetivo desta etapa é encontrar a melhor permutação de s, desde que as regras de *pickup* e *delivery* do PDVRP sejam respeitadas. A resolução deste estágio é feita por uma heurística construtiva. Na segunda fase, um problema de alocação de requisições aos helicópteros é resolvido pelo AG. Nesta etapa, o objetivo é encontrar, para cada helicóptero, o conjunto de requisições de tal forma que a distância total percorrida seja mínima. Os testes computacionais foram executados em instâncias reais fornecidas pela *Petróleos Mexicanos* (PEMEX), as quais incluíam 70 pessoas e 5 aeronaves. Segundo os autores, a proposta consegue obter soluções ótimas para certos parâmetros do AG.

Velasco et al. (2009) apresentaram um Algoritmo Memético (AM) para o problema de PDVRP com aplicação no transporte de funcionários de plataformas de petróleo. O modelo proposto contém uma aeronave, múltiplas viagens, capacidade de pessoas no helicóptero e tempo máximo para cada viagem. A proposta do AM é dividida em duas partes. Na primeira, uma solução inicial via algoritmos construtivos é instanciada. Na segunda parte, um algoritmo genético com busca local é aplicado à solução gerada na fase anterior. A população inicial do AM é gerada a partir de uma heurística construtiva que usa os mesmos procedimentos da busca local. O resultado desta fase é um conjunto de cromossomos. A constituição do genoma dos indivíduos é uma sequência de nós representando pontos de *pickup* ou *delivery*, sem a delimitação de rotas. A partir das soluções construtivas, na segunda fase, o AG gera uma nova população usando o operador de *crossover*. Este operador seleciona dois indivíduos aleatoriamente e reorganiza a sequência de *pickup* e *delivery*. Finalmente, desta nova população um elemento é escolhido aleatoriamente para a busca local. Este procedimento utiliza diversos movimentos que reinserem, permutam e reordenam as requisições de uma solução. A busca local continua em execução enquanto ocorrer melhoras na solução. Por fim, este novo indivíduo gerado pela busca local é reinserido na população corrente do AG. O processo de gerar novas populações via *crossover* e execuções da busca local, no AM, continua até que não haja melhora da solução ou um limite de iterações seja alcançado. Os autores testaram a proposta em instâncias reais e construídas aleatoriamente. Os resultados encontrados para o AM são 8% melhores, em média, em relação aos resultados obtidos pela heurística construtiva e busca local quando executadas isoladamente. Quando se compara com os resultados de instâncias reais, o AM conseguiu resultados de 0% a 78% melhores que os conhecidos.

Menezes et al. (2010) desenvolveram um sistema para gerar planos de voo para o transporte de funcionários em plataformas da Petrobras. O problema consiste em uma frota heterogênea, capacidade máxima de passageiros na aeronave e tabela de horário de voos disponíveis pré-fixada. Além disso, há restrições relativas à segurança, como o número máximo de pousos numa plataforma em determinado período, número máximo de voos para cada aeronave e tempo de manutenção dos helicópteros ao retornar para a base. Os autores apresentaram um modelo PLIM usando fluxo em redes para designar aeronaves, passageiros e plataformas a uma rota. O objetivo do modelo é reduzir o número de pousos e custo de operação. Dado que a proposta contém bilhões de variáveis, um procedimento de GC foi implementado. O primeiro passo deste método é desagregar as demandas com a mesma origem e o mesmo destino. O subproblema do GC é determinar um helicóptero h e um voo f com menor custo, respeitando a quantidade máxima de pousos, tempo máximo de voo e capacidade da aeronave h. Para a encontrar as colunas do GC é utilizada uma heurística que procura, de maneira independente, para cada helicóptero h e horário de partida t, um voo que sirva adequadamente um número fixo de plataformas. A ideia geral da proposta é decompor o problema em subunidades, as quais geram voos para cada aeronave disponível e, então, utilizando um modelo PI, agrupar as rotas criadas numa solução para o problema. Os experimentos computacionais foram realiza-

dos utilizando o planejamento manual de 354 dias de 2004. Segundo os autores, os resultados mostraram que a solução proposta gerou planos de voo com o mesmo número de passageiros que a solução manual. Além disso, ela foi capaz de reduzir em 18% o número de pousos, em 8% o tempo total de voo, em 14% o custo de operação, o que resultou numa economia de US$ 20 milhões de dólares.

Motta, Vieira e Soletti (2011) construíram uma meta-heurística AG para o atendimento de requisições de transporte, usando helicópteros, para plataformas em alto-mar. Os autores definiram o genoma dos indivíduos como a sequência de locais que são atendidos pela aeronave. Neste problema existe somente um helicóptero e todas as rotas começam e terminam no aeroporto. Na meta-heurística também estão modeladas as restrições de voo como peso carregado e consumo de combustível. A partir dos testes realizados, os autores observaram que a proposta não atende ainda as necessidades reais do setor petrolífero, pois as diversas execuções do AG não retornaram respostas ótimas conhecidas.

Ozdamar (2011) propôs um sistema de planejamento de logística aplicado ao roteamento de helicópteros para o atendimento de vítimas de desastres naturais. O sistema consiste em um modelo matemático e um Procedimento de Gerenciamento de Rota (PGR), uma ferramenta que auxilia no pós-processamento dos resultados obtidos pelo modelo matemático. A função objetivo é minimizar o tempo total da missão, o qual inclui o tempo de voo e o tempo de carregamento/descarregamento de pessoas e de itens nas aeronaves. As requisições representam o deslocamento de suprimentos e de pessoas acidentadas. Os helicópteros possuem restrições aéreas especiais: a) o peso máximo de carga depende da temperatura e da altitude do local; b) o consumo de combustível está relacionado ao peso bruto da aeronave, à altitude de voo e à temperatura externa. O modelo matemático proposto é um PLIM que utiliza a abordagem de fluxo dinâmico em rede. Os autores alegaram que este paradigma diminui a complexidade se comparado ao modelo de três índices. Entretanto, o resultado do modelo proposto não é um conjunto de rotas, mas o fluxo de pessoas e de materiais entre os nós da rede. Para criar um conjunto de rotas com a designação de cada helicóptero utilizado, a rotina de pós-processamento PGR é executada. Este procedimento é realizado por um operador manual, o qual tem a opção de definir parâmetros de tempo máximo de voo e quantidade máxima de aeronaves disponíveis. Além disso, as restrições relacionadas ao abastecimento não são consideradas no modelo PLIM e são garantidas pelo PGR. Por isso, em rotas que é necessária a adição de paradas para reabastecimento de aeronaves, a otimalidade da solução não é garantida. Os autores testaram a solução num conjunto de dados que representavam 10.414 pessoas feridas para serem recolhidas para até 60 centros médicos. O número de pessoas feridas por nó na rede é, em média, de 173. O total de carga externa a ser transportada é 1054,43 toneladas. Bons resultados foram obtidos em menos de 10 minutos com um *GAP* de 6% para a melhor solução.

Qian, Gribkovskaia e Halskau Sr (2011) analisaram o HRP para plataformas em alto-

mar, incluindo critérios de segurança para os passageiros. Os autores desenvolveram um modelo PI cuja função objetivo minimiza o número esperado de fatalidades. Usando a frequência de acidentes f, o número de fatalidades é calculado em função do número de horas de voo, de passageiros a bordo e da probabilidade de fatalidade de um passageiro em caso de acidente no pouso ou na decolagem. O modelo matemático não inclui critérios em relação a consumo de combustível, de peso ou de tempo de voo dos helicópteros. Nos testes computacionais, um conjunto aleatório de requisições é criado para ser atendido com uma frota homogênea de aeronaves. A partir dos resultados obtidos, os autores concluem que o número esperado de fatalidades é minimizado quando cada rota realiza o atendimento de uma única plataforma.

Qian et al. (2012) analisaram como aumentar a segurança ao transportar funcionários de plataformas marítimas por helicópteros. Considerando três políticas de roteamento, um modelo PLIM e uma meta-heurística BT foram propostos com o objetivo de reduzir o número esperado de fatalidades. As políticas propostas foram de voo direto, voo hamiltoniano e voo geral. Na política de voo direto, cada plataforma é servida por uma rota exclusiva, ou seja, o número de rotas é igual ao número de plataformas marítimas. Na política de voo hamiltoniano, cada plataforma é visitada exatamente uma vez. Na política de voo geral uma plataforma pode ser visitada duas vezes, uma como *pickup* e outra como *delivery*. A BT utilizada foi adaptada de Cordeau, Gendreau e Laporte (1997). Ela inicia com uma solução viável ou inviável e trabalha com uma função objetivo penalizada. Cada solução possui um conjunto de atributos que designa a plataforma visitada pela aeronave. A vizinhança de uma solução é definida como todas as soluções que podem ser geradas após remover uma plataforma de uma rota e incluí-la em outra. A diversificação nas soluções é alcançada ao penalizar os movimentos mais frequentes. Os resultados dos testes computacionais mostraram que o risco de transporte de passageiros pode ser reduzido com o aumento do tempo de viagem, mas que, em contrapartida, aumenta o risco de fatalidade do piloto. A análise comparativa entre políticas também evidenciou que a política de roteamento direto é ótima para os passageiros, enquanto a política hamiltoniana é a melhor para os pilotos.

Van Urk, Mes e Hans (2013) desenvolveram um sistema de suporte para a unidade de polícia holandesa que roteia, antecipadamente, helicópteros em função da previsão de acidentes futuros. Segundo os autores, apesar de as ocorrências não serem conhecidas previamente, é muito importante que ocorra uma resposta em tempo adequado. Portanto, as aeronaves devem estar próximas do local de atendimento quando uma emergência acontece. Em função disso, é proposto no trabalho um modelo que é a combinação do VRP e do *Location Convering Proplem* (LCP), gerando uma nova classe de problema chamada *Anticipatory Emergency Vehicle Routing Problem* (AEVRP). Um PLIM é apresentado para o AEVRP, o qual faz o roteamento dos helicópteros baseado na previsão de incidentes num dado período e cujo objetivo é maximizar o número de acidentes em que os helicópteros fornecem assistência com sucesso. Entretanto, dada a impossibilidade da execução do modelo matemático para instâncias reais, um método usando heurística é apresentado. Neste caso, o problema é resolvido sequencialmente. Em cada

passo, somente um helicóptero precisa ser roteado, desde que as propriedades de local de início da rota, local de fim da rota e de tempo máximo de voo sejam mantidos fixos. O horário de partida é calculado a partir de uma fórmula desenvolvida pelos autores. Os testes foram executados utilizando dados de 2 anos da polícia holandesa. Os dados do primeiro ano foram usados para realizar as previsões e os dados do segundo para executar a simulação. De acordo com os resultados, a taxa de sucesso do novo método pode ser até noves vezes melhor que a série histórica disponível.

Özdamar et al. (2013) apresentaram um sistema para coordenar operações em que helicópteros são usados no atendimento de vítimas em desastres naturais. Os autores utilizaram e adaptaram a proposta de Özdamar e Demir (2012) para resolver o problema logístico. O resultado é um sistema hierárquico, em que no nível mais baixo é executado um PLIM, modelado como fluxo em redes, que otimiza o deslocamento de veículos e materiais nas localidades demandantes. No nível mais alto, está uma heurística que subdivide a rede de depósitos e de demandas recursivamente, agrupando nós em grupos e os representando como nós individuais na rede. O processo de agrupamento de heurística continua até que o tamanho dos grupos permita a geração de uma solução ótima pelo modelo PLIM. De modo a garantir a viabilidade das rotas produzidas, um pós-processamento é executado para inserir estações de reabastecimento nas rotas quando necessárias. Os testes foram executados em dois cenários. O primeiro relacionado a um potencial terremoto em Istambul e o segundo uma aplicação de regaste via helicópteros na enchente causada pelo furacão Katrina. Os resultados mostraram que soluções quase ótimas foram obtidas dentro de um intervalo de tempo razoável para o atendimento das demandas.

Halskau (2014) apresentou diferentes políticas de roteamento de helicópteros para plataformas marítimas utilizando intermediários (*hubs*) em alto-mar com o objetivo de reduzir o número de fatalidades. Neste modelo, todas as aeronaves partem de aeroporto, realizam operações de *pickup* e de *delivery* e retornam à base em terra. As políticas estudadas pelo autor são: i) a utilização do aeroporto com base intermediária; ii) uma plataforma como intermediária; iii) duas ou mais plataformas como intermediárias. Quando se utiliza uma localidade como intermediária, todo o fluxo de pessoas que saia ou entre no aeroporto deve passar por este *hub*. Além disso, todo atendimento de *pickup* ou de *delivery* numa plataforma deve ser atendido por um voo entre o intermediário e a plataforma. Assim, no caso de um voo partindo do aeroporto com d pessoas para *delivery*, o intermediário i irá receber d pessoas e fará rotas para a entrega de cada passageiro estacionado em i, sempre visitando uma plataforma e voltando a i. Em simultâneo, fará o *pickup* de cada passageiro que será atendido pela aeronave. Após todos os *pickups* e *deliveries*, o intermediário i terá p pessoas que serão transportadas ao aeroporto. No caso de k intermediários, k aeronaves realizam o atendimento das requisições utilizando k plataformas intermediárias distintas de tal forma que cada plataforma seja atendida por somente uma aeronave. O autor provou os cenários ótimos quando há somente um intermediário e um algoritmo polinomial para seleção deste local. Além disso, apresentou

também um modelo de Programação Inteira Binária (PIB) que cria rotas com m intermediários quando há m aeronaves.

Gribkovskaia, Halskau e Kovalyov (2015) analisaram a complexidade computacional e derivaram algoritmos para o problema de roteirização de helicópteros para o caso de frota unitária e frota múltipla. As soluções propostas incluem um modelo de Programação Quadrática Binária (PQB) e dois algoritmos de PD. Um algoritmo polinomial foi apresentado para o caso do problema com somente uma aeronave. As restrições do problema são o número de helicópteros e a capacidade máxima de passageiros da aeronave. A função objetivo é uma combinação linear da quantidade de passageiros que participam de pousos e decolagens numa rota. Os experimentos computacionais com os métodos apresentados são capazes de resolver, num prazo de até 5 minutos, instâncias com até 20 locais de atendimento e até 6 aeronaves.

Andreeva-Mori, Kobayashi e Shindo (2015) propuseram um modelo de roteamento de helicóptero para o atendimento de vítimas em desastres naturais usando a meta-heurística PSO hibridizada com uma Busca Local Gulosa (BLG). A abordagem dos autores exige poucos parâmetros na meta-heurística e permite diversas restrições de desempenho para aeronaves e rotas. O modelo prevê uma frota heterogênea, múltiplas viagens, consumo de combustível e tempo máximo para cada missão (local de uma requisição) executada pelos helicópteros. A BLG é utilizada para predefinir as posições iniciais das partículas no PSO. Nesta meta-heurística, os indivíduos são modelados como matrizes que designam a missão executada por cada aeronave. As linhas representam as aeronaves enquanto as colunas representam as missões. Dependendo do valor presente na linha i da coluna j, determina-se que o helicóptero i executa a missão j. Para que a solução seja factível, adiciona-se uma aeronave fantasma nos indivíduos do PSO e missões que não podem ser atendidas pela frota são mapeadas para esta aeronave fantasma. Assim, a proposta dos autores permite que certas requisições não sejam atendidas. Os testes foram executados num conjunto de dados elaborado pelos autores a partir de informações disponíveis do resgate para o terremoto e *tsunami* ocorrido no Japão em 2011. Os dados englobam 160 missões, as quais possuem de 1 a 100 indivíduos esperando por resgate. Há três modelos de helicópteros com diferentes tempos de voo e de capacidades. O procedimento proposto foi comparado com um PLIM e um AG. Os resultados mostraram que a meta-heurística PSO hibridizada com BLG obtém bons resultados quando comparada com o *upper bound* do PLIM relaxado. Entretanto, quando se permite tempo de execução maior para o modelo PLIM, este alcança resultados superiores. O AG não foi capaz de encontrar uma solução viável para o problema.

Rosa et al. (2016) propuseram um modelo PLIM para o roteamento de helicópteros capacitados com várias restrições específicas e uma função objetivo realista. A proposta inclui a modelagem do consumo de combustível, frota heterogênea, tempo máximo de rota e capacidade de peso para os passageiros nas aeronaves. A função objetivo minimiza o custo operacional, definido como uma soma ponderada do número de helicópteros utilizados e a

distância total percorrida. O PLIM somente encontrou soluções ótimas para instâncias pequenas e apresentou *GAP* ou não encontrou nenhuma resposta para as instâncias maiores. Por isso, os autores aplicaram a meta-heurística *Clustering Search* (CS) para resolver o problema. Os testes computacionais foram executados com instâncias reais com até 1000 requisições e 100 aeronaves. O CS foi capaz de encontrar soluções boas e estáveis dentro de um intervalo de tempo adequado.

Abbasi-Pooya e Kashan (2017) propuseram dois modelos matemáticos e uma meta-heurística híbrida para o HRP com frota unitária, múltiplas viagens e função objetivo minimizando o tempo total de voo. Os modelos matemáticos desenvolvidos são definidos em função da característica que é representada no grafo do problema. No primeiro modelo, chamado pelos autores de *Jacket-based formulation* (JBF), os nós do grafo representam as plataformas em alto-mar. No segundo modelo, chamado pelos autores de *Passenger-based formulation* (PBF), os nós representam os passageiros do problema. Usando os modelos JBF e PBF, os autores definiram dois PLIM. Um conjunto de instâncias foi gerado e avaliou-se o tempo e a solução obtida para cada modelo. Em geral, o PLIM do JBF obteve melhores resultados se comparado com o PLIM do PBF, pois normalmente há menos plataformas que passageiros, o que implica num menor número de variáveis e restrições para o JBF. Entretanto, os modelos matemáticos resolveram somente instâncias pequenas e médias. Por isso, os autores propuseram também um meta-heurística híbrida chamada $(1 + \lambda)$-HGES, a qual utiliza a *Grouping Evolution Strategy* (GES) hibridizada com uma BLG. A solução inicial da meta-heurística utiliza uma abordagem de hierárquica, em que primeiro se cria uma única rota que atenda todas as requisições e, posteriormente, utiliza-se um procedimento para dividir a rota inicial em diversas rotas. Os resultados da meta-heurística $(1 + \lambda)$-HGES foram comparados com os resultados da meta-heurística PSO e da meta-heurística GES sem hibridização. Os autores mostraram que a meta-heurística proposta possui resultados semelhantes ao PSO e ao GES para instâncias de tamanho médio. Entretanto, para instâncias grandes a meta-heurística supera os resultados do PSO e do GES.

Brachner e Hvattum (2017) propuseram um método integrado para o planejamento de transporte de funcionários marítimos por helicópteros e a organização de socorro a acidentes em alto-mar. A finalidade é garantir que as unidades de resgate (UR) estejam localizadas de tal forma que, se algum acidente acontece, todos os passageiros da aeronave sejam resgatados dentro de um prazo máximo. Os autores apresentaram um modelo PI que cria um conjunto de rotas para as requisições de funcionários em plataformas marítimas e, também, designa os recursos a certos pontos da área abrangida pelos voos. A função objetivo é reduzir a distância percorrida pelas aeronaves. Não existem restrições para o peso da aeronave, consumo de combustível ou capacidade de passageiros. O modelo proposto, chamado pelos autores de *Combined Routing and Covering Problem* (CRCP), não consegue resolver instâncias reais. Então, é proposto um método de 3-passos. Primeiro, modifica-se o CRCP para um problema LP com a função objetivo minimizando o número de unidades de socorro que precisam ainda ser

alocadas. No segundo passo, o CRCP é resolvido a partir da solução da primeira fase, mantendo fixas as unidades de resgate utilizadas e os locais utilizadas por estas. Finalmente, no terceiro passo, o CRCP é resolvido com o resultado da segunda fase e sem valores fixos para as unidades de resgate. Esta última fase é uma tentativa de atingir o ótimo do problema. Os testes, realizados em instâncias reais e criadas, mostraram uma mútua interdependência entre a parte logística de transporte e a organização de socorro, abrindo a oportunidade de economia de recursos.

Fernández-Cuesta et al. (2017) apresentaram um modelo PLIM que cria uma frota ótima de helicópteros, designa bases de operação em terra e seleciona *hubs* de abastecimento em alto-mar. Para fazer isso, o PLIM inclui requisições de transporte de passageiros para plataformas marítimas com o objetivo de criar a alocação ótima dos recursos. Os autores designam o problema assim definido como *Base Location and Fleet Composition Problem* (BLFCP). O modelo PLIM é resolvido utilizando GC. Os voos, representados pelas colunas, são gerados *a priori*. Nesta proposta, o GC tem dois objetivos. O primeiro consiste em criar todos os voos que são necessários para rotear os helicópteros. O segundo é encontrar todas as combinações possíveis de transporte de empregados que podem conflitar com a capacidade de carga dos helicópteros. As colunas são geradas usando PD baseada no algoritmo de rotulação. Para reduzir o número de colunas, duas heurísticas foram usadas para diminuir o número de voos testados. Na primeira, o espaço é reduzido ao agrupar plataformas que sejam próximas geograficamente. Na segunda heurística, a lista de vizinhos de uma plataforma n é reduzida a um valor que seja menor ou igual ao máximo entre o número de plataformas próximas de n multiplicado por um fator α e o número de plataformas no maior agrupamento da rede. Os experimentos computacionais foram realizados em dados dos campos de pré-sal Brasileiro e na expansão Norueguesa no Mar de Barents no Ártico. O BLFCP proposto foi capaz de resolver os casos de tamanhos mais relevantes, mas com a presença de *GAP* e tempo de execução alto. Segundo os autores, a qualidade das soluções está entre 50% e 90% superior em relação às abordagens utilizadas na literatura.

Oh et al. (2018) propuseram um PLIM para o socorro de vítimas com diferentes níveis de urgência. Os autores propõem uma arquitetura para o gerenciamento de desastres naturais em três fases: coleta de informações, planejamento e execução do plano de evacuação. O modelo proposto opera nesta última etapa. Esse plano compreende a designação de rotas às aeronaves com diferentes capacidades, enquanto diversas condições complexas para a operação precisam ser atendidas. Em especial, veículos começam e terminam na mesma base, mas podem estacionar em outras localidades se necessário. Além disso, é possível reabastecer a aeronave na base se for necessário. O objetivo é salvar o maior número de vítimas e preveni-las de um evento adicional no futuro. Dado a dificuldade de resolver PLIM para instâncias grandes, eles propuseram um algoritmo baseado em multiagentes para obter soluções em tempo aceitável. Nessa abordagem, veículos e vítimas são considerados como agentes e ele cooperam para construir uma solução subótima. Os testes computacionais foram realizados usando

dados reais.

Kashan, Abbasi-Pooya e Karimiyan (2019) desenvolveram um modelo PLIM e uma meta-heurística *League Championship Algorithm* (LCA) para uma rede de transporte em que os nós representam plataformas *offshore*. O problema consiste na designação das requisições de transporte a um conjunto pré-determinado de *tours* com o objetivo de minimizar o tempo total de viagem. As restrições do problema consistem em capacidade e peso dos passageiros, tempo máximo de voo e uma aeronave disponível para a execução dos *tours*. A meta-heurística LCA é um algoritmo baseado em população para problemas restritos e não restritos. O modelo PLIM foi aplicado a instâncias pequenas e médias, enquanto o LCA foi utilizado em instâncias grandes.

Machado et al. (2019) propuseram um PLIM que é uma extensão do modelo proposto por Rosa et al. (2016) com frota heterogênea, capacidade de peso, capacidade de ocupação das aeronaves, restrições que controlam o uso de combustível, janelas de tempo e tempo máximo de voo para o transporte de equipes em plataformas marítimas. Além disso, esse modelo incorpora a possibilidade de múltiplas viagens por helicóptero e um novo tipo de restrição, a qual impede a utilização concorrente de helipontos em plataformas. Essa nova característica visa aumentar a segurança das rotas geradas e colocar em evidência uma funcionalidade ainda não modelada pelos modelos da literatura. Os testes computacionais foram executados em instâncias construídas a partir de dados reais.

Adarang et al. (2020) analisaram o problema de roteirização e alocação de recursos em situações de incerteza para atendimento médico em desastres. O objetivo consiste em minimizar o tempo de resgate e o custo total de alocação e roteirização dos veículos. A frota contém diversos modelos de equipamentos, incluindo helicópteros. Um novo PLIM foi proposto para formular o problema de alocação e roteirização de recursos e duas meta-heurísticas foram desenvolvidas para resolver as instâncias de médio e grande porte. O CPLEX foi usado para resolver exatamente as instâncias pequenas. Segundo os autores, os resultados da proposta são importantes porque ajudam os responsáveis por operação de resgate a planejá-las e gerenciá-las em condições de incerteza.

Caballero-Morales e Martinez-Flores (2020) elaboraram um modelo de roteirização que integra taxas não determinísticas de falha na construção de rotas para atendimento de requisições de transporte em plataformas de petróleo em alto-mar. O modelo PLIM e a meta-heurística AG foram apresentados para resolver o problema. As instâncias de teste foram geradas usando uma abordagem bayesiana e os resultados mostram que a confiabilidade das rotas diminui quando a frequência das falhas aumenta. Como consequência, o tempo para falha e a distância para falha das rotas diminuem. Isso implica que para aumentar a segurança é necessário o aumento de helicópteros disponíveis.

Xavier et al. (2020) desenvolveram uma estratégia de roteamento de helicópteros na área de operações humanitárias que foi aplicado nas enchentes e deslizamentos que ocorre-

ram na região montanhosa do Rio de Janeiro em 2011. Os autores justificam a utilização do transporte por helicópteros devido às estradas e pontes inoperantes durante eventos climáticos extremos. A proposta contém cinco estágios de planejamento para o atendimento das requisições. Inicialmente, define-se a rede de transporte. Em seguida, estima-se as requisições baseando-se nas demandas. Posteriormente, a rede de transporte é reelaborada baseada na capacidade e limitações impostas pelas condições climáticas. Com esses dados, o número de helicópteros necessários para a operação é estimado. A partir desses dados, o problema é modelado como um CVRP com janelas de tempo através de um modelo PLIM. A função objetivo é minimizar a quantidade total de voo das aeronaves. O modelo proposto é resolvido usando o algoritmo CW com movimentos 2-OPT.

Nafstad et al. (2021) apresentaram o problema de programar o atendimento de requisições de transporte marítimo em instalações *offshore*. O objetivo é determinar um conjunto de rotas para uma frota heterogênea de helicópteros operando a partir de vários depósitos e múltipla viagens. Dois modelos, designados como compactos e extensos são apresentados. Esse último, aplica uma geração de restrições para permitir a convergência rápida para uma solução subótima. Nessa proposta é apresentada, também, uma restrição que impede o desembarque, simultâneo, em plataformas de aeronaves diferentes. Essa restrição é operada através da geração atrasada de restrições no modelo estendido. Ou seja, o modelo matemático é resolvido sem nenhuma restrição em relação ao pouso simultâneo. Caso um conflito ocorra em duas rotas distintas atendendo a mesma plataforma, então novas restrições são adicionadas que excluam essa possibilidade. Nesse caso, o método utilizado para resolver o modelo é o BB. Segundo os autores, a utilização do BB com essa estratégia permite desempenho superior aos pacotes comerciais.

A Tabela 1 mostra um resumo dos artigos analisados nesta seção.

Tabela 1 – Lista de Trabalhos.

Trabalho	Referência	Ano	Título
R_{01}	Galvão e Guimarães (1990)	(1990)	The control of helicopter operations in the Brazilian oil industry: Issues in the design and implementation of a computerized system (1990)
R_{02}	Fiala Timlin e Pulleyblank (1992)	(1992)	Precedence Constrained Routing and Helicopter Scheduling: Heuristic Design (1992)
R_{03}	Sierksma e Tijssen (1998)	(1998)	Routing helicopters for crew exchanges on off-shore locations (1998)
R_{04} [tático]	Barbarosoğlu, Özdamar e Cevik (2002)	(2002)	An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations (2002)
R_{05} [operacional]	Barbarosoğlu, Özdamar e Cevik (2002)	(2002)	An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations (2002)
R_{06}	Hernádvölgyi (2004)	(2004)	Automatically generated lower bounds for search (2004)
R_{07}	Armstrong-Crews e Mock (2005)	(2005)	Helicopter Routing for Maintaining Remote Sites in Alaska using a Genetic Algorithm. (2005)
R_{08}	Moreno et al. (2005)	(2005)	Planning Offshore Helicopter Flights on the Campos Basin (2005)
R_{09}	Rosero e Torres (2006)	(2006)	Ant colony based on a heuristic insertion for a family of helicopter routing problems (2006)

R_{10}	Romero, Sheremetov e Soriano (2007)	(2007)	A genetic algorithm for the pickup and delivery problem: An application to the helicopter offshore transportation (2007)
R_{11}	Velasco et al. (2009)	(2009)	A memetic algorithm for a pick-up and delivery problem by helicopter (2009)
R_{12}	Menezes et al. (2010)	(2010)	Optimizing helicopter transport of oil rig crews at Petrobras (2010)
R_{13}	Motta, Vieira e Soletti (2011)	(2011)	Optimal routing offshore helicopter using Genetic Algorithm (2011)
R_{14}	Ozdamar (2011)	(2011)	Planning helicopter logistics in disaster relief (2011)
R_{15}	Qian, Gribkovskaia e Halskau Sr (2011)	(2011)	Helicopter routing in the Norwegian oil industry: Including safety concerns for passenger transport (2011)
R_{16}	Qian et al. (2012)	(2012)	Passenger and pilot risk minimization in offshore helicopter transportation (2012)
R_{17}	Van Urk, Mes e Hans (2013)	(2013)	Anticipatory routing of police helicopters (2013)
R_{18}	Özdamar et al. (2013)	(2013)	Hierarchical optimization for helicopter mission planning in large-scale emergencies (2013)
R_{19}	Halskau (2014)	(2014)	Offshore helicopter routing in a hub and spoke fashion: Minimizing expected number of fatalities (2014)
R_{20}	Gribkovskaia, Halskau e Kovalyov (2015)	(2015)	Minimizing takeoff and landing risk in helicopter pickup and delivery operations (2015)
R_{21}	Andreeva-Mori, Kobayashi e Shindo (2015)	(2015)	Particle Swarm Optimization/Greedy-Search Algorithm for Helicopter Mission Assignment in Disaster Relief (2015)
R_{22}	Rosa et al. (2016)	(2016)	A mathematical model and a Clustering Search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas (2016)
R_{23}	Abbasi-Pooya e Kashan (2017)	(2017)	New mathematical models and a hybrid Grouping Evolution Strategy algorithm for optimal helicopter routing and crew pickup and delivery (2017)
R_{24}	Brachner e Hvattum (2017)	(2017)	Combined emergency preparedness and operations for safe personnel transport to offshore locations (2017)
R_{25}	Fernández-Cuesta et al. (2017)	(2017)	Base location and helicopter fleet composition in the oil industry (2017)
R_{26}	Oh et al. (2018)	(2018)	Cooperative multiple agent-based algorithm for evacuation planning for victims with different urgencies (2018)
R_{27}	Kashan, Abbasi-Pooya e Karimiyan (2019)	(2019)	A rig-based formulation and a league championship algorithm for helicopter routing in offshore transportation (2019)
R_{28}	Machado et al. (2019)	(2019)	A MILP Model and a GRASP Algorithm for the Helicopter Routing Problem with Multi-Trips and Time Windows (2019)
R_{29}	Adarang et al. (2020)	(2020)	A robust bi-objective location-routing model for providing emergency medical services (2020)
R_{30}	Caballero-Morales e Martinez-Flores (2020)	(2020)	Helicopter routing model with non-deterministic failure rate for evacuation of multiple oil platforms (2020)
R_{31}	Xavier et al. (2020)	(2020)	Planning the use of helicopters in distribution of supplies in response operations of natural disasters (2020)
R_{32}	Nafstad et al. (2021)	(2021)	An exact solution method for a rich helicopter flight scheduling problem arising in offshore oil and gas logistics (2021)

2.3.2 Detalhamento das Propostas

Nesta seção é apresentada uma avaliação das características das propostas de roteamento de helicópteros apresentadas nas seções anteriores. Para isso, dividiu-se o estudo das

propriedades em quatro grupos:

- O primeiro grupo (G_1) corresponde ao método proposto para resolver o HRP. Este grupo analisa a metodologia utilizada (modelo matemático, heurística, meta-heurística ou proposta híbrida) e objetivo do modelo (minimizar o custo, o tempo, a distância, o número de fatalidades, o número de acidentes ou o número de pousos/decolagens);
- O segundo grupo (G_2) corresponde às restrições e propriedades que o trabalho proposto impõe ao atendimento dos passageiros;
- O terceiro grupo (G_3) reúne as restrições e propriedades relacionadas aos helicópteros;
- O quarto grupo (G_4) corresponde a propriedades que não se encaixam nos grupos anteriores.

Nos próximos parágrafos são apresentados estes grupos e a análise de cada trabalho da Tabela 1. Com o objetivo de classificar e identificar corretamente as propriedades examinadas, utilizou-se um conjunto de símbolos para designar diferentes valores para cada característica. A Tabela 2 mostra estes símbolos com os seus significados.

Tabela 2 – Símbolos e significados utilizados para a análise das propriedades dos trabalhos.

Símbolo	Significado
	Frota com um único veículo
	Frota com dois ou mais veículos
	Frota com um único tipo de veículo
	Frota com dois ou mais tipos de veículos
	A rota inicia/termina na base (aeroporto)
	A rota inicia/termina em qualquer local da rede de transporte
	Presença da propriedade no trabalho
	Ausência da propriedade no trabalho
	Não é claro se a propriedade é implementada ou não no trabalho
	Aplicação na área de transporte de funcionários em plataformas de petróleo
	Aplicação no socorro a locais de desastre humanitário
	Minimizar a distância percorrida
	Minimizar o tempo de viagem
	Minimizar o custo operacional
	Minimizar o número esperado de fatalidades
	Minimizar o número de acidentes
	Minimizar o número de pousos e decolagens

Fonte: Autor

A Tabela 3 apresenta o primeiro grupo de propriedades (G_1), o qual analisa os métodos de resolução implementados nos trabalhos da Seção 2.3. A coluna **Trabalho** designa as

propostas da Tabela 1. As colunas em **Método de Resolução** apresentam os métodos de resolução encontrados no referencial teórico. A coluna **Objetivo** designa qual é a função objetivo do trabalho. A coluna **Aplicação** indica se a proposta é dirigida ao transporte de funcionários em plataformas ou ao socorro humanitário em desastres naturais. Os símbolos usados nesta tabela estão detalhados na Tabela 2. A partir dos dados apresentados, as seguintes observações podem ser feitas para as propostas do HRP:

- Mais da metade das propostas (56%) apresentam um modelo PLIM;
- Menos da metade das propostas (34%) apresentam algum modelo distinto do PLIM;
- Mais da metade das propostas (53%) apresentam uma heurística/meta-heurística;
- Na função objetivo, 31,0% dos modelos buscam minimizar o custo da rota, 28,1% a distância das rotas, 25,2% o tempo das rota, 9,4% o número de fatalidades e 6,3% o número de acidentes ou pousos;
- As aplicações são, em 75% dos casos, na área de transporte marítimo, enquanto 25% aplicam na área de resgate humanitário.

A Tabela 4 apresenta as restrições referentes aos passageiros e restrições gerais (miscelânea) propostas nos trabalhos da Seção 2.3. A coluna **Trabalho** mostra as propostas da Tabela 1. As colunas em **Passageiros** designam as características referentes a passageiros modeladas nos trabalhos em análise. Os símbolos usados nesta tabela são apresentados na Tabela 2. A Tabela 5 explica o significado de cada característica da Tabela 4.

A partir dos dados apresentados na Tabela 4, as seguintes observações podem ser feitas para passageiros e restrições gerais do HRP:

- Menos de 35% consideram o peso dos passageiros nas rotas;
- Mais de 96% definem as localidades de origem e destino das requisições;
- Em torno de 21% das propostas apresentam janela de tempo;
- As restrições de segurança e tripulação estão presentes em 18,7% e 15,6% das propostas.

A Tabela 6 apresenta as restrições referentes aos helicópteros implementadas nos trabalhos da Seção 2.3. A coluna **Trabalho** referencia os trabalhos da Tabela 1. As colunas em **Helicóptero** apresentam as propriedades dos helicópteros. A Tabela 7 mostra o significado de cada coluna em **Helicóptero** da Tabela 6

A partir dos dados apresentados na Tabela 6, as seguintes observações podem ser feitas para as restrições do helicóptero HRP:

Tabela 3 – Grupo G_1: Propriedades dos métodos de resolução dos trabalhos analisados.

Trabalho	Método de resolução													Objetivo	Aplicação
	BB	BLG	BT	GC	GES	HS	PD	PI	PL	PLIM	PQB	PSO	SOP		
R_{01}	●	●	●	●	●	✓	●	●	●	●	●	●	●		
R_{02}	●	●	●	●	●	✓	●	●	●	●	●	●	●		
R_{03}	●	●	●	✓	●	✓	●	●	✓	●	●	●	●		
R_{04}	●	●	●	●	●	●	●	✓	●	●	●	●	●		
R_{05}	●	●	●	●	●	●	●	●	●	✓	●	●	●		
R_{06}	●	●	●	●	●	●	●	●	●	●	●	●	✓		
R_{07}	●	●	●	●	●	✓	●	●	●	●	●	●	●		
R_{08}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{09}	●	●	●	●	●	✓	●	●	●	●	●	●	●		
R_{10}	●	●	●	●	●	✓	●	●	●	●	●	●	●		
R_{11}	●	●	●	●	●	✓	●	●	●	●	●	●	●		
R_{12}	●	●	●	✓	●	●	●	●	●	✓	●	●	●		
R_{13}	●	●	●	●	●	✓	●	●	●	●	●	●	●		
R_{14}	●	●	●	●	●	●	●	●	●	✓	●	●	●		
R_{15}	●	●	●	●	●	●	●	✓	●	●	●	●	●		
R_{16}	●	●	✓	●	●	●	●	●	●	✓	●	●	●		
R_{17}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{18}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{19}	●	●	●	●	●	●	●	✓	●	●	●	●	●		
R_{20}	●	●	●	●	●	●	✓	●	●	●	✓	●	●		
R_{21}	●	●	●	●	●	●	●	●	●	●	●	✓	●		
R_{22}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{23}	●	✓	●	●	✓	●	●	●	●	✓	●	●	●		
R_{24}	●	●	●	●	●	●	●	●	✓	✓	●	●	●		
R_{25}	●	●	●	✓	●	●	●	●	●	✓	●	●	●		
R_{26}	●	●	●	●	●	●	●	●	●	✓	●	●	●		
R_{27}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{28}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{29}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{30}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{31}	●	●	●	●	●	✓	●	●	●	✓	●	●	●		
R_{32}	✓	●	●	●	●	●	●	●	●	✓	●	●	●		

Fonte: Autor

- Em geral, a minoria das propostas (37,5%) apresentam um modelo com frota homogênea;
- A maioria das propostas (75,0%) possuem frota com vários veículos;
- A maioria das propostas (93,7%) começam e terminam a rota na mesma localidade;
- Menos da metade dos modelos (48,1%) permitem múltiplas viagens;
- A maioria das propostas (62,5%) modelam algum limitante para o tempo de roteamento;
- Somente duas propostas (6,25%) definem uma restrição para o pouso seguro em plataformas.

De forma geral, o objetivo dos novos modelos HRP é incorporar características inéditas do mundo real. Nesse sentido, uma das principais preocupações atuais está na operação segura

Tabela 4 – Grupo G_2: Propriedades relacionadas aos passageiros. Grupo G_4: Propriedades miscelânea.

	Passageiros (G_2)					Miscelânea (G_4)			
Trabalho	Peso	Base Plataforma	Plataforma Base	Plataforma Plataforma	Janela Tempo	Segurança	Tripulação	*Split Deliveries*	*Split Pickups*
R_{01}	●	✓	✓	✓	●	●	●	●	●
R_{02}	●	✓	✓	✓	●	●	●	●	●
R_{03}	●	✓	✓	✓	●	●	●	●	●
R_{04}	●	●	●	●	●	●	✓	●	●
R_{05}	●	✓	✓	✓	●	●	✓	✓	✓
R_{06}	●	✓	✓	✓	●	●	●	●	●
R_{07}	●	✓	✓	✓	●	●	●	●	●
R_{08}	✓	✓	✓	✓	●	●	●	●	●
R_{09}	●	✓	✓	✓	●	●	●	●	●
R_{10}	✓	✓	✓	✓	●	●	●	●	●
R_{11}	●	✓	✓	✓	●	●	●	●	●
R_{12}	✓	✓	✓	✓	✓	●	✓	●	●
R_{13}	✓	✓	✓	✓	●	●	●	●	●
R_{14}	✓	✓	✓	✓	●	●	●	✓	✓
R_{15}	●	✓	✓	●	●	✓	●	●	●
R_{16}	●	✓	✓	✓	●	●	●	●	●
R_{17}	●	✓	✓	✓	✓	●	●	●	●
R_{18}	✓	✓	✓	✓	●	●	●	✓	✓
R_{19}	●	✓	✓	✓	●	✓	●	●	●
R_{20}	●	✓	✓	●	●	✓	●	●	●
R_{21}	●	●	✓	●	●	●	●	✓	✓
R_{22}	✓	✓	✓	✓	●	●	●	●	●
R_{23}	✓	✓	✓	✓	●	●	●	●	●
R_{24}	●	✓	✓	✓	●	✓	●	●	●
R_{25}	●	✓	✓	✓	●	●	●	●	●
R_{26}	●	✓	✓	✓	✓	●	●	●	●
R_{27}	✓	✓	✓	✓	●	●	✓	●	●
R_{28}	✓	✓	✓	✓	✓	✓	✓	●	●
R_{29}	✓	✓	✓	✓	✓	●	●	●	●
R_{30}	●	✓	✓	✓	●	●	●	●	●
R_{31}	●	✓	✓	✓	✓	●	●	●	●
R_{32}	●	✓	✓	✓	✓	✓	●	●	●

Fonte: Autor

Tabela 5 – Significado das propriedades dos passageiros e miscelânea.

Característica	Significado
1. Passageiros	
1.1 Peso	Designa se o peso do passageiro é considerado no cálculo de peso máximo da aeronave
1.2 Base Plataforma	Designa se a aeronave voa da base para a plataforma
1.3 Plataforma Plataforma	Designa se a aeronave voa de plataforma para plataforma
1.4 Plataforma Base	Designa se a aeronave voa da plataforma para a base
1.5 Janela Tempo	Designa se um passageiro possui janela de tempo no atendimento
2. Miscelânea	
2.1 Segurança	Designa se há restrições relativas à segurança
2.2 Tripulação	Designa se há restrições relativas à tripulação
2.3 *Split Deliveries*	Designa se é possível um passageiro ou uma carga ser *delivery* em mais de uma aeronave
2.4 *Split Pickups*	Designa se é possível um passageiro ou uma carga ser *pickup* em mais de uma aeronave

Fonte: Autor

de helicópteros em plataformas marítimas, pois este é o tipo modal que apresenta o maior risco de acidente (BRACHNER; HVATTUM, 2017). Segundo Brachner e Hvattum (2017), entre 2008 e 2010 houve o aumento de acidentes com fatalidades nos Estados Unidos e no Canadá e a maioria desses eventos ocorrem no pouso ou decolagem Gribkovskaia, Halskau e Kovalyov (2015).

Um aspecto ainda não presente nesses modelos, até a publicação do trabalho Machado et al. (2019) relacionado a esta tese, é a exigência das aeronaves executarem o atendimento em plataformas sem a concorrência de outros helicópteros. A incorporação dessa restrição, denominada Pouso Seguro na Tabela 6, adiciona mais um grau de segurança no planejamento das rotas. Além disso, a publicação de Machado et al. (2019) contribuiu com o avanço do estado da arte, visto que, recentemente, o trabalho de Nafstad et al. (2021) propôs um modelo incorporando o mesmo tipo de exigência.

Em relação às abordagens usadas nas heurísticas para resolução do HRP, 30 dos 32 trabalhos na Tabela 1 usam CFRS. Somente os trabalho de Velasco et al. (2009) e Abbasi-Pooya e Kashan (2017) empregam RFCS, mas com algumas restrições. Em Velasco et al. (2009), a criação de rotas via *splitting* pode gerar, conforme proposto pelo trabalho, soluções inviáveis, as quais são descartadas pelo método. O motivo é que se usa um algoritmo adequado para o CVRP, mas inadequado para problemas com restrições de *pickup* e *delivery*. Em Abbasi-Pooya e Kashan (2017), a abordagem é usada somente para permitir a criação de uma solução inicial. Após isso, todas as operações do algoritmo trabalham com o mesmo tipo de solução.

Tabela 6 – Grupo G_3: Propriedades referentes aos helicópteros.

Trabalho	Helicóptero (G_3)																	
	Frota	Tipo	Cap. Peso	Cap. Passageiros	Carga Externa	Rota Início	Rota Fim	T. Max. Rota	T. Pouso	T. Embarque	T. Desembarque	T. Decolagem	T. Manutenção	T. Abastecimento	Combustível	Viagens	Pousos/Decolagens	Pouso Seguro
R_{01}	✓	✓	✓	✓	●	✓	✓	●	●	●	●	●	●	●	✓	●	●	●
R_{02}	✓	?	●	✓	●	✓	✓	●	●	●	●	●	●	●	●	●	●	●
R_{03}	✓	?	●	✓	●	✓	✓	✓	●	●	●	●	●	●	✓	●	●	●
R_{04}	✓	✓	●	●	●	✓	✓	●	●	●	●	●	●	●	●	✓	●	●
R_{05}	✓	✓	✓	✓	●	✓	✓	✓	●	●	●	●	●	✓	✓	✓	●	●
R_{06}	✓	✓	●	●	●	✓	✓	●	●	●	●	●	●	●	●	●	●	●
R_{07}	✓	✓	●	●	●	✓	✓	●	●	●	●	●	●	✓	✓	●	●	●
R_{08}	✓	✓	✓	✓	●	✓	✓	✓	●	●	●	●	●	●	✓	✓	●	●
R_{09}	?	?	●	✓	●	?	?	●	●	●	●	●	●	●	●	?	●	●
R_{10}	✓	✓	●	●	●	✓	✓	●	●	●	●	●	●	●	●	●	●	●
R_{11}	✓	✓	●	✓	●	✓	✓	✓	●	●	●	●	●	●	✓	✓	●	●
R_{12}	✓	✓	✓	✓	●	✓	✓	✓	●	●	●	●	✓	●	✓	✓	✓	●
R_{13}	✓	✓	✓	●	●	✓	✓	●	●	●	●	●	●	●	✓	●	●	●
R_{14}	✓	✓	✓	✓	✓	✓	✓	✓	●	✓	✓	●	✓	✓	✓	●	●	●
R_{15}	✓	✓	●	✓	●	✓	✓	✓	●	●	●	●	●	●	●	✓	●	●
R_{16}	✓	✓	●	●	●	✓	✓	●	●	●	●	●	●	●	●	●	●	●
R_{17}	✓	✓	●	●	●	✓	✓	✓	●	●	●	✓	✓	✓	✓	✓	●	●
R_{18}	✓	✓	✓	✓	✓	✓	✓	✓	●	✓	✓	●	✓	✓	✓	●	●	●
R_{19}	✓	✓	●	✓	●	✓	✓	●	●	●	●	●	●	●	●	●	●	●
R_{20}	✓	✓	●	✓	●	✓	✓	●	●	●	●	●	●	●	●	●	●	●
R_{21}	✓	✓	●	✓	●	✓	✓	✓	●	✓	●	✓	●	●	✓	✓	●	●
R_{22}	✓	✓	✓	✓	●	✓	✓	✓	✓	●	●	✓	●	●	✓	●	●	●
R_{23}	✓	✓	✓	✓	●	✓	✓	✓	✓	✓	✓	✓	✓	●	●	✓	●	●
R_{24}	✓	✓	●	●	●	✓	✓	●	●	●	●	●	●	●	●	●	●	●
R_{25}	✓	✓	●	✓	●	✓	✓	✓	●	✓	●	●	✓	✓	✓	✓	✓	●
R_{26}	✓	✓	✓	✓	●	✓	✓	✓	●	✓	✓	●	●	✓	●	✓	●	●
R_{27}	✓	✓	✓	✓	●	✓	✓	✓	✓	✓	✓	✓	●	✓	✓	✓	●	●
R_{28}	✓	✓	✓	✓	●	✓	✓	✓	●	✓	✓	●	●	●	✓	✓	●	✓
R_{29}	✓	✓	●	✓	●	✓	✓	✓	●	●	●	●	●	●	●	✓	●	●
R_{30}	✓	✓	●	✓	●	✓	✓	✓	●	●	●	●	●	●	●	●	●	●
R_{31}	✓	✓	✓	●	●	✓	✓	✓	●	✓	✓	●	●	●	●	●	●	●
R_{32}	✓	✓	●	✓	●	✓	✓	✓	●	✓	✓	●	●	✓	✓	✓	●	✓

Fonte: Autor

Tabela 7 – Significado das propriedades dos helicópteros.

Característica	Significado
Frota	Designa a quantidade de veículos na frota
Tipo	Designa os tipos de veículos na frota
Cap. Peso	Designa se existem restrições relativas ao peso da aeronave
Cap. Passageiros	Designa se existem restrições relativas à quantidade de passageiros
Carga Externa	Designa se existe o transporte de carga externa à aeronave
Rota Início	Local de início da rota
Rota Fim	Local de fim da rota
T. Max. Rota	Designa se existe tempo máximo para execução de uma rota
T. Pouso	Designa se é contabilizado o tempo de pouso
T. Embarque	Designa se é contabilizado o tempo de embarque de passageiros
T. Desembarque	Designa se é contabilizado o tempo de desembarque de passageiros
T. Decolagem	Designa se é contabilizado o tempo de decolagem
T. Manutenção	Designa se há período de manutenção da aeronave
T. Abastecimento	Designa se é contabilizado o tempo de reabastecimento da aeronave
Combustível	Designa se há restrições referentes ao combustível da aeronave
Viagens	Designa se uma mesma aeronave pode fazer mais de uma rota
Pousos/Decolagens	Designa se há restrições referentes ao número de pousos ou decolagens para plataformas ou passageiros
Pouso Seguro	Designa se há restrições que impedem o pouso simultâneo de aeronaves em plataformas

Fonte: Autor

2.4 Considerações Finais

Nesse capítulo foram apresentados os conceitos gerais dos problemas VRP, CVRP e HRP. Para o CVRP, uma análise sucinta foi realizada, enquanto para o HRP, uma extensa análise de trabalhos correlatos foi executada. Além disso, mostrou-se que para o HRP existe um processo em curso de proposição de novas características, como a sugerida por este trabalho para garantir a segurança das rotas no atendimento de plataformas marítimas.

3 Definição dos Problemas

Neste capítulo são apresentados o modelo padrão do CVRP e os dois modelos para o HRP. O primeiro modelo para o HRP chama-se *Helicopter Routing Problem with Single Trip* (HRPS) e representa o roteamento de helicópteros com restrições de capacidade, tempo, peso e combustível. Nessa versão, o helicóptero só pode executar uma viagem. O segundo modelo, denominado *Helicopter Routing Problem with Multiple Trips* (HRPM), é uma extensão do primeiro com a inclusão da possibilidade de múltiplas viagens para cada aeronave, janelas de tempo e restrição de pouso simultâneo em plataformas marítimas.

3.1 O *Capacitated Vehicle Routing Problem* (CVRP)

O *Capacitated Vehicle Routing Problem* (CVRP) é descrito como um grafo $G = (N, E)$ em que $N = \{0, 1, \cdots, n\}$ é o conjunto de nós e $E \subset N \times N$ é o conjunto de arestas. Cada elemento $i \in N \setminus \{0\}$ representa um cliente com a demanda q_i, enquanto $i = 0 \in N$ designa o depósito central. A aresta $(i, j) \in E$ tem o peso $c_{i,j} > 0$ indicando o custo de translado entre os nós $i, j \in N$. O conjunto de veículos $K = \{1, 2, \cdots, k\}$ com capacidade máxima Q deve iniciar no depósito central $(i = 0)$ para servir cada cliente $i \in N \setminus \{0\}$ e retornar no final da rota ao depósito. O objetivo é encontrar um conjunto de rotas que atenda a todas as demandas com o menor custo.

Dada uma variável binária $x_{k,i,j}$ que designa se o veículo $k \in K$ atravessa o arco (i, j), então o Problema de Programação Inteira (PPI) para o CVRP pode ser descrito como (TOTH; VIGO, 2014):

$$\text{Minimizar} \quad \sum_{k}^{K} \sum_{i}^{N} \sum_{j}^{N} c_{i,j}\, x_{k,i,j} \tag{3.1}$$

Sujeito a:

$$\sum_{k=1}^{K} \sum_{i=0, i \neq j}^{N} x_{k,i,j} = 1, \qquad \forall j \in N \setminus \{0\} \tag{3.2}$$

$$\sum_{j=1}^{N} x_{k,0,j} = 1, \qquad \forall k \in K \tag{3.3}$$

$$\sum_{i=0, i \neq j}^{N} x_{k,i,j} = \sum_{i=0}^{N} x_{k,j,i}, \qquad \forall j \in N, \forall k \in K \tag{3.4}$$

$$\sum_{i=0}^{N} \sum_{j=1, i \neq j}^{N} q_j\, x_{k,i,j} \leq Q, \qquad \forall k \in K \tag{3.5}$$

$$\sum_{k=1}^{K} \sum_{i \in Y} \sum_{j \in Y, i \neq j} x_{k,i,j} \leq |Y| - 1, \qquad \forall Y \subset N, 2 \leq |Y| \leq |N| - 1 \tag{3.6}$$

$$x_{k,i,j} \in \{0,1\}, \qquad \forall k \in K, \forall i,j \in N \tag{3.7}$$

A função objetivo (3.1) minimiza o custo total das rotas. As Restrições (3.2) garantem que cada cliente é visitado exatamente por um veículo. As Restrições (3.3) e (3.4) são as restrições de fluxo, as quais asseguram que o veículo $k \in K$ deixa somente uma vez o depósito central e o número de veículos chegando em cada nó $i \in N$ é igual ao número de veículos saindo dele. As Restrições (3.5) garantem que as demandas dos clientes visitados numa rota não é maior que a capacidade do veículo $k \in K$. As Restrições de eliminação de sub-rotas (3.6) asseguram que a solução não contenha ciclos. As Restrições (3.7) definem o domínio das variáveis.

3.2 O *Helicopter Routing Problem with Multiple Trips* (HRPM)

O HRPM é uma extensão do modelo HRPS descrito por Rosa et al. (2016). No HRPM tem-se como objetivo atender um conjunto R de requisições de transporte usando uma frota Z de helicópteros. Os helicópteros em Z podem fazer, por dia, até t viagens. Uma viagem é definida como uma sequência de visitas começando e terminando no aeroporto. A requisição de viagem $r(i,j) \in R$ determina um nó de embarque (*pickup*) $i \in N_P$ e um nó de desembarque (*delivery*) $j = i + n$, $j \in N_D$, $n = |R|$. O aeroporto é designado como o nó 0 e o nó $2n + 1$ em referência ao local de partida e ao local de término das viagens, respectivamente. Assim, o HRPM pode ser representado pelo grafo $G(N, A)$ em que $N = N_P \cup N_D \cup N_G$, com $N_P = \{1, 2, \ldots, n\}$, $N_D = \{n + 1, n + 2, \ldots, 2n\}$, $N_G = \{0, 2n + 1\}$ e $A = N \times N$.

O HRPM possui três novidades em relação ao HRPS:

- A exigência que não ocorra a sobreposição de horários de diferentes aeronaves na mesma plataforma. Essa imposição decorre de uma característica do problema de transporte em alto-mar, em que plataformas só possuem capacidade para um helicóptero pousado. Até o trabalho de Machado et al. (2019), oriundo desta tese, nenhuma proposta de HRP impunha esse requisito. Assim, ao impor essa condição, aprimora-se a segurança das rotas ao prevenir pousos simultâneos na mesma plataforma;
- A possibilidade de múltiplas viagens para cada helicóptero. Nesse caso, uma rota é composta por múltiplas viagens. Além disso, todas as aeronaves estão limitadas a um número máximo de viagens por rota;
- Janela de tempo para o atendimento das requisições.

Para representar as múltiplas viagens, dado que todos os helicópteros possuem o mesmo número máximo de viagens, define-se um novo conjunto K de helicópteros para representá-los e elimina-se a necessidade de adicionar uma nova indexação no conjunto K. Mantém-se,

portanto, a formulação do PLIM do HRPM com a utilização de variáveis de até três índices. Isso é feito da seguinte forma. Seja $V_z = \{k_{z_1}, k_{z_2}, \ldots, k_{z_t}\}$, $|V_z| = t$, o conjunto de viagens do veículo z, em que k_{z_1} é a primeira viagem de z, k_{z_2} é a segunda viagem de z e assim por diante. O novo conjunto K de helicópteros é definido como a união de todos V_z, ou seja, $K = \cup_{z \in Z} V_z$ e $|K| = t \times |Z|$. Logo, cada viagem em V_z é, no modelo proposto para o HRPM, visto como um helicóptero que tem as mesmas propriedades de z. Além disso, as viagens de V_z são agrupadas numa sequência que forma a rota de $z \in Z$. Essa rota $z \in Z$ é definida como a sequência de viagens executadas por cada helicóptero $k_{z_1}, k_{z_2}, \ldots, k_{z_t}$. Por fim, define-se o conjunto $K_1 \subset K$ que contém a primeira viagem k_{z_1} de cada helicóptero $z \in Z$.

Uma vez dada essa definição de K, ainda é necessário assegurar que cada viagem do helicóptero $z \in Z$ é sequenciada da maneira correta. Para isso, sem perda de generalidade, define-se o parâmetro $y_{k,l}$ que controla a ordem correta dos helicópteros $k, l \in K$ para a rota $z \in Z$. O parâmetro $y_{k,l}$ é igual a 1 se os helicópteros $k, l \in V_z$ e o helicóptero k deve executar a viagem anterior a realizada pelo helicóptero l. Caso contrário, $y_{k,l} = 0$. Além disso, a rota do helicóptero $z \in Z$ é considerada válida quando as viagens são usadas em sequência, isto é, a viagem $k_{z_{i+1}}$ só pode ser usada quando a viagem k_{z_i} atende, pelo menos, uma requisição de transporte. Por fim, no retorno ao aeroporto, cada helicóptero deve aguardar um tempo mínimo, denominado tempo de serviço no aeroporto, antes de iniciar a próxima viagem.

Assim, o conjunto K representa todas as viagens dos helicópteros em Z. Por simplicidade, designa-se $k \in K$ também como um helicóptero. Dessa forma, os seguintes parâmetros são definidos para $k \in K$:

- a^k : capacidade em peso da tripulação;
- g^k : capacidade máxima de combustível;
- f^k : tempo máximo de voo;
- s^k : capacidade máxima de assentos para passageiros;
- p^k : peso do helicóptero;
- P^k : limite de peso do helicóptero durante o voo;
- z^k : consumo médio de combustível por hora;
- v^k : velocidade média do helicóptero em voo; do aeroporto; ao aeroporto;
- M_S^k : tempo de segurança do helicóptero. Este parâmetro exige que para toda a rota que o helicóptero $k \in K$ execute, ele consiga voar mais M_S^k minutos além do necessário para executar a rota. Isto implica que deve existir uma quantidade mínima de combustível no helicóptero $k \in K$ ao retornar ao aeroporto;

- H_{sr}^k : horário mínimo do dia no qual o helicóptero pode levantar voo;
- H_{sd}^k : horário máximo de retorno do helicóptero ao aeroporto;
- $y_{k,l}$: define se a viagem k deve ser realizada antes da viagem l;
- M_G: tempo de serviço no aeroporto. É o tempo mínimo entre o fim de uma viagem e o começo da próxima viagem;
- M_P: tempo de segurança para pouso na plataforma. É o tempo mínimo entre o pouso de dois helicópteros distintos na mesma plataforma;
- α^k : custo de utilizar o helicóptero $k \in K$ numa rota qualquer;
- β^k : custo de utilização do helicóptero por distância percorrida;
- θ^k : conversão de litros de combustível para peso do combustível.

Cada nó $i \in N$ do problema tem os seguintes parâmetros:

- q_i : a quantidade de passageiros;
- w_i : o peso do passageiro e bagagem;
- $c_{i,j}$: distância entre os nós $i, j \in N$;
- $d_{i,j}$: tempo de serviço no nó $j \in N$ quando se parte do nó $i \in N$.
- L_i: limite inferior da janela de atendimento do nó i;
- U_i: limite superior da janela de atendimento do nó i.

Cada requisição $r(i, j), i, j \in N$, especifica um local i de embarque (***pickup***) e um local $j = i + n$ de desembarque (***delivery***). Os locais de embarque e desembarque são constituídos pelas localidades aeroporto e plataformas marítimas. Para cada arco (i, j) que representa um percurso entre os nós i, j, existe um custo $c_{i,j}$ representando a distância entre os locais i e j. Caso um ponto de embarque ou desembarque possua mais de um passageiro, cada passageiro será representado por um novo nó e o tempo de serviço $d_{i,j}$ e a distância $c_{i,j}$ entre estes nós serão nulos. Além disso, tem-se pelo modelo matemático que $\forall i \in N$, $q_i = -q_{n+i}$ e $w_i = -w_{n+i}$.

O parâmetro $d_{i,j}$ representa o tempo de serviço no nó j quando o helicóptero parte do nó i. É necessário a informação de origem i, pois no modelo matemático o tempo de serviço numa plataforma p é fixo, independente do número de passageiros atendidos na plataforma. Assim, é preciso computar o tempo de serviço numa plataforma somente quando o helicóptero está partindo de outra plataforma, ou seja, $d_{i,j} > 0$ somente se $c_{i,j} > 0$.

No HRPM é possível que passageiros saiam ou entrem no aeroporto. Em função disso, o seguinte procedimento é executado para tratar esses casos:

- Para toda requisição que tem como origem o aeroporto, nó 0, é adicionado um novo nó i em N_P que representa o embarque desta requisição tendo $c_{0,i} = 0$ e $d_{0,i} = 0$, ou seja, a distância e o tempo de serviço do nó i em relação ao aeroporto são nulos. Além disso, dados os nós $i, j \in N$ representando o embarque de passageiros no aeroporto, tem-se $c_{i,j} = 0$ e $d_{i,j} = 0$, ou seja, a distância e tempo de serviço entre os nós i e j também são nulos;
- Para toda requisição que tem como destino o aeroporto, nó $2n + 1$, é adicionado um novo nó i em N_D, representando o desembarque desta requisição, com $c_{i,2n+1} = 0$ e $d_{i,2n+1} = 0$, ou seja, a distância e o tempo de serviço do nó i em relação ao aeroporto são nulos. Além disso, dados os nós $i, j \in N$ representando o desembarque de passageiros no aeroporto, tem-se $c_{i,j} = 0$ e $d_{i,j} = 0$, ou seja, a distância e tempo de serviço entre os nós i e j também são nulos.

As variáveis de decisão do problema são:

- $x_{i,j}^k$: variável binária. Igual a 1 caso o helicóptero $k \in K$ voe do local i ao j, $i, j \in N$. Caso contrário, igual a 0;
- u^k : variável binária. Igual a 1 caso o helicóptero $k \in K$ seja utilizado. Caso contrário, igual a 0;
- B_i^k : instante que o helicóptero $k \in K$ começa a atender o nó $i \in N$;
- Q_i^k : número de assentos ocupados após o helicóptero $k \in K$ visitar o nó $i \in N$;
- W_i^k : o peso dos passageiros (passageiros e bagagens), peso da tripulação e peso do helicóptero $k \in K$ após visitar o nó $i \in N$;
- F_i^k : a quantidade de combustível do helicóptero $k \in K$ ao visitar o nó $i \in N$.

Baseando-se nas definições expostas anteriormente, a seguinte formulação de PLIM define o HRPM:

$$\text{Minimizar:} \quad \sum_{k \in K_1} \alpha^k u^k + \sum_{k \in K} \sum_{i \in N} \sum_{j \in N, j \neq i} \beta^k c_{i,j} x_{i,j}^k \tag{3.8}$$

Sujeito a:

$$u^k = \sum_{j \in N_P} x_{0,j}^k, \quad \forall k \in K_1 \tag{3.9}$$

$$\sum_{k\in K}\sum_{j\in N} x_{i,j}^k = 1, \quad \forall i \in N_p \tag{3.10}$$

$$\sum_{j\in N} x_{i,j}^k - \sum_{j\in N} x_{n+i,j}^k = 0, \quad \forall i \in N_p, k \in K \tag{3.11}$$

$$\sum_{j\in N_P} x_{0,j}^k \leq 1, \quad \forall k \in K \tag{3.12}$$

$$\sum_{j\in N} x_{j,i}^k - \sum_{j\in N} x_{i,j}^k = 0, \quad \forall i \in N_p \cup N_D, k \in K \tag{3.13}$$

$$\sum_{j\in N_D} x_{i,2n+1}^k \leq 1, \quad \forall k \in K \tag{3.14}$$

$$\sum_{j\in N_P} x_{0,j}^k \geq \sum_{j\in N_P} x_{0,j}^l, \quad \forall k,l \in K \mid y_{k,l} = 1 \tag{3.15}$$

$$\sum_{i\in S}\sum_{j\in S} x_{i,j}^k \leq |S| - 1, \quad S \subset N, |S| \geq 2, \forall k \in K \tag{3.16}$$

$$Q_j^k \geq (Q_i^k + q_j)x_{i,j}^k, \quad \forall k \in K, i,j \in N \tag{3.17}$$

$$Q_i^k \leq s^k, \quad \forall k \in K, i \in N \tag{3.18}$$

$$Q_0^k = 0, \quad \forall k \in K \tag{3.19}$$

$$Q_{2n+1}^k = 0, \quad \forall k \in K \tag{3.20}$$

$$W_j^k \geq (W_i^k + w_j)x_{i,j}^k, \quad \forall k \in K, i,j \in N \tag{3.21}$$

$$W_i^k + \theta^k F_i^k \leq P^k, \quad \forall k \in K, i \in N \tag{3.22}$$

$$F_j^k \leq (F_i^k - z^k\frac{c_{i,j}}{v^k} - z^k d_{i,j})x_{i,j}^k, \quad \forall k \in K, i,j \in N \tag{3.23}$$

$$F_i^k \leq g^k, \quad \forall k \in K, i \in N \tag{3.24}$$

$$\sum_{i\in N}\sum_{j\in N}(\frac{c_{i,j}}{v^k} + d_{i,j})x_{i,j}^k \leq f^k, \quad \forall k \in K \tag{3.25}$$

$$B_j^k \geq (B_i^k + d_{i,j} + \frac{c_{i,j}}{v^k})x_{i,j}^k, \quad \forall k \in K, i,j \in N \tag{3.26}$$

$$B_{i+n}^k \geq B_i^k, \quad \forall k \in K, i \in N_P \tag{3.27}$$

$$B_0^l \geq B_{2n+1}^k + \left(1 - x_{l,2n+1}^k\right) M_G, \quad \forall k,l \in K \mid y_{k,l} = 1 \tag{3.28}$$

$$|B_i^k - B_j^l| \geq M_P, \quad \forall k \neq l \in K, i < j \in N_P \cup N_D \mid c_{i,j} = 0 \tag{3.29}$$

$$L_i \leq B_i^k \leq U_i, \quad \forall k \in K, i \in N_P \cup N_D \tag{3.30}$$

$$B_0^k \geq H_{sr}^k x_{0,j}^k, \quad \forall k \in K, j \in N_P \tag{3.31}$$

$$B_{2n+1}^k\ x_{i,2n+1}^k \leq H_{sd}^k, \quad \forall k \in K, i \in N_D \tag{3.32}$$

$$x_{i,j}^k \in \{0,1\}, \quad \forall k \in K, i,j \in N \tag{3.33}$$

$$u^k \in \{0,1\}, \quad \forall k \in K \tag{3.34}$$

$$Q_i^k \geq 0, \quad \forall k \in K, i \in N \tag{3.35}$$

$$W_i^k \geq p^k + a^k, \quad \forall k \in K, i \in N \tag{3.36}$$

$$F_i^k \geq z^k M_S^k, \quad \forall k \in K, i \in N \tag{3.37}$$

A função objetivo (3.8) minimiza o custo de utilização dos helicópteros para atender as requisições mais o custo da distância percorrida por cada helicóptero. Para determinar se uma

aeronave $z \in Z$ é usada em algum atendimento, verifica-se se a primeira viagem $k_{z_1} \in V_z$ atende alguma requisição.

A função objetivo é sujeita às seguintes restrições. As Restrições (3.9) relacionam as variáveis u^k e $x^k_{0,j}$, determinando quais helicópteros são utilizados. As Restrições (3.10) garantem que todas as requisições sejam atendidas. As Restrições (3.11) exigem que o mesmo helicóptero faça o embarque e desembarque do passageiro. As Restrições (3.12) a (3.14) exigem que a rota de cada helicóptero utilizado comece e termine no aeroporto.

As Restrições (3.15) exigem que, se os helicópteros $k, l \in K$ são viagens de um mesmo veículo em Z, k representa uma viagem que é anterior a l ($y_{k,l} = 1$) e l atende alguma requisição no modelo, então o helicóptero k também deve atender, pelo menos, uma requisição. As Restrições (3.16) proíbem que sejam geradas sub-rotas, sendo S um subgrafo de G e $|S|$ o número de vértices de S.

As Restrições (3.17) e (3.18) garantem que a quantidade de assentos ocupados no helicóptero $k \in K$ não passe do limite máximo de assentos s^k. As Restrições (3.19) e (3.20) exigem que o helicóptero saia e entre no aeroporto, nós $N_G = \{0, 2n + 1\}$, com zero passageiros, pois na modelagem matemática do problema, os nós de requisições que partem ou chegam no aeroporto são representados como nós em N_P ou N_D, respectivamente, com distância zero em relação ao aeroporto.

As Restrições (3.21) representam o fluxo de peso dos passageiros após a visita de cada nó. As Restrições (3.22) garantem que o limite de peso do helicóptero $k \in K$ após a visita de cada nó $i \in N$ não exceda o limite máximo P^k. As Restrições (3.23) calculam o consumo de combustível do helicóptero após a visita de cada nó em função da distância percorrida e do tempo de serviço no nó visitado. O termo $z^k \frac{c_{i,j}}{v^k}$ representa a quantidade de combustível consumida devido ao tempo necessário para percorrer $c_{i,j}$ na velocidade média v^k. O termo $z^k d_{i,j}$ quantifica a quantidade de combustível consumida enquanto o helicóptero está estacionado na plataforma j.

As Restrições (3.24) exigem que a quantidade de combustível em cada nó esteja abaixo do limite máximo de combustível g^k para o helicóptero $k \in K$. As Restrições (3.25) determinam que o tempo da rota percorrida pelo helicóptero $k \in K$ na velocidade média v^k seja menor ou igual ao tempo máximo de voo f^k do helicóptero. As Restrições (3.26) modelam o momento de visita do helicóptero no local j em função do tempo de serviço $d_{i,j}$ e do tempo necessário para percorrer o trajeto $c_{i,j}$ na velocidade média v^k.

As Restrições (3.27) garantem que o helicóptero $k \in K$ visita o nó de embarque $i \in N$ antes do nó de desembarque $i+n \in N$. Essa restrição é designada como precedência de *pickup* e *delivery*. As Restrições (3.28) asseguram que, se o helicóptero $l \in K$ inicia sua viagem após a do helicóptero $k \in K$ ($y_{k,l} = 1$), então o tempo inicial B^l_0 do helicóptero l deve ser igual ou maior ao tempo de chegada B^k_{2n+1} do helicóptero k ao aeroporto mais o tempo de serviço no

aeroporto M_G.

As Restrições (3.29) exigem que distintos helicópteros $k, l \in K$, $l \neq k$, visitando a mesma plataforma, designada pelos nós $i \neq j$, $i, j \in N_D \cup N_P$, devem ter o tempo de visita entre B_i^k e B_j^l maior que M_P. A condição para que os nós i, j representem a mesma plataforma é respeitada quando $c_{i,j} = 0$.

As Restrições (3.30) definem as janelas de tempo de cada requisição. As Restrições (3.31) garantem que o helicóptero somente saia do aeroporto após o horário H_{sr}^k. As Restrições (3.32) exigem que o helicóptero retorne ao aeroporto antes do horário H_{sd}^k.

As Restrições (3.33) a (3.37) definem as variáveis $x_{i,j}^k$ e u^k como binárias, Q_i^k como maior ou igual a zero, W_i^k como maior ou igual à soma do peso do helicóptero e da tripulação e F_i^k como maior ou igual à quantidade de combustível necessária para voar o tempo de segurança M_S^k.

Devido às Restrições (3.17), (3.21), (3.23), (3.26) e (3.32) a formulação apresentada é não linear. Contudo, estas restrições podem ser linearizadas incluindo uma constante em cada equação. Definindo $L_Q \geq max\{s^k, s^k + q_i\}$, $L_w \geq max\{p^k, p^k + w_i\}$, $L_F \geq max\{g^k\}$ e $L_B \geq max\{H_{sd}^k\}$, as equações anteriores podem ser reescritas e substituídas pelas seguintes equações, respectivamente:

$$Q_j^k \geq Q_i^k + q_j - L_Q \cdot (1 - x_{i,j}^k), \forall k \in K, i, j \in N \tag{3.38}$$

$$W_j^k \geq W_i^k + w_j - L_W \cdot (1 - x_{i,j}^k), \forall k \in K, i, j \in N \tag{3.39}$$

$$F_j^k \leq F_i^k - z^k \frac{c_{i,j}}{v^k} - z^k d_{i,j} + L_F \cdot (1 - x_{i,j}^k), \forall k \in K, i, j \in N \tag{3.40}$$

$$B_j^k \geq B_i^k + d_{i,j} + \frac{c_{i,j}}{v^k} - L_B \cdot (1 - x_{i,j}^k), \forall k \in K, i, j \in N \tag{3.41}$$

$$B_{2n+1}^k \leq H_{sd}^k + L_B \cdot (1 - x_{i,j}^k), \forall k \in K, i \in N \tag{3.42}$$

As Restrições (3.29) também são não lineares. Para a linearização, elas precisam de uma constante M grande e de novas variáveis. Seja $m_{i,j}^{k,l}$ uma variável binária relacionando os nós $i < j$, $i, j \in N$, e os helicópteros $k \neq l$, $k, l \in K$. Então, as Restrições (3.29) podem ser reescritas da seguinte forma:

$$B_i^k - B_j^l \geq M_P - M\left(m_{i,j}^{k,l}\right), \quad \forall k \neq l \in K, i < j \in N_P \cup N_D \mid c_{i,j} = 0 \tag{3.43}$$

$$-B_i^k + B_j^l \geq M_P - M\left(1 - m_{i,j}^{k,l}\right), \quad \forall k \neq l \in K, i < j \in N_P \cup N_D \mid c_{i,j} = 0 \tag{3.44}$$

$$m_{i,j}^{k,l} \in \{0, 1\} \quad \forall k, l \in K, i, j \in N_P \cup N_D \tag{3.45}$$

3.2.1 Restrições de Geração de Sub-rotas

As Restrições de fluxo (3.12) a (3.16) garantem que a solução obtida seja acíclica, começando no aeroporto 0 e terminando no aeroporto $2n + 1$. Nas Restrições (3.16), o conjunto S representa as sub-rotas do modelo matemático, ou seja, subgrafos cíclicos de $G(N, A)$.

O conjunto S deve ser conhecido antes da execução do modelo. Entretanto, gerar todas as sub-rotas para o problema é computacionalmente inviável, pois a quantidade de subgrafos de $G(N, A)$ é exponencial em relação ao seu número de nós.

Nesta tese, a seguinte estratégia é usada para garantir que a solução obtida para o modelo matemático proposto não contenha ciclos:

- Geração de sub-rotas prováveis: as sub-rotas que envolvem somente nós em N com distância zero entre si são incluídas no conjunto S antes da execução do modelo. Estas sub-rotas têm uma probabilidade maior de aparecer em alguma solução, pois a função objetivo minimiza a distância percorrida;
- Reexecução do modelo: caso a execução do modelo retorne uma sub-rota na solução, o modelo é novamente reexecutado com esta sub-rota incluída no conjunto S das Restrições (3.16).

3.3 O *Helicopter Routing Problem with Single Trip* (HRPS)

O HRPS é o modelo descrito por Rosa et al. (2016). Nesse problema, têm-se um conjunto R de requisições de transporte, $|R| = n$, a serem atendidas por no máximo $|K|$ helicópteros. A rede de transporte pode ser representada como o grafo completo $G(N, A)$ em que $N = N_P \bigcup N_D \bigcup N_G$, com $N_P = \{1, 2, \ldots, n\}$, $N_D = \{n + 1, n + 2, \ldots, 2n\}$, $N_G = \{0, 2n + 1\}$ e $A = N \times N$. N_P e N_D representam os nós de *pickup* e de *delivery*, respectivamente. O conjunto N_G é constituído do nó 0, representando o local de partida dos helicópteros e do nó $2n + 1$, o local de destino dos helicópteros no final na rota.

Dado o HRPM, o HRPS pode ser definido a partir dele removendo ou restringindo algumas exigências. Assim, para reduzir o HRPM ao HRPS, as seguintes modificações são necessárias. No HRPS, cada rota realiza uma única viagem. Logo, o conjunto de helicópteros Z se transforma em K, em que $|K| = |Z|$ e $y_{k,l} = 0$, para quaisquer $k, l \in K$. No HRPS não existem janelas de tempo, então as Restrições (3.30) são removidas. Além disso, no HRPS não existe restrição relacionada a pousos simultâneos em plataformas. Portanto, as Restrições (3.29) devem ser desconsideradas. Além disso, é necessário adicionar as Restrições (3.46) que garantem que o tempo necessário para percorrer a distância da rota pelo helicóptero $k \in K$ na velocidade média v^k mais os tempos de serviço $d_{i,j}$ seja menor ou igual à soma do tempo máximo de voo, representada pela capacidade inicial de combustível F_0^k dividida pelo consumo médio

por tempo z^k menos os tempos M_T^k, M_A^k e M_S^k representando o tempo de taxiamento, o tempo de aproximação e o tempo de segurança, respectivamente.

$$\sum_{i \in N} \sum_{j \in N} (\frac{c_{i,j}}{v^k} + d_{i,j}) x_{i,j}^k \leq \frac{F_0^k}{z^k} - \\ - M_T^k - M_A^k - M_S^k \qquad \forall k \in K, i, j \in N \tag{3.46}$$

Feito essas alterações, o HRPM se transforma no HRPS.

3.4 Considerações Finais

Neste capítulo foram apresentados os modelos de PLIM para o CVRP, o HRPS e o HRPS. Dadas essas definições, os próximos capítulos descrevem os métodos de resolução usando a abordagem RFCS para estes problemas.

4 Metodologias para o Roteamento de Veículos

Neste capítulo são apresentados os métodos básicos para resolver problemas de roteamento de veículos através do uso de heurísticas, meta-heurísticas e modelos matemáticos. A partir deles, as matheurísticas para o CVRP e HRP são definidas nos Capítulos 5 e 6, respectivamente.

4.1 Introdução

Algoritmos de otimização para o roteamento de veículos podem ser divididos em duas categorias (SÖRENSEN, 2015): algoritmos exatos e heurísticas. A diferença entres eles é que algoritmos exatos são projetados para encontrar soluções ótimas em tempo finito, enquanto heurísticas não possuem essa garantia. Portanto, estas produzem, geralmente, soluções subótimas em tempo finito.

Algoritmos exatos para roteamento são, geralmente, aplicados a modelos de PL ou PLIM que representam as restrições do problema como equações ou inequações (VANDERBEI, 2020). Entretanto, outra estratégia de resolução é, a partir de um conjunto C de rotas do problema, gerar uma solução através da seleção de $C^* \subset C$ que minimize o custo da solução. Esses modelos são denominados PCC ou PPC e esses diferem, entre si, em como o atendimento das requisições são realizadas (KELLY; XU, 1999).

Heurísticas são baseadas, essencialmente, em conhecimento e experiência de especialistas com o objetivo de explorar o espaço de busca de modo conveniente. Normalmente, esse conhecimento é representado por uma função de avaliação que afere a qualidade do estado atual no espaço de busca ou o custo de mover-se do estado corrente para outro (GAVRILAS, 2010). Além disso, existem também no campo das heurísticas três casos especiais: as meta-heurísticas, as matheurísticas e as hibridizações.

Uma meta-heurística é um *framework* de alto nível independente de problema que provê um conjunto de estratégias para desenvolver algoritmos de otimização (SÖRENSEN, 2015). As matheurísticas são algoritmos heurísticos construídos com a interoperação de meta-heurísticas e técnicas de algoritmos exatos. Hibridizações de heurísticas consistem da integração de meta-heurística com conceitos relacionados com outras técnicas, inclusive com outras heurísticas. Nessa visão, matheurísticas é um caso particular de hibridização (BLUM; ROLI, 2008).

No restante do capítulo são apresentados os componentes exatos e heurísticas utiliza-

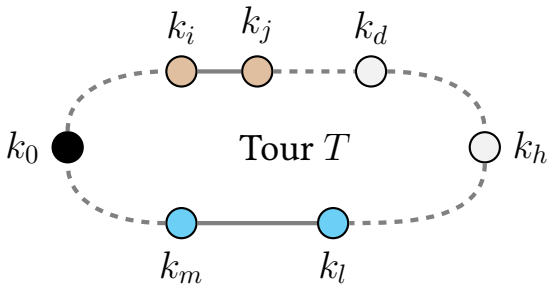
dos no método proposto para a resolução de problemas CVRP e HRP. Esses métodos operam em dois tipos de solução. Nos procedimentos construtivos, geralmente, o objetivo é gerar uma solução TSP, enquanto as heurísticas de aprimoramento lidam com soluções do CVRP ou do HRP. Dado que o CVRP e o HRP são tipos de VRP, usa-se no restante do capítulo a sigla VRP como referência genérica àqueles. Além disso, quando as soluções são designadas ótimas, seja local ou globalmente, subtende-se que o problema é de minimização. Nas próximas seções, definem-se essas soluções e seus respectivos espaços.

4.2 Convenções e Representação das Soluções

No restante desse capítulo as seguintes convenções e definições são usadas. Variáveis denominadas como S são soluções do VRP, mas quando designadas como T representam um *tour* do TSP. Para as rotas r_i do VRP, o *j*-ésimo nó nessa rota é representado como $t_j \in r_i$. De maneira similar, $t_j \in T$ indica o *j*-ésimo nó em T. A aresta $[t_k, t_l] \in r_i$ conecta os nós t_k e t_l em r_i. Os parâmetros de entrada de cada algoritmo são definidos usando a letra M e a função $\text{last}(T)$ retorna o último elemento no *tour* T.

Em termos de representação das soluções, os seguintes esquemas são usados. A solução T do TSP definida pelo grafo $G(N, E)$ é representada pela tupla fechada $T = [k_0, k_1, \cdots, k_i, k_0]$. O *tour* começa no depósito k_0, visita cada um dos nós $k_i \neq k_0$ em N e retorna ao depósito k_0. A Figura 2 retrata uma solução T do TSP. Nesse exemplo, algumas regras são adotadas. Em especial, o nó que representa o depósito (k_0) sempre é representado em preto. Uma linha cheia, como entre os nós k_i e k_j, indica que eles são adjacentes em T. Além disso, linhas tracejadas, como entre os nós k_h e k_l, significa que podem existir mais nós entre k_h e k_l. Quando necessário, os nós podem estar coloridos como k_i, k_j ou k_m e k_l para destacar a importância dos nós na representação. Por fim, o *tour* começa em k_0 e segue em sentido horário até retornar a k_0.

Figura 2 – Representação gráfica do *tour* T do TSP.

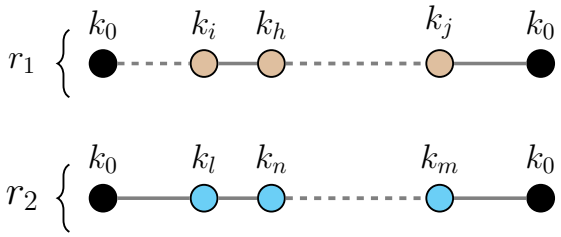


Fonte: Autor

A solução S do VRP definida pelo grafo $G(N, E)$ é representada pelo conjunto de tuplas $r \in S$, em que cada tupla $r = [k_0, k_1, \cdots, k_i, k_0]$ caracteriza uma rota. A rota começa no depósito k_0, visita pelo menos um nó $k_i \neq k_0$ em N e retorna ao depósito k_0. A Figura 3

retrata um exemplo de solução com duas rotas r_1 e r_2. De forma similar à representação do TSP, aqui também se aplica o padrão de linhas cheias/tracejadas e cores dos nós. Além disso, as rotas começam em k_0 e seguem o sentido da esquerda para a direita até chegarem em k_0 novamente.

Figura 3 – Representação gráfica das rotas r_1 e r_2 do VRP.



Fonte: Autor

4.3 Espaço de Soluções

Os métodos propostos nesse trabalho utilizam, pelo menos, dois espaços de soluções distintos: $\Omega(\mathrm{TSP})$ e $\Omega(\mathrm{VRP})$. O espaço $\Omega(\mathrm{TSP})$ é o conjunto de *tours* TSP de um grafo $G(N, E)$. Por contraste, o espaço $\Omega(\mathrm{VRP})$ é conjunto de soluções do VRP. As restrições que uma solução precisa atender para pertencer ao $\Omega(\mathrm{TSP})$ são mais brandas que no espaço $\Omega(\mathrm{VRP})$, dado que, no primeiro, a única condição é usar todos os nós em N num *tour* fechado, enquanto, no último, é necessário obedecer às restrições do VRP. Portanto, essa característica pode ser explorada para produzir e explorar mais facilmente soluções em $\Omega(\mathrm{VRP})$ a partir da construção de soluções em $\Omega(\mathrm{TSP})$.

4.4 Heurísticas Construtivas

Uma heurística construtiva é um algoritmo que constrói, iterativamente, uma solução conforme uma regra pré-determinada e termina quando uma solução é encontrada (REINELT, 2003). É esperado que as soluções assim produzidas estejam distantes entre 10% e 15% da solução ótima. Portanto, é um método usual para criar soluções iniciais em heurísticas de aprimoramento (DAVENDRA, 2010).

Nessa subseção são apresentadas as quatro heurísticas construtivas utilizadas no método proposto: a CLOSEST, a NEAREST, a FARTHEST e a MF. Elas recebem o grafo $G(N, E)$ representando o VRP, um *tour* parcial T em $\Omega(\mathrm{TSP})$, o conjunto $U \subset N$ de nós ainda não visitados em T e um número M_α, $0 \leq M_\alpha \leq 1$, que controla o aspecto guloso ou aleatório das heurísticas. A ideia geral dessas heurísticas é construir uma lista CL de movimentos candidatos a partir de T e U, filtrar CL numa lista restrita RCL de movimentos candidatos

segundo o parâmetro M_α e aplicar um movimento, aleatoriamente escolhido de RCL, a T. Isso é realizado enquanto houver algum nó U ainda não incluso em T.

O pseudocódigo da heurística CLOSEST é apresentado no Algoritmo 1. Nessa heurística, o objetivo é construir T, a partir de U, sempre incluindo novos nós ao final de T. Enquanto houver algum nó em U (linha 1), é avaliada a aplicação do movimento em T em função do custo de inseri-lo ao final de T (linha 5). Caso esse movimento seja factível segundo o método VALIDO (linha 6), ele é adicionado à lista CL de movimentos candidatos (linha 7). Cada par ordenado de CL contém na primeira posição o custo da inclusão e, na segunda posição, como a inserção deve ser feita. Após o preenchimento de CL, uma lista restrita RCL de movimentos candidatos é gerada usando CL e M_α (linhas 10-12). Em seguida, na linha 13, um movimento é escolhido aleatoriamente de RCL. Esse movimento é aplicado em T gerando a inclusão do nó $u_r \in U$ ao final de T (linha 14) e, na linha 15, u_r é removido de U. A heurística retorna a solução construída T quando U torna-se vazio (linha 17).

Algoritmo 1: CLOSEST$(G, T, U, M_\alpha,$ VALIDO$)$

entrada: Grafo $G(N, E)$, um *tour* parcial T em $\Omega(\text{TSP})$, o conjunto U de nós não visitados, um número $M_\alpha \in [0, 1]$ e a função VALIDO
saída : O *tour* $T \in \Omega(\text{TSP})$

```
1  enquanto U ≠ ∅ faça
2      CL ← ∅;
3      l ← last(T);
4      para cada u ∈ U faça
5          m ← {(d_{l,u}, Adicione a aresta (l, u) a T)};
6          se VALIDO(m, T) = TRUE então
7              CL ← CL ∪ m;
8          fim
9      fim
10     p_min ← min {w | (w, u) ∈ RCL};
11     p_max ← max {w | (w, u) ∈ RCL};
12     RCL ← {(w_u, u) ∈ CL | w_u ≤ p_min + M_α( p_max − p_min )};
13     Selecione (w_r, u_r) aleatoriamente de RCL;
14     Aplique u_r a T;
15     Remova o nó em u_r de U;
16 fim
17 retorne T;
```

O pseudocódigo da heurística NEAREST é apresentado no Algoritmo 2. Nessa heurística, o objetivo é construir T, a partir de U, incluindo novos nós em T em qualquer posição. A avaliação do custo é em função da menor distância. Enquanto houver algum nó em U (linha 1), uma lista CL de movimentos candidatos é construída usando cada nó $u \in U$ (linha 8), avaliando o custo de inseri-lo em T entre dois nós consecutivos $v, x \in T$ (linha 4), desde que o movimento seja válido (linha 7). Esse custo é representado pela distância entre v, u e u, x, menos a distância entre v, x (linha 5). Cada par ordenado de CL contém na primeira posição

o custo dessa inclusão e, na segunda posição, como a inserção deve ser feita. Após o preenchimento de CL, uma lista restrita RCL de movimentos candidatos é gerada usando CL e M_α (linhas 12-14). Em seguida, na linha 15, um movimento é escolhido aleatoriamente de RCL. Esse movimento é aplicado em T gerando a inserção do nó $u_r \in U$ em T (linha 16) e, na linha 17, u_r é removido de U. A heurística NEAREST retorna a solução construída T quando U torna-se vazio (linha 19).

Algoritmo 2: NEAREST$(G, T, U, M_\alpha, \text{VALIDO})$

entrada: Grafo $G(N, E)$, um *tour* parcial T em $\Omega(\text{TSP})$, o conjunto U de nós não visitados, um número $M_\alpha \in [0, 1]$ e a função VALIDO
saída : O *tour* $T \in \Omega(\text{TSP})$

```
1  enquanto U ≠ ∅ faça
2      CL ← ∅;
3      para cada u ∈ U faça
4          para cada aresta e = (v, x) ∈ T faça
5              w ← d_{v,u} + d_{u,x} − d_{v,x};
6              m ← {(w, Adicione as arestas (v, u), (u, x) e remova (v, x) de T)};
7              se VALIDO(m, T) = TRUE então
8                  CL ← CL ∪ m;
9              fim
10         fim
11     fim
12     p_min ← min {w | (w, u) ∈ RCL};
13     p_max ← max {w | (w, u) ∈ RCL};
14     RCL ← {(w_u, u) ∈ CL | w_u ≤ p_min + M_α( p_max − p_min )};
15     Selecione (w_r, u_r) aleatoriamente de RCL;
16     Aplique u_r a T;
17     Remova nó em u_r de U;
18 fim
19 retorne T;
```

O pseudocódigo da heurística FARTHEST é apresentado no Algoritmo 3. Nessa heurística, o objetivo é construir T, a partir de U, incluindo novos nós em qualquer posição de T. Entretanto, diferente de NEAREST, nesse procedimento é dado prioridade a movimentos que aumentem, o máximo possível, a distância total de T. Assim, enquanto houver algum nó em U (linha 1), uma lista de candidatos CL é construída usando U (linha 8), avaliando o custo de inseri-lo em T entre dois nós consecutivos $v, x \in T$ (linha 4) e desde que o movimento seja válido (linha 7). Esse custo é representado pelo negativo da distância entre v, u e u, x, mais a distância entre v, x (linha 5). Cada par ordenado de CL contém na primeira posição o custo dessa inclusão e, na segunda posição, como a inserção deve ser feita. Após o preenchimento de CL, uma lista restrita de candidatos é gerada usando CL e M_α (linhas 12-14). Em seguida, na linha 15, um movimento é escolhido aleatoriamente de RCL. Esse movimento é aplicado em T gerando a inserção do nó $u_r \in U$ em T (linha 16) e, na linha 17, u_r é removido de U. A heurística retorna a solução construída T quando U torna-se vazio (linha 19).

Algoritmo 3: FARTHEST(G, T, U, M_α, VALIDO)

entrada: Grafo $G(N, E)$, um *tour* parcial T em Ω(TSP), o conjunto U de nós não visitados, um número $M_\alpha \in [0, 1]$ e a função VALIDO
saída : O *tour* $T \in \Omega$(TSP)

1 **enquanto** $U \neq \emptyset$ **faça**
2 $CL \leftarrow \emptyset$;
3 **para cada** $u \in U$ **faça**
4 **para cada** *aresta* $e = (v, x) \in T$ **faça**
5 $w \leftarrow -(d_{v,u} + d_{u,x} - d_{v,x})$;
6 $m \leftarrow \{(w,$ Adicione as arestas $(v, u), (u, x)$ e remova (v, x) de $T)\}$;
7 **se** VALIDO$(m, T) =$ TRUE **então**
8 $CL \leftarrow CL \cup m$;
9 **fim**
10 **fim**
11 **fim**
12 $p_{min} \leftarrow min\{w \mid (w, u) \in RCL\}$;
13 $p_{max} \leftarrow max\{w \mid (w, u) \in RCL\}$;
14 $RCL \leftarrow \{(w_u, u) \in CL \mid w_u \leq p_{min} + M_\alpha(\, p_{max} - p_{min}\,)\}$;
15 Selecione (w_r, u_r) aleatoriamente de RCL;
16 Aplique u_r em T;
17 Remoa nó em u_r de U;
18 **fim**
19 **retorne** T;

O pseudocódigo da heurística MF é apresentado no Algoritmo 4. Nessa heurística, um *tour* T é gerado a partir da construção de *tours* menores chamados fragmentos (BENTLEY, 1992). A ideia é gerar, aumentar ou combinar fragmentos usando as arestas em E. Nessa lógica, pode acontecer: a) a criação de um novo fragmento, quando ao incluir a aresta $e = (v_1, v_2)$ não existe ainda os nós v_1 e v_2 em T; b) a adição de uma aresta e no início ou no fim de um fragmento existente, quando a aresta e incide somente num fragmento em T ou c) unir dois fragmentos, quando a aresta $e = (v_1, v_2)$ tem incidência em dois fragmentos. Assim, para gerar uma solução T, o procedimento executa um laço enquanto houver algum nó em U (linha 1). Dentro desse laço, a lista CL de movimentos candidatos é construída usando a lista de arestas E (linha 3), avaliando se a inclusão dessa aresta no conjunto de fragmentos é inviável. Isso acontece quando uma aresta $e = (v_1, v_2) \in E$ gera um ciclo em T com $|T| < |N|$ ou quando os nós v_1 ou v_2 são internos a algum fragmento em T (linha 4). Caso a inclusão seja viável, o movimento representado pela inclusão da aresta e é adicionado a CL como um par ordenado, em que a primeira posição representa o peso da aresta e e, na segunda posição, como a inserção deve ser feita. Após o preenchimento de CL, uma lista restrita RCL de candidatos é gerada usando CL e M_α (linhas 12-14). Em seguida, na linha 15, um movimento é escolhido aleatoriamente de RCL. Esse movimento é aplicado em T gerando a inserção da aresta e em T (linha 16) e, na linha 17, os nós adicionados em T pelo movimento são removidos de U. A heurística retorna a solução construída T quando U torna-se vazio (linha 19).

Algoritmo 4: MF(G, T, U, M_α, VALIDO)

entrada: Grafo $G(N, E)$, um *tour* parcial T em Ω(TSP), o conjunto U de nós não visitados, um número $M_\alpha \in [0, 1]$ e a função VALIDO
saída : O *tour* $T \in \Omega$(TSP)

```
enquanto U ≠ ∅ faça
    CL ← ∅;
    para cada aresta e = (v1, v2) ∈ E faça
        se (e fecha o tour T e |T| < |N|) ou (∃F ∈ T ⟹ v1 ou v2 são interiores a F )
         então
            Vá para a próxima aresta;
        fim
        m ← (weight(e), Adicione a aresta e a T);
        se VALIDO(m, T) então
            CL ← CL ∪ m;
        fim
    fim
    p_min ← min {w | (w, u) ∈ RCL};
    p_max ← max {w | (w, u) ∈ RCL};
    RCL ← {(w_u, u) ∈ CL | w_u ≤ p_min + M_α( p_max − p_min )};
    Selecione (w_r, u_r) aleatoriamente de RCL;
    Aplique u_r em T;
    Remova nó em u_r de U;
fim
retorne T;
```

O Algoritmo 5 mostra o pseudocódigo para a geração de uma solução construtiva. O procedimento escolhe aleatoriamente uma heurística H dentre as heurísticas CLOSEST, NEAREST, FARTHEST e MF. Então, usa H para gerar T. Note que a solução T pode ser vazia no início do procedimento HEURISTICA-CONSTRUTIVA. Neste caso, a heurística construtiva gera toda a solução. Caso T seja um *tour* parcial, a heurística construtiva somente completa T usando o nós ainda não presentes em T, ou seja, os nós no conjunto U.

Algoritmo 5: HEURISTICA-CONSTRUTIVA(G, T, M_α, VALIDO)

entrada: Grafo $G(N, E)$, um *tour* parcial T em $\Omega(TSP)$, um número $M_\alpha \in [0, 1]$ e a função VALIDO
saída : O *tour* T

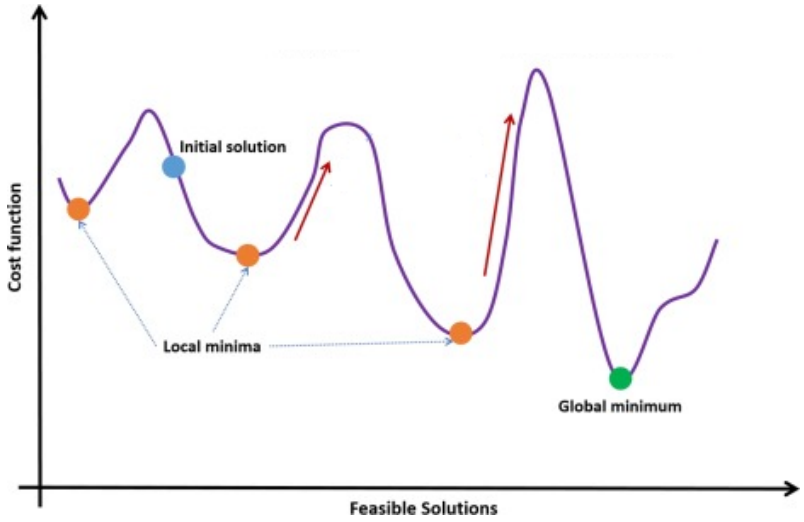
```
U ← U \ N;
H ← Selecione aleatoriamente uma heurística construtiva em {CLOSEST, FARTHEST,
  NEAREST, MF};
T ← H(G, T, U, M_α, VALIDO);
retorne T;
```

4.5 Heurística de Busca Local

Uma heurística de aprimoramento é um algoritmo que inicia como uma solução inicial e a melhora por sucessivas mudanças. Um tipo comum de heurística de aprimoramento é a busca local (LOURENÇO; MARTIN; STÜTZLE, 2019). Nela, uma série de modificações na solução corrente S é definida com o objetivo de gerar um conjunto de soluções vizinhas a S, em que cada modificação é conhecida como um movimento. A busca local compreende uma laço em que a melhor solução vizinha torna-se a nova solução corrente na próxima iteração. A heurística termina quando não é mais possível melhorar a solução corrente S.

A Figura 4 mostra um exemplo de aplicação de busca local. Ela parte de uma solução inicial (*initial solution* na figura). A partir de movimentos locais, novas regiões do espaço de busca podem ser explorados. Em especial, existe o interesse em dois deles: os mínimos locais (*local minima* na figura) e o mínimo global (*global minimum*). Num problema de minimização, o objetivo da busca é encontrar o mínimo global, a solução como o menor custo. Entretanto, geralmente a heurística fica presa num mínimo local, uma solução que não possui nenhum vizinho melhor.

Figura 4 – Exemplo de Busca Local.

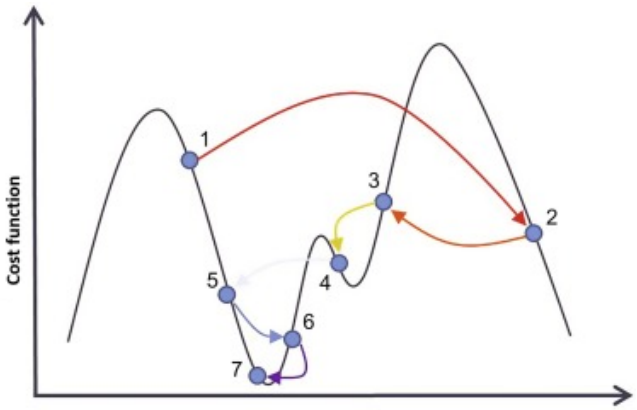


Fonte: Blocho (2020)

A Figura 5 mostra um exemplo da aplicação de movimentos na busca local. Ela parte de uma solução inicial rotulada como 1. Diversos movimentos são aplicados gerando as soluções rotuladas de 2 a 7. Essa última é uma solução localmente ótima para os movimentos disponíveis.

A heurística de busca local é específica para cada tipo de VRP. Entretanto, o funcionamento dela pode ser esboçado conforme mostrado no Algoritmo 6. Ela funciona com um conjunto MOV de movimentos para o problema. Aplicam-se esses movimentos à solução corrente S e gera-se o conjunto NH de soluções vizinhas. Em seguida, seleciona-se a melhor

Figura 5 – Exemplo de Movimento



Fonte: Martinez e Cao (2018)

solução S^* presente em NH. Essa solução é comparada com a atual S. Se for melhor, a solução S^* torna-se a solução da próxima iteração. O laço de aplicação de movimentos e seleção de soluções é realizado enquanto é possível melhorar a solução corrente. O método retorna como resultado a melhor solução encontrada.

Algoritmo 6: BL-GENERICA(S)

entrada: Uma solução $S \in \Omega(\text{VRP})$
saída : Uma solução $S \in \Omega(\text{VRP})$
repetir $\leftarrow$ TRUE;
enquanto *repetir* = TRUE **faça**
  Seja MOV o conjunto de movimentos da busca local;
  Preenche NH com as soluções vizinhas a S usando os movimentos em MOV;
  Selecione a melhor solução S^* de NH;
  se $\text{cost}(S^*) <$ S **então**
    $S \leftarrow S^*$;
  senão
    repetir $\leftarrow$ FALSE;
  fim
fim
retorne S;

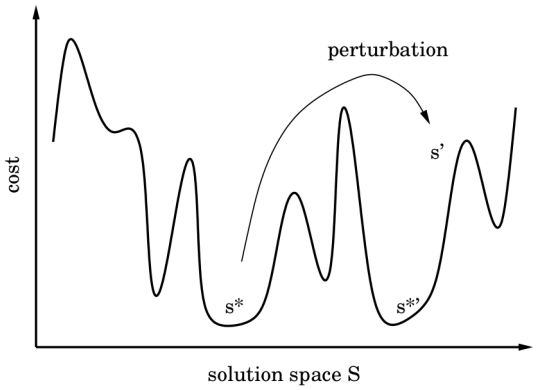
4.6 Heurística de Pertubação

A heurística de pertubação adiciona, aleatoriamente, uma modificação na solução $T \in \Omega(\text{TSP})$ para aumentar a diversidade de um conjunto de soluções e evitar que a busca local permaneça presa num mínimo local. As modificações são chamadas pertubação (LOURENÇO; MARTIN; STÜTZLE, 2019).

Na Figura 6 é mostrado um exemplo de pertubação. Inicialmente, existe uma solução s^*

que é localmente ótima em relação aos movimentos da busca local. Para continuar a explorar o espaço de busca, uma pertubação é realizada em s^* gerando como resultado uma nova solução s'. Após isso, uma heurística de busca local é aplicada, a qual pode encontrar um novo mínimo local na solução $s^{*\prime}$.

Figura 6 – Exemplo de Pertubação.



Fonte: Lourenço, Martin e Stützle (2019)

De forma semelhante à busca local, a heurística de pertubação é específica para cada tipo de VRP. De forma genérica, conforme mostrado no Algoritmo 7, a heurística seleciona um caminho de tamanho p e um movimento. A partir disso, o movimento usa p para fazer a pertubação na solução T. A solução perturbada T é retornada como resultado.

Algoritmo 7: PERTUBA-GENERICO(T, M_t)

entrada: O *tour* $T \in \Omega(TSP)$ e um número $M_t \in [0, 1]$
saída : O *tour* $T \in \Omega(TSP)$

1 Seja MOV o conjunto de movimentos de pertubação;
2 Selecione o movimento de pertubação m, aleatoriamente, em MOV;
3 Seja p um número inteiro aleatório no intervalo $[1, 1 + M_t * |T|]$;
4 Aplique m a T usando o caminho p;
5 **retorne** T;

4.7 Meta-heurísticas VNS e GRASP

A *Variable Neighborhood Search* (VNS), proposta por Hansen e Mladenović (1999), é uma meta-heurística para resolver problemas combinatórios que consiste de duas etapas: pertubação e busca local. A ideia geral dessa meta-heurística é a mudança metódica de vizinhança combinada com uma fase de descida para encontrar ótimos locais e a aplicação de pertubações para sair desses mínimos locais (HANSEN; MLADENOVIĆ, 2014). O Algoritmo 8 mostra um pseudocódigo genérico do VNS. A meta-heurística recebe uma solução inicial e o número de iterações que devem ser realizadas. Em cada uma destas, uma pertubação é realizada na solução corrente S, gerando uma nova solução. Nela, é aplicado algum procedimento de busca

local. Caso essa solução localmente ótima seja melhor que a solução corrente, ela é utilizada nas próximas iterações. O resultado da meta-heurística é melhor solução encontrada.

Algoritmo 8: VNS-GENERICO(S, M_v)

entrada: Uma solução $S \in \Omega(\text{VRP})$ e o número máximo M_v de iterações do VNS
saída : Uma solução $S^* \in \Omega(\text{VRP})$

```
S* ← S;
para k = 1, ··· , M_v faça
    S ← S*;
    Pertube S;
    Aplique a buscal local a S;
    se cost(S) < cost(S*) então
        S* ← S;
    fim
fim
retorne S*;
```

O *Greedy Randomized Adaptive Search Procedure* (GRASP), proposto por Feo e Resende (1989), é uma meta-heurística que incorpora componentes gulosos, probabilísticos e adaptativos na resolução de problemas combinatórios. Cada iteração do GRASP é dividida em duas partes: i) fase de construção e ii) fase de busca local. Na fase de construção, uma solução S é gulosa e probabilisticamente construída para ser usada como solução inicial. A segunda fase, a busca local, repetidamente substitui a solução atual S pela melhor solução em NH, em que NH representa o conjunto de soluções vizinhas a S. A solução final do GRASP é o melhor S^* encontrado entre todas as iterações (RESENDE; RIBEIRO, 2010).

A meta-heurística GRASP proposta é específica para cada tipo de problema VRP. Entretanto, é possível esboçar um pseudocódigo genérico conforme mostrado no Algoritmo 9. O procedimento recebe os dois parâmetros principais que controlam o número de iterações e a aleatoriedade. Ele itera por um número específico de vezes, em que uma solução é construída usando um algoritmo construtivo cujos movimentos são selecionados a partir da lista restrita RCL. Dada uma solução inicial S, a busca local é aplicada para encontrar o mínimo local. Após a finalização das iterações, o procedimento retorna a melhor solução S^* encontrada.

Com o GRASP é uma meta-heurística com aplicação de busca local, ele pode ser hibridizado com outras meta-heurísticas que realizem exploração do espaço de busca. No caso da hibridização de GRASP e VNS, este torna-se o procedimento usado na fase de busca local do GRASP. Essa hibridização é mostrada no Algoritmo 10.

Algoritmo 9: GRASP-GENERICO(G, M_i, M_α)

entrada: Grafo $G(N, E)$, o número máximo M_i de iterações do GRASP e um número $M_\alpha \in [0, 1]$
saída : Uma solução $S^* \in \Omega(\text{VRP})$

```
S* ← ∅;
para k = 1, ··· , M_i faça
    Construa S com uma heurística construtiva usando M_α;
    Aplique a buscal local a S;
    se cost(S) < cost(S*) então
        S* ← S;
    fim
fim
retorne S*;
```

Algoritmo 10: GRASP-VNS-HIBRIDO(G, M_i, M_α)

entrada: Grafo $G(N, E)$, o número máximo M_i de iterações do GRASP e um número $M_\alpha \in [0, 1]$
saída : Uma solução $S^* \in \Omega(\text{VRP})$

```
S* ← ∅;
para k = 1, ··· , M_i faça
    Construa S com uma heurística construtiva usando M_α;
    Aplique o VNS a S;
    se cost(S) < cost(S*) então
        S* ← S;
    fim
fim
retorne S*;
```

4.8 Problema de Cobertura de Conjuntos (PCC) e Problema de Particionamento de Conjuntos (PPC)

O Problema de Cobertura de Conjuntos (PCC) e o Problema de Particionamento de Conjuntos (PPC) são formulações clássicas para resolver o VRP (BALINSKI; QUANDT, 1964; AGARWAL; MATHUR; SALKIN, 1989). A ideia base em ambos métodos é gerar todas as rotas viáveis (isto é, rotas que iniciam e terminam no depósito e que atendam todas as restrições do problema) e então determinar, entre essas rotas, um subconjunto que atenda a todas as requisições do VRP com o menor custo possível (BRAMEL; SIMCHI-LEVI, 2002). A principal diferença entre esses métodos é se um nó pode ou não ser visitado por mais de uma rota. No PCC, isso é possível, enquanto o PPC demanda que qualquer nó deve ser visitado exatamente por uma rota.

Os modelos matemáticos clássicos do PCC e do PPC são definidos as seguir (CAPRARA; TOTH; FISCHETTI, 2000). Seja $A = (a_{i,j})$ uma matriz binária de dimensão $m \times n$, $c = (c_j)$

um vetor real de dimensão n, $I_M = \{1, \cdots, m\}$ e $I_N = \{1, \cdots, n\}$ conjuntos de indexação. A coluna $j \in I_N$ representa o índice de uma rota. O valor $c_j > 0$, $j \in I_N$, representa o custo da coluna j (custo da rota j). Define-se que a coluna $j \in I_N$ cobre a linha $i \in I_M$ se $a_{i,j} = 1$, isto é, a rota j visita o nó i. O objetivo é encontrar o subconjunto de custo mínimo $C \subseteq I_N$ de colunas tal que que cada linha $i \in I_M$ é coberta por, pelo menos, uma coluna $j \in C$. Seja x_j, $j \in C$, uma variável de decisão binária com $x_j = 1$ se $j \in C$ ou $x_j = 0$, caso contrário. O modelo de Programação Inteira para o Problema de Cobertura de Conjuntos (PI-PCC) para o VRP pode ser escrito da seguinte forma:

$$\text{Minimizar} \sum_{j \in I_N} c_j x_j \tag{4.1}$$

Sujeito a:

$$\sum_{j \in I_N} a_{i,j} x_j \geq 1, \forall i \in I_M \tag{4.2}$$

$$\sum_{j \in I_N} x_j \leq |K|, \tag{4.3}$$

$$x \in \{0, 1\}, \qquad \forall j \in I_N \tag{4.4}$$

O modelo de Programação Inteira para o Problema de Particionamento de Conjuntos (PI-PPC) é definido como se segue (BRAMEL; SIMCHI-LEVI, 2002):

$$\text{Minimizar} \sum_{j \in I_N} c_j x_j \tag{4.5}$$

Sujeito a:

$$\sum_{j \in I_N} a_{i,j} x_j = 1, \forall i \in I_M \tag{4.6}$$

$$\sum_{j \in I_N} x_j \leq |K|, \tag{4.7}$$

$$x \in \{0, 1\}, \qquad \forall j \in I_N \tag{4.8}$$

Neste trabalho, duas modificações são realizadas nesses modelos. A primeira alteração é na formulação do PI-PCC (4.1)-(4.4), pois o modelo exato é difícil de resolver mesmo para instâncias relativamente pequenas (BRAMEL; SIMCHI-LEVI, 1997), visto que o PCC é classificado como NP-difícil (LENSTRA; KAN, 1981). De modo a superar essa característica, a abordagem utilizada é resolver a relaxação linear do modelo PI-PCC. O modelo de Relaxação Linear do Problema de Corbetura de Conjuntos (RL-PCC) é definido da seguinte forma:

$$\text{Minimizar} \sum_{j \in I_N} c_j x_j \tag{4.9}$$

Sujeito a: (4.10)

$$\sum_{j \in I_N} a_{i,j} x_j \geq 1, \forall i \in I_M \tag{4.11}$$

$$\sum_{j \in I_N} x_j \leq |K|, \tag{4.12}$$

$$0 \leq x \leq 1, \qquad \forall j \in I_N \tag{4.13}$$

A segunda alteração é a inclusão de novas restrições no modelo PI-PPC para colunas (rotas) i, j que não possam estar na mesma solução, mesmo que não atendam os mesmos nós. Para isso, define-se uma nova função chamada $\mathrm{PPC\text{-}RESTRICAO}(i, j)$ que determina se as rotas i e j podem pertencer a uma mesma solução no PI-PPC. Essa função é específica para os problemas CVRP e HRP. Neste último caso, o objetivo é impedir que se gere uma solução para o HRPM com rotas que violem as restrições de pouso seguro. A nova restrição é proposta usando a função PPC-RESTRICAO da seguinte forma:

$$(x_i - 1)B_i + \sum_{j \in I_M} x_j * \mathrm{PPC\text{-}RESTRICAO}(x_i, x_j) \leq 0, \forall i \in I_M \tag{4.14}$$

em que $B_i = \sum_{j \in I_M} \mathrm{PPC\text{-}RESTRICAO}(x_i, x_j)$ é uma constante.

Os modelos PCC e PPC precisam do conjunto π_s de soluções em $\Omega(\mathrm{VRP})$ para montar a matriz $A_{m \times n}$ e o vetor coluna $c_{1 \times n}$. No Algoritmo 11 esse procedimento é mostrado para a matriz do PCC. Ele recebe como entrada o conjunto π_s de soluções no espaço $\Omega(\mathrm{VRP})$. No procedimento, inicialmente, o conjunto R é preenchido com cada rota que ocorre no conjunto de soluções π_S (linha 3). Então, a matriz $A_{m \times n}$ e o vetor $c_{1 \times n}$ são preenchidos com os valores apropriados no laço principal do procedimento (linhas 10-19). Os resultados do procedimento são a matriz A e o vetor c.

No Algoritmo 12 é mostrado o procedimento para construir a matriz do PPC. Ele recebe como entrada o conjunto π_s de soluções no espaço $\Omega(\mathrm{VRP})$ e a função $\mathrm{PPC\text{-}RESTRICAO}$. No procedimento, inicialmente, constrói-se a matriz A e o vetor coluna c usando o método PCC-MATRIZES (linha 1). Então, um laço é realizado para detectar rotas r_j que possuem alguma restrição com outras rotas representadas na matriz A (linhas 4-14). Para cada rota r_j, uma nova linha é adicionada a A. O resultado do algoritmo é a matriz modificada A e o vetor coluna c.

Definidos os modos de construir as matrizes do PCC e do PPC, o próximo passo é definir os métodos de resolução usando esses modelos. No caso do PCC, optou-se por gerar uma solução T em $\Omega(\mathrm{TSP})$, pois nele usa-se a versão relaxada do modelo matemático. Assim, é mais simples construir uma solução nesse espaço que gerar uma solução em $\Omega(\mathrm{VRP})$. O método para o PPC gera uma solução completa no espaço $\Omega(\mathrm{VRP})$.

No método proposto, a solução inicial é construída pelo PCC-SOLUCAO a partir de soluções locais ótimas geradas pelo VNS e pelas heurísticas construtivas. Dado que o RL-PCC é um modelo relaxado para o VRP, o conjunto R de rotas produzidas por esse modelo não representa uma solução viável em $\Omega(\mathrm{VRP})$ ou mesmo um conjunto ordenado de rotas para o VRP em termos de qualidade de solução. Portanto, a estratégia escolhida é mapear esse conjunto C

Algoritmo 11: PCC-MATRIZES(G, π_S)

entrada: Grafo $G(N, E)$ e o conjunto π_S de soluções em $\Omega(\text{VRP})$
saída : Uma matriz $A_{m\times n}$ e um vetor coluna $c_{1\times n}$

$R \leftarrow \emptyset$;
para cada $S \in \pi_S$ **faça**
  $R \leftarrow R \leftarrow \{r \mid r$ é um rota em $S\}$;
fim
Seja n o número de rotas em R;
Seja $m = |N|$;
$A_{m\times n} = [a_{i,j}]$, em que $a_{i,j} = 0$;
$c = [0, \cdots, 0]_{1\times n}$;
$j = 1$;
para cada $r \in R$ **faça**
  para cada $i \in N \setminus k_0$ **faça**
    se $i \in r$ **então**
      $A[i-1, j] = 1$;
    fim
  fim
  $c[j] = cost(r)$;
  $A[m, j] = 1$;
  $j \leftarrow j + 1$;
fim
retorne $[A_{m\times n}, c_{1\times n}]$;

Algoritmo 12: PPC-MATRIZES$(G, \pi_S, \text{PPC-RESTRICAO})$

entrada: Grafo $G(N, E)$, conjunto π_S de soluções $\Omega(\text{VRP})$ e a função PPC-RESTRICAO
saída : Uma matriz $A_{m\times n}$ e um vetor coluna $c_{1\times n}$

$[A_{k\times n}, c_{1\times n}] \leftarrow$ PCC-MATRIZES(G, π_S);
Seja $n = |N|$;
Seja I_N o conjunto que indexa as colunas de A;
para cada *coluna* j *de* I_N **faça**
  Seja r_j a rota que a coluna j representa;
  Seja v um vetor linha de n colunas;
  Seja $B_i = \sum_{j\in I_N} \text{PPC-RESTRICAO}(x_i, x_j)$;
  $v[j] = B_i$;
  para cada *coluna* k *de* I_N, $k \neq j$ **faça**
    Seja r_k a rota que a coluna k representa;
    $v[k] = \text{PPC-RESTRICAO}(r_j, r_k)$;
  fim
  Adicione o vetor linha v como uma nova linha de A;
fim
retorne $[A_{m\times n}, c_{1\times n}]$;

numa solução parcial T de $\Omega(\text{TSP})$. Para fazer isso, é preferível que o conjunto R esteja ordenado em função de alguma característica, por isso, define-se quatro estratégias designadas pela letra grega ϕ com esse objetivo. Ao aplicar uma dessas estratégias, é possível construir uma solução parcial $T \in \Omega(\text{TSP})$ a partir do conjunto R e, se necessário, finalizar a solução T com uma heurística construtiva. Esse esquema de usar RL-PCC, soluções geradas pelo VNS e heurísticas construtivas é aplicado como uma tática para diversificar a exploração do espaço de busca e para aumentar a qualidade da solução inicial $T \in \Omega(\text{TSP})$.

O procedimento PCC-SOLUCAO para a construção de uma solução é apresentado no Algoritmo 13. Ele recebe como entrada o grafo $G(N, E)$ representando o problema, o conjunto $\pi_{PCC} \subset \Omega(\text{VRP})$ de soluções e a estratégia de ordenação ϕ. Na linha 2, as matrizes A e c são montadas usando as soluções em π_{PCC}. Em seguida, na linha 3, o RL-PCC é resolvido usando essas matrizes com a geração do vetor solução x, o qual é transformado no vetor R de rotas selecionadas ($x_j > 0$) na solução do VRP (linha 4). Na linha 5, o vetor R de soluções é ordenado conforme a estratégia ϕ. Em seguida, no laço 6-11, a solução T é construída usando R como base. Nesse laço, para cada rota $r \in R$, é verificado se algum elemento de r já ocorre em T (linha 8). Se isso for falso, então significa que r pode ser adicionado ao final de T (linha 9). O procedimento PCC-SOLUCAO retorna como resultado a solução (possivelmente parcial) $T \in \Omega(\text{TSP})$.

Algoritmo 13: PCC-SOLUCAO(G, π_{PCC}, ϕ)

entrada: Grafo $G(N, E)$, o conjunto π_{PCC} de soluções em $\Omega(\text{VRP})$ e a estratégia de ordenamento ϕ
saída : Uma solução $T \in \Omega(\text{TSP})$

```
1  T ← ∅;
2  [A, c] ← PCC-MATRIZES(G, π_PCC);
3  Seja x a solução exata do RL-PCC usando A_{m×n} and c_{1×n};
4  Seja R = {(x_j, r_j)| x_j > 0 e r_j a rota da coluna j} ;
5  R ← φ(R);
6  para cada i ← 1, ··· , |R| faça
7  |  r ← rota armazenada em R[i];
8  |  se Nenhum nó em r está em T então
9  |  |  T ← T ∪ r;
10 |  fim
11 fim
12 retorne T;
```

A estratégia $\phi(R)$ é uma função que ordena o vetor R de rotas de acordo com o valor de x_j na primeira componente do par ordenado. Quatro estratégias são propostas conforme é apresentado na Tabela 8. Utiliza-se o modelo relaxado do PCC, portanto o número de rotas pode ser maior que o número de veículos $|K|$ e a estratégia considera essa característica.

No Algoritmo 14 é apresentado o método para gerar soluções S em $\Omega(\text{VRP})$ usando o PPC. Ele recebe como entrada o grafo $G(N, E)$ representando o problema, o conjunto $\pi_{PPC} \subset$

Tabela 8 – Lista de estratégias propostas para ordenar o vetor R de rotas.

Estratégia	Descrição
$\phi_b(R)$	Ordena o vetor R de rotas em ordem decrescente. Assim, rotas r_j com os melhores valores x_j são colocadas no começo do vetor ordenado.
$\phi_w(R)$	Ordena o vetor R de rotas em ordem crescente. Assim, rotas r_j com os melhores valores x_j são colocadas no final do vetor ordenado.
$\phi_{sb}(R)$	Ordena o vetor R de rotas em ordem decrescente. Então, permute aleatoriamente as primeiras $\|K\|$ posições do vetor ordenado R. Assim, as rotas com os melhores valores de x_j são colocadas nas primeiras $\|K\|$ posições do vetor ordenado, mas em posições aleatórias.
$\phi_{sw}(R)$	Ordena o vetor R de rotas em ordem crescente. Assim, as rotas com os piores valores de x_j são colocadas nas primeiras $\|K\|$ posições do vetor ordenado, mas em posições aleatórias.

$\Omega(\text{VRP})$ de soluções e a função PPC-RESTRICAO. Na linha 1, as matrizes A e c são montadas usando as soluções em π_{PPC}. Em seguida, na linha 2, o PPC é resolvido usando essas matrizes com a geração do vetor solução x, o qual é transformado na solução final S a partir das rotas selecionadas ($x_j = 1$) do VRP (linha 3).

Algoritmo 14: PPC-SOLUCAO(G, π_{PPC}, PPC-RESTRICAO)

entrada: Grafo $G(N, E)$, o conjunto π_{PPC} de soluções em $\Omega(\text{VRP})$ e a função PPC-RESTRICAO
saída : Uma solução $S \in \Omega(\text{VRP})$

1 $[A, c] \leftarrow$ PPC-MATRIZES(G, π_{PPC}, PPC-RESTRICAO);
2 Seja x a solução exata do PPC usando $A_{m\times n}$ and $c_{1\times n}$;
3 Seja $S = \{r_j | x_j = 1$ e r_j a rota da coluna $j\}$;
4 **retorne** S;

A seguir, um exemplo de aplicação dos algoritmos PCC-SOLUCAO e PPC-SOLUCAO é apresentado. Seja $\pi = \pi_{PCC} = \pi_{PPC}$ o conjunto de soluções para o grafo $G(N, E)$ com $N = \{0, 1, 2, 3, 4\}$ para algum problema VRP com três veículos ($|K| = 3$):

$$\pi = \{S_1, S_2, S_3\}$$

em que S_i, $1 \leq i \leq 3$, são as seguintes soluções:

$$\begin{aligned}
S_1 &= \{r_{1,1} : [0, 1, 2, 0]\,, \\
&\quad\; r_{1,2} : [0, 3, 4, 0]\} \\
S_2 &= \{r_{2,1} : [0, 1, 0]\,, \\
&\quad\; r_{2,2} : [0, 3, 2, 0]\,, \\
&\quad\; r_{2,3} : [0, 4, 0]\} \\
S_3 &= \{r_{3,1} : [0, 1, 3, 0]\,,
\end{aligned}$$

$r_{3,2} : [0, 2, 4, 0]\}$

A matriz A construída por PCC-MATRIZES para π é :

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ K \end{array} \begin{array}{c} \begin{array}{ccccccc} r_{1,1} & r_{1,2} & r_{2,1} & r_{2,2} & r_{2,3} & r_{3,1} & r_{3,2} \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

em que as colunas $r_{i,j}$ representam as rotas j da solução S_i, as linhas $1 \leq i \leq 4$ representam o atendimento de cada nó i em V e a linha K relaciona-se ao número máximo de veículos utilizados (Restrições 4.3))

O vetor coluna c que representa o custo de cada rota construída por PCC-MATRIZES para π pode ser, por exemplo, o seguinte:

$$c^T = \begin{array}{c} \begin{array}{ccccccc} r_{1,1} & r_{1,2} & r_{2,1} & r_{2,2} & r_{2,3} & r_{3,1} & r_{3,2} \end{array} \\ \begin{pmatrix} 20 & 20 & 10 & 15 & 16 & 24 & 29 \end{pmatrix} \end{array}$$

Resolvendo A e c no modelo relaxado RL-PCC conforme realizado em PCC-SOLUCAO (linha 3), o seguinte vetor solução é obtido:

$$x^T = \begin{array}{c} \begin{array}{ccccccc} r_{1,1} & r_{1,2} & r_{2,1} & r_{2,2} & r_{2,3} & r_{3,1} & r_{3,2} \end{array} \\ \begin{pmatrix} 0{,}5 & 0 & 0{,}5 & 0{,}5 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Em seguida, formam-se pares ordenados de valor solução e índice da rota quando $x_i > 0$:

$$R = [(0{,}5; r_{1,1}), (0{,}5; r_{2,1}), (0{,}5; r_{2,2}), (1; r_{2,3})]$$

Ainda no PCC-SOLUCAO (linha 5), aplica-se uma função de ordenação em R. Suponha que seja $\phi_b(R)$, a qual ordena de forma decrescente em relação ao valor da primeira componente do par ordenado. Então, R torna-se:

$$R = [(1; r_{2,3}), (0{,}5; r_{1,1}), (0{,}5; r_{2,1}), (0{,}5; r_{2,2})]$$

O próximo passo no método PCC-SOLUCAO é construir uma solução T segundo o vetor ordenado R. Assim, primeiro se adiciona a rota $r_{2,3} : [0, 4, 0]$ a T:

$$T = [0, 4, 0]$$

adiciona-se, a seguir, a rota $r_{1,1} : [0, 1, 2, 0]$:

$$T = [0, 4, 1, 2, 0]$$

As adições de $r_{2,1} : [0, 1, 0]$ e de $r_{2,2} : [0, 3, 2, 0]$ não podem ser feitas a T, pois 1 e 2 já estão em T. Logo, a solução (parcial) de PCC-SOLUCAO é:

$$T = [0, 4, 1, 2, 0]$$

Em relação ao método PPC-SOLUCAO, ele usa as matrizes A e c como base, sendo que aquela ainda é completada com a adição de novas linhas que indicam restrições entre rotas. Suponha que $\text{PPC-RESTRICAO}(v, w) = \text{PPC-RESTRICAO}(w, v) = 1$ somente quando $v = r_{2,2}$ e $w = r_{3,1}$ e quando $v = r_{2,2}$ e $w = r_{3,2}$. Para os outros casos, é 0. Então a matriz modificada A utilizada por PPC-SOLUCAO é:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ K \\ r_{1,1} \\ r_{1,2} \\ r_{2,1} \\ r_{2,2} \\ r_{2,3} \\ r_{3,1} \\ r_{3,2} \end{array} \begin{array}{c} \begin{array}{ccccccc} r_{1,1} & r_{1,2} & r_{2,1} & r_{2,2} & r_{2,3} & r_{3,1} & r_{3,2} \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Note que as linhas nulas de A não precisam ser representadas e são, geralmente, removidas pelos pacotes de resolução de PL. Resolvendo A e c no método PPC-SOLUCAO, a seguinte solução é encontrada para o modelo matemático:

$$x^T = \begin{array}{c} \begin{array}{ccccccc} r_{1,1} & r_{1,2} & r_{2,1} & r_{2,2} & r_{2,3} & r_{3,1} & r_{3,2} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Então a solução S gerada por PPC-SOLUCAO é o conjunto de rotas $\{r_{1,1}, r_{1,2}\}$.

4.9 Considerações Finais

Nesse capítulo foram apresentados os componentes do método proposto. No próximo capítulo é apresentada a matheurística proposta aplicada a problemas CVRP.

5 Metodologia de Resolução do CVRP com Matheurística

Neste capítulo é apresentado o método proposto, chamado Matheurística GRASP+VNS para o CVRP (MGV-C), para resolver o problema CVRP através do uso de matheurística, abordagem hierárquica e hibridizações. Para isso, todos os algoritmos apresentados no Capítulo 4 são utilizados no novo método.

5.1 Introdução

Neste capítulo, uma abordagem hierárquica segundo a estratégia RFCS é proposta para resolver o CVRP usando uma matheurística híbrida de *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Variable Neighborhood Search* (VNS). Inicialmente, o roteamento (*route-first*) é realizado usando heurísticas construtivas e o modelo matemático do Problema de Cobertura de Conjuntos (PCC). O PCC utiliza soluções locais ótimas encontradas em iterações prévias do VNS para criar um *tour* parcial, o qual é completado por heurísticas construtivas, se necessário. O roteamento compreende a fase construtiva do GRASP. Em seguida, o passo de *clustering* (*cluster-second*) é realizado usando um algoritmo de *splitting*, uma busca local e o VNS. Esse passo compreende a busca local do GRASP. Essa abordagem de roteamento e *clustering* é executada por um número determinado de vezes. Por fim, o modelo matemático do Problema de Particionamento de Conjuntos (PPC) fornece uma solução final que é, pelo menos, melhor ou igual às soluções locais encontradas anteriormente no procedimento. O método apresentado é considerado uma matheurística visto que o PCC, os métodos construtivos e o VNS colaboram ativamente entre si através de um mecanismo de retroalimentação para explorar distintas regiões do espaço de soluções.

5.2 Método Proposto

O MGV-C utiliza diversos espaços de soluções. Dada a necessidade de operar em espaços distintos, duas funções, chamadas de SPLIT-CVRP e CONCAT, fazem a conversão entre esses diferentes espaços. Além disso, utilizam-se os modelos matemáticos PCC e PPC para gerar soluções parciais e a solução final da matheurística, respectivamente.

Na primeira fase do GRASP, uma solução candidata T é construída usando o modelo matemático do Problema de Cobertura de Conjuntos (PCC), o qual usa as soluções prévias geradas no VNS para resolver o modelo. Como a solução candidata T pode ser incompleta, heurísticas construtivas são usadas para terminar a construção de T se necessário. Nessa parte

da matheurística, o PCC faz uma ligação entre as soluções prévias geradas pelo VNS e as soluções parciais fornecidas pela fase construtiva do GRASP. Note que no início do GRASP, não há nenhuma solução gerada pelo VNS, portanto, o PCC fornece uma solução vazia e a solução dessa fase é construída inteiramente pela heurística construtiva. Além disso, para impedir que o PCC fique preso num mínimo local, o conjunto de soluções usadas no PCC é esvaziado caso o VNS fique um certo número de iterações sem encontrar uma solução superior à melhor solução já encontrada.

Após a construção da solução inicial, inicia-se a fase de busca local do GRASP. Na matheurística MGV-C, essa fase é realizada pelo VNS. Essa heurística usa uma série de procedimentos de pertubação e de busca local para escapar de mínimos locais e explorar novas regiões do espaço de busca. Além disso, o VNS mantém o registro dos ótimos locais encontrados para serem usados na fase construtiva. Nessa fase, é necessário converter a solução T para a solução S num espaço de busca distinto e vice-versa. Isso é necessário porque as operações do VNS operam em ambos espaços de busca.

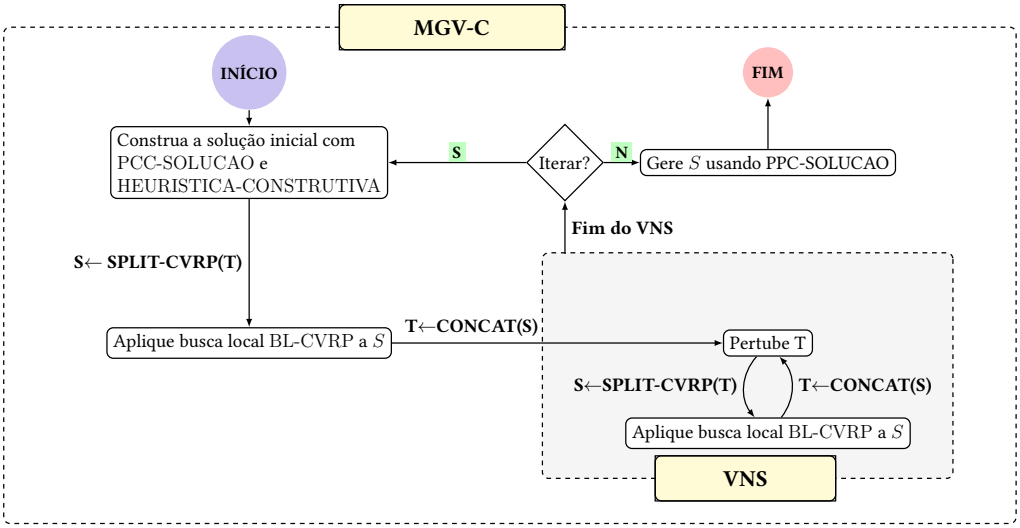
As fases de construção e busca local são realizadas por um número determinado de iterações. Com a finalização dessas iterações do GRASP, o Problema de Particionamento de Conjuntos (PPC) é usado para criar uma solução final a partir das soluções localmente ótimas geradas no decorrer do GRASP.

A Figura 7 mostra uma visão geral do método proposto. Os seguintes passos são realizados na matheurística MGV-C:

- Inicia-se pela fase construtiva do GRASP;
- Uma solução construtiva T (espaço $\Omega(\mathrm{TSP})$) é gerada pelo PCC usando o método PCC-SOLUCAO;
- A solução T pode ser incompleta. Por isso, o método HEURISTICA-CONSTRUTIVA é usado para completá-la;
- A solução T (espaço $\Omega(\mathrm{TSP})$) é convertida para a solução S (espaço $\Omega(\mathrm{VRP})$) usando o método de *splitting* SPLIT-CVRP;
- Inicia-se a fase de busca local do GRASP;
- Uma busca local é realizada na solução S;
- Entra-se no ciclo de pertubação e busca local do VNS;
- Na pertubação do VNS, a solução S é convertida para uma solução T usando o método CONCAT;
- A solução T é modificada usando um movimento de pertubação;

- A solução modificada T é convertida na solução S usando SPLIT-CVRP. Em S é aplicada uma busca local;
- Esse ciclo de pertubação e busca local do VNS é executado por um número determinado de vezes;
- Finalizado o VNS, verifica-se se há ainda iterações do GRASP. Se sim, o procedimento inicia um novo ciclo de fase construtiva e busca local do GRASP. Caso contrário, uma solução é construída usando o modelo matemático PPC.

Figura 7 – Visão geral do método proposto MGV-C.



Fonte: Autor

O pseudocódigo para o CVRP, denominado nesta tese de MGV-C, é definido no Algoritmo 15. Ele recebe o grafo $G(N, E)$ representado o CVRP, o número máximo de iterações M_i, o número $M_\alpha \in [0, 1]$ que controla o grau de aleatoriedade nas heurísticas construtivas, o número de iterações M_v do VNS, o número máximo de iterações M_s sem melhoria no VNS, o número $M_t \in [0, 1]$ que controla o grau de pertubação e a estratégia de ordenação ϕ. O resultado do algoritmo é uma solução $S \in \Omega(\mathrm{VRP})$. No início do procedimento, nas linhas 1-4, as variáveis de controle de iterações e os conjuntos de soluções são definidos com os seus valores iniciais. No laço, entre as linhas 5 e 32, encontra-se o bloco principal da meta-heurística GRASP com a construção e melhoria de uma solução. Nesse bloco, a primeira etapa é construir uma solução inicial T no espaço $\Omega(\mathrm{TSP})$ via procedimento PCC-SOLUCAO (Algoritmo 13) na linha 6. Esse procedimento recebe o grafo $G(N, E)$ do CVRP e as soluções construídas anteriormente contidas em π_{pcc}. Uma vez que uma solução inicial é construída pelo PCC-SOLUCAO, ela pode ser incompleta ou vazia. Assim, o método HEURISTICA-CONSTRUTIVA é aplicada a T para completá-la. O método HEURISTICA-CONSTRUTIVA (Algoritmo 5) recebe como entrada o grafo $G(N, E)$ do CVRP, o parâmetro M_α que controla a lista restrita RCL de movimentos candidatos e a função VALIDO-CVRP que determina se uma solução T é válida para o CVRP.

A seguir, a solução T é transformada para o espaço $\Omega(\mathrm{VRP})$ (linha 8), uma busca local é aplicada nela (linha 9) e a solução resultante é repassada para o laço interno que representa o VNS (linhas 11-25). No VNS, um laço de aplicação de pertubação e busca local é executado. O resultado dele é o número acumulado de iterações sem melhoria i_c, a melhor solução encontrada S e o conjunto π_S preenchido com as soluções localmente ótimas encontradas no VNS. Finalizada a sequência clássica de construção e melhoria do GRASP, o procedimento atualiza as variáveis de controle em seguida. Os conjuntos π_{PCC} e π_{PPC} são preenchidos com as novas soluções encontradas no VNS (linhas 26 e 27). No bloco de controle da linha 28 examina-se se o limite máximo de iterações sem melhoria foi atingido. Se for o caso, então o conjunto π_{PCC} é esvaziado e a variável i_c recebe o valor nulo. O objetivo aqui é reiniciar as soluções em π_{PCC} de modo a evitar que o procedimento PCC-SOLUCAO (Algoritmo 13) prenda-se numa solução localmente ótima. Como último passo, após todas as iterações do algoritmo, o PPC-SOLUCAO (Algoritmo 14) é aplicado, na linha 33, usando o conjunto π_{PPC}, que contém todas as soluções localmente ótimas encontradas pelo VNS, para gerar a solução final $S \in \Omega(\mathrm{VRP})$.

Note que o Algoritmo 15 utiliza os espaços $\Omega(\mathrm{TSP})$ e $\Omega(\mathrm{VRP})$. Apesar de ser uma aplicação no CVRP, manteve-se a designação das soluções do CVRP como $\Omega(\mathrm{VRP})$. Além disso, os métodos VALIDO-CVRP, SPLIT-CVRP, BL-CVRP, PERTUBA-CVRP e RESTRICAO-CVRP são específicos para problemas CVRP e são apresentados no restante do capítulo. Os outros métodos do MGV-C já foram analisados e discutidos no Capítulo 4. No Apêndice B mostra-se um exemplo de aplicação da matheurística MGV-C.

5.3 Representação Computacional

As soluções dos espaços $\Omega(\mathrm{TSP})$ e $\Omega(\mathrm{VRP})$ são representadas como vetores de inteiros em memória, em que os inteiros designam cada nó do grafo $G(N, E)$ do problema CVRP. No Algoritmo 15 da matheurística MGV-C, uma das partes mais custosas computacionalmente é a geração das vizinhanças no método de busca local BL-CVRP (Algoritmo 19).

Nesse trabalho, a representação das soluções no espaço $\Omega(\mathrm{VRP})$ também possui vetores auxiliares utilizados pelos movimentos da busca local, conforme proposto por Prins (2009). O emprego desses vetores auxiliares permite que a busca local BL-CVRP (Algoritmo 19) calcule o custo da solução vizinha a S sem a necessidade da aplicação computacional do movimento em análise. Ou seja, não é necessário gerar a solução S^+ vizinha a S para determinar seu real custo. Isso pode ser feito somente com as informações armazenadas nos vetores auxiliares. Dessa forma, a quantidade de tempo computacional dispendido durante a aplicação do procedimento BL-CVRP (Algoritmo 19) é reduzida.

Seja T uma solução no espaço $\Omega(\mathrm{TSP})$ e $G(N, E)$ o grafo do problema CVRP, então o vetor Z_T de tamanho $|Z_T| = |N| + 1$ é definido da seguinte forma:

Algoritmo 15: MGV-C(G, M_i, M_α, M_t, M_v, M_s, ϕ)

entrada: Grafo $G(N, E)$, o número máximo M_i de iterações do GRASP, um número $M_\alpha \in [0, 1]$, um número $M_t \in [0, 1]$, o número máximo M_v de iterações do VNS, o número máximo M_s de iterações sem aprimoramento e a estratégia de ordenamento ϕ

saída : Uma solução $S \in \Omega(\text{VRP})$

$i_c \leftarrow 0$;
$\pi_{PCC} \leftarrow \emptyset$;
$\pi_{PPC} \leftarrow \emptyset$;
$\bar{S} \leftarrow \emptyset$;
para $i = 0, \cdots, M_i$ **faça**
  $T \leftarrow \text{PCC-SOLUCAO}(G, \pi_{PCC}, \phi)$;
  $T \leftarrow \text{HEURISTICA-CONSTRUTIVA}(G, T, M_\alpha, \text{VALIDO-CVRP})$;
  $S \leftarrow \text{SPLIT-CVRP}(T)$;
  $S \leftarrow \text{BL-CVRP}(S)$;
  $\pi_S \leftarrow \emptyset$;
  para $k = 1, \cdots, M_v$ **faça**
    $i_c \leftarrow i_c + 1$;
    $T^* \leftarrow \text{CONCAT}(S)$;
    $T^* \leftarrow \text{PERTUBA-CVRP}(T^*, M_t)$;
    $S^* \leftarrow \text{SPLIT-CVRP}(T^*)$;
    $S^* \leftarrow \text{BL-CVRP}(S)$;
    $\pi_S \leftarrow \pi_S \cup \{S^*\}$;
    se $\text{cost}(S^*) < \text{cost}(S)$ **então**
      $S \leftarrow S^*$;
    fim
    se $\text{cost}(S^*) < \text{cost}(\bar{S})$ **então**
      $i_c \leftarrow 0$;
      $\bar{S} \leftarrow S$;
    fim
  fim
  $\pi_{PCC} \leftarrow \pi_{PCC} \cup \pi_S$;
  $\pi_{PPC} \leftarrow \pi_{PPC} \cup \pi_S$;
  se $i_c \geq M_s$ **então**
    $\pi_{PCC} \leftarrow \emptyset$;
    $i_c \leftarrow 0$;
  fim
fim
$S \leftarrow \text{PPC-SOLUCAO}(G, \pi_{PPC}, \text{RESTRICAO-CVRP})$;
retorne S;

$$Z_T[i] = \Big\{ T\,[i] \qquad \text{se } 0 \leq i < |N| + 1$$

Seja $S = \{s_1, s_2, \cdots, s_r\}$ uma solução com r rotas no espaço $\Omega(\text{VRP})$ e $G(N, E)$ o grafo do problema CVRP, então o vetor Z_S de tamanho $|Z_S| = |N| + 2 * (r + 1)$ é definido da seguinte forma:

$$Z_S = Z_{s_1} + Z_{s_2} + \cdots Z_{s_r}$$

em que $+$ é a operação de concatenação de dois vetores Z_i e Z_j e $Z_{s_1}, Z_{s_2}, \cdots, Z_{s_r}$ são as representações computacionais de cada rota de S:

$$Z_{s_k}[i] = \Big\{ s_k\,[i] \qquad \text{se } 0 \leq i < |s_k|$$

Os vetores auxiliares de S são o vetor de rota Z_R, os vetores de demanda Z_Q^b e Z_Q^a e os vetores de custo Z_C^b e Z_C^a. Todos os vetores possuem o tamanho de Z_S. O vetor de rota Z_R retorna na posição i a rota a que $Z_S[i]$ pertence:

$$Z_R[i] = \Big\{ \text{o número representando a rota } s \in S \text{ a que } Z_S[i] \text{ pertence}$$

Os vetores de demanda Z_Q^b e Z_Q^a denotam, na posição i, a demanda total atendida da rota antes do atendimento de i e após o atendimento de i, respectivamente. Assim,

$$Z_Q^b[i] = \begin{cases} \text{a demanda total atendida até } Z_S[i-1], \text{ partindo de } Z_S[j] = k_0, \\ \text{em que } Z_R[j] = Z_R[i] \text{ e } j < i \end{cases}$$

$$Z_Q^a[i] = \begin{cases} \text{a demanda total atendida após } Z_S[i], \text{ até de } Z_S[j] = k_0, \\ \text{em que } Z_R[j] = Z_R[i] \text{ e } i < j \end{cases}$$

De maneira semelhante, os vetores de custo Z_C^b e Z_C^a denotam, na posição i, o custo total da rota antes do atendimento de i e após o atendimento de i, respectivamente. Assim,

$$Z_C^b[i] = \begin{cases} \text{O custo total até } Z_S[i-1], \text{ partindo de } Z_S[j] = k_0, \\ \text{em que } Z_R[j] = Z_R[i] \text{ e } j < i \end{cases}$$

$$Z_C^a[i] = \begin{cases} \text{O custo total após } Z_S[i], \text{ até de } Z_S[j] = k_0, \\ \text{em que } Z_R[j] = Z_R[i] \text{ e } i < j \end{cases}$$

5.4 Heurística de Busca Local

Na busca local do CVRP, três movimentos são utilizados como descrito por Prins (2009): 2-OPT, CROSSOVER e OR-OPT. No movimento 2-OPT, conforme descrito no Algoritmo 16, as soluções vizinhas são geradas a partir de S usando cada par de rotas $r_i, r_j \in S$. Nesse

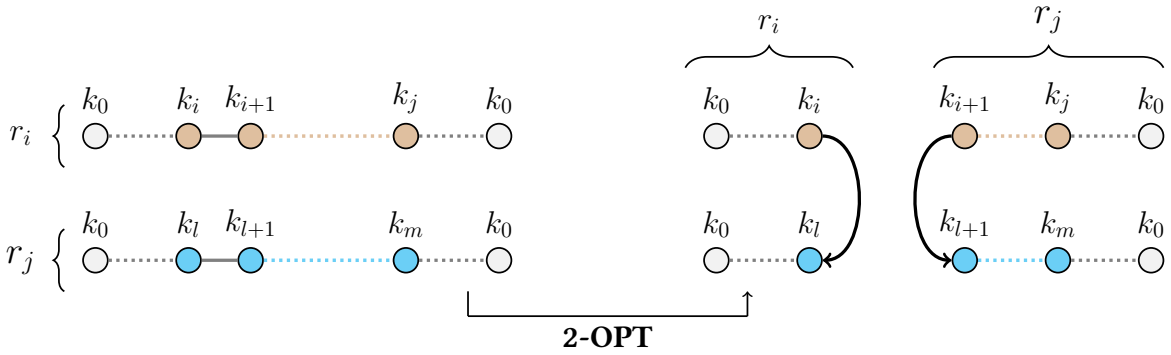
movimento, as arestas $[k_i, k_{i+1}] \in r_i$ e $[k_l, k_{l+1}] \in r_j$ são substituídas pelas novas arestas $[k_i, k_l]$ e $[k_{i+1}, k_{l+1}]$ (linhas 5-6). Cada nova solução vizinha gerada é registrada no conjunto P (linha 7). O algoritmo retorna o conjunto P de soluções vizinhas do movimento 2-OPT. A Figura 8 mostra um movimento genérico do 2-OPT.

Algoritmo 16: 2-OPT(S)

entrada: Uma solução $S \in \Omega(\text{VRP})$
saída : Um conjunto P de soluções em $\Omega(\text{VRP})$

```
1  P ← ∅;
2  para cada r_i ∈ S e r_j ∈ S, r_i ≠ r_j faça
3      para cada [k_i, k_{i+1}] ∈ r_i e [k_l, k_{l+1}] ∈ r_j faça
4          S^+ ← S;
5          Remova as arestas [k_i, k_{i+1}] e [k_l, k_{l+1}] de S^+;
6          Adicione as arestas [k_i, k_l] and [k_{i+1}, k_{l+1}] em S^+;
7          P ← P ∪ {S^+};
8      fim
9  fim
10 retorne P;
```

Figura 8 – Movimento 2-OPT no espaço $\Omega(\text{VRP})$.



Fonte: Autor

De forma semelhante, o movimento CROSSOVER substitui as arestas $[k_i, k_{i+1}] \in r_i$ e $[k_l, k_{l+1}] \in r_j$ pelas arestas $[k_l, k_{i+1}]$ e $[k_i, k_{l+1}]$ em S para gerar uma solução vizinha S^+. O Algoritmo 17 e a Figura 9 apresentam o pseudocódigo e um movimento genérico do CROSSOVER, respectivamente.

O movimento OR-OPT consiste em realocar um caminho de p nós consecutivos da rota r_i para alguma posição da rota r_j. O Algoritmo 18 e a Figura 10 apresentam o pseudocódigo e um movimento genérico do OR-OPT, respectivamente.

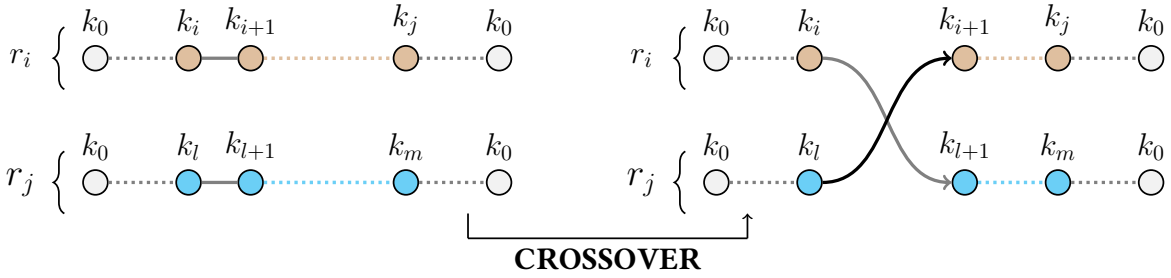
O Algoritmo 19 mostra o pseudocódigo da busca local BL-CVRP. Ela mantém uma solução corrente S e preenche o conjunto $\text{NH}(S)$ com os vizinhos de S gerados pelos movimentos 2-OPT, CROSSOVER e OR-OPT (linha 3). Uma vez que $\text{NH}(S)$ esteja preenchido, a melhor

Algoritmo 17: CROSSOVER(S)

entrada: Uma solução $S \in \Omega(\text{VRP})$
saída : Um conjunto P de soluções em $\Omega(\text{VRP})$

$P \leftarrow \emptyset$;
para cada $r_i \in S$ *e* $r_j \in S, r_i \neq r_j$ **faça**
  para cada $[k_i, k_{i+1}] \in r_i$ *e* $[k_l, k_{l+1}] \in r_j$ **faça**
    $S^+ \leftarrow S$;
    Remova as arestas $[k_i, k_{i+1}]$ e $[k_l, k_{l+1}]$ de S^+;
    Adicione as arestas $[k_l, k_{i+1}]$ e $[k_i, k_{l+1}]$ em S^+;
    $P \leftarrow P \cup \{S^+\}$;
  fim
fim
retorne P;

Figura 9 – O movimento CROSSOVER no espaço $\Omega(\text{VRP})$.



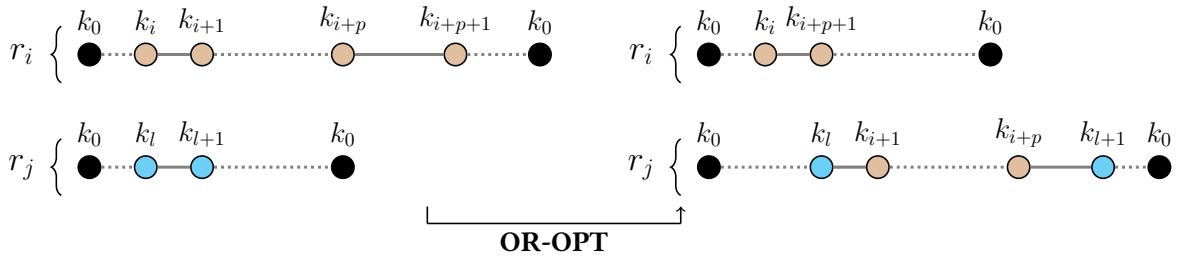
Fonte: Autor

Algoritmo 18: OR-OPT(S, p)

entrada: Uma solução $S \in \Omega(\text{VRP})$ e um número inteiro p
saída : Um conjunto P de soluções em $\Omega(\text{VRP})$

$P \leftarrow \emptyset$;
para cada $r_i \in S$ *e* $r_j \in S$ **faça**
  para cada *caminho* $c = [t_k, \cdots, t_l]$ *em* $r_i \in S$ *com tamanho* p **faça**
    $S^+ \leftarrow S$;
    Remova c de r_i em S^+;
    Adicione, aleatoriamente, o caminho c em $r_j \in S^+$;
    $P \leftarrow P \cup \{S^+\}$;
  fim
fim
retorne P;

Figura 10 – O movimento OR-OPT no espaço $\Omega(\text{VRP})$.



Fonte: Autor

solução S^* de $\text{NH}(S)$ é comparada com a solução atual S (linha 5). Se S^* é melhor que S, então ela se torna a nova solução inicial S na próxima iteração. A busca local termina quando nenhuma solução $\text{NH}(S)$ melhora o valor da solução corrente S. O algoritmo retorna como resultado a última solução corrente S.

Algoritmo 19: BL-$\text{CVRP}(S)$

```
   entrada: Uma solução S ∈ Ω(VRP)
   saída  : Uma solução S ∈ Ω(VRP)
1  repetir ← TRUE;
2  enquanto repetir = TRUE faça
3  |  NH(S) ← 2-OPT(S) ∪ CROSSOVER(S) ∪ OR-OPT(S,1) ∪ OR-OPT(S,2);
4  |  Selecione a melhor solução S* de NH(S);
5  |  se cost(S*) < S então
6  |  |  S ← S*;
7  |  senão
8  |  |  repetir ← FALSE;
9  |  fim
10 fim
11 retorne S;
```

Conforme descrito anteriormente, a representação computacional das soluções permite fazer a seleção da melhor vizinhança sem a necessidade de executar o movimento da busca local. Para detalhes de como isso pode ser feito para os movimentos 2-OPT, CROSSOVER e OR-OPT, os trabalhos de Prins (2009) e Irnich, Funke e Grünert (2006) podem ser consultados.

5.5 Heurística de Pertubação

A pertubação adotada para o CVRP usa dois tipos de movimentos: SWAP e REINSERTION. O operador SWAP seleciona, aleatoriamente, dois caminhos l_1 e l_2 de tamanho p in $T \in \Omega(\text{TSP})$ e os permuta em T, desde que l_1 e l_2 não compartilhem nenhum nó ($l_1 \cap l_2 = \emptyset$).

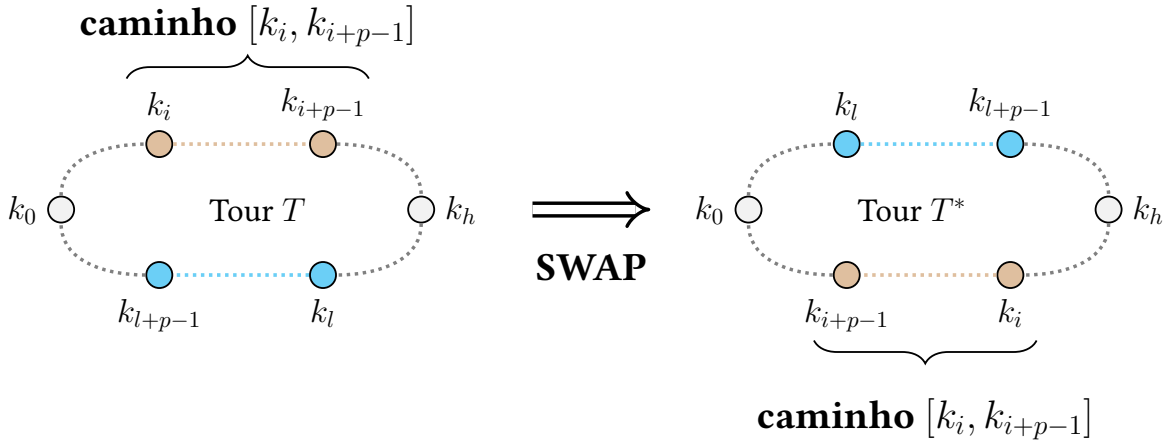
O Algoritmo 20 e a Figura 11 representam o pseudocódigo e um exemplo genérico do movimento SWAP, respectivamente.

Algoritmo 20: SWAP(T, p)

entrada: O *tour* $T \in \Omega(\text{TSP})$ e um número inteiro p
saída : O *tour* $T \in \Omega(\text{TSP})$

```
Seleciona, aleatoriamente, dois caminhos l_1 e l_2 de tamanho p na solução T, tal que
  l_1 ∩ l_2 = ∅;
para cada i ← 1, ··· , p faça
    Permute p_1[i] e p_2[i] em T;
fim
retorne T ;
```

Figura 11 – Movimento SWAP de p nós no espaço $\Omega(\text{TSP})$.



Fonte: Autor

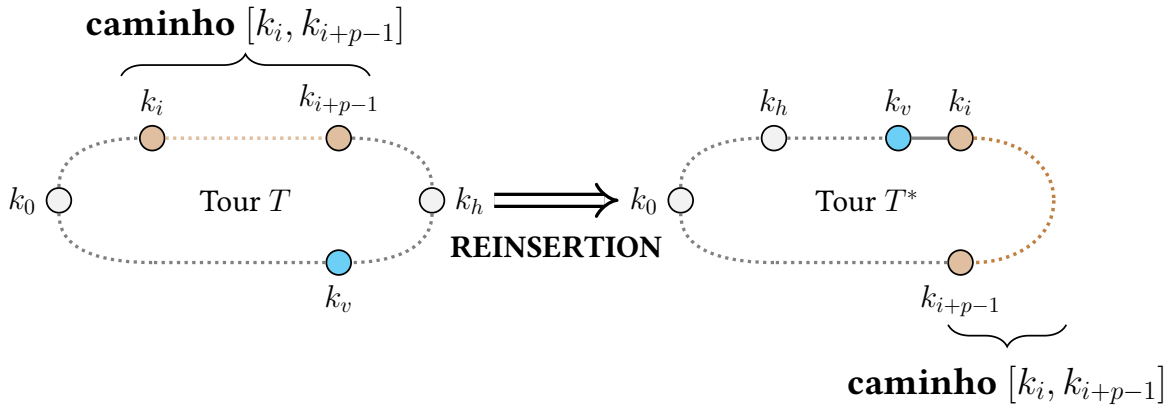
O movimento REINSERTION, aleatoriamente, seleciona um caminho l de tamanho p e um nó v em em $T \in \Omega(\text{TSP})$, em que $v \notin l$. A partir disso, o caminho l é removido da sua posição original em T e adicionado imediatamente após o nó v em T. O Algoritmo 21 e a Figura 12 representam o pseudocódigo e um exemplo genérico do movimento REINSERTION, respectivamente.

Algoritmo 21: REINSERTION(T, p)

entrada: O *tour* $T \in \Omega(\text{TSP})$ e um número inteiro p
saída : O *tour* $T \in \Omega(\text{TSP})$

```
Selecione, aleatoriamente, um caminho l de tamanho p em T;
Selecione, aleatoriamente, um nó v de T tal que v ∉ l;
Remova l de T;
Adicione l imediatamente após v em T;
retorne T ;
```

Figura 12 – Movimento REINSERTION de p nós no espaço $\Omega(\text{TSP})$.



Fonte: Autor

O Algoritmo 22 mostra a heurística de pertubação PERTUBA-CVRP. O parâmetro M_t determina o número máximo de nós em T que podem ser selecionados em algum caminho a partir dos operadores disponíveis. Em cada aplicação da heurística, um operador m é escolhido aleatoriamente e um número p é selecionado no intervalo $[1, 1 + M_t * |T|]$. Por fim, o movimento é aplicado a T e uma nova solução $T^* \in \Omega(\text{TSP})$ é obtida.

Algoritmo 22: PERTUBA-CVRP(T, M_t)

entrada: O *tour* $T \in \Omega(\text{TSP})$ e um número $M_t \in [0, 1]$
saída : O *tour* $T^* \in \Omega(\text{TSP})$
1 Selecione o operador de pertubação m, aleatoriamente, de $\{\text{SWAP}, \text{REINSERTION}\}$;
2 Seja p um número inteiro aleatório no intervalo $[1, 1 + M_t * |T|]$;
3 $T^* \leftarrow m(T, p)$;
4 **retorne** T^*;

5.6 Métodos de Transformação de Espaço

O método SPLIT-CVRP, mostrado no Algoritmo 23, usa como base o procedimento de Bellman (GOLDBERG; RADZIK, 1993). Ele é formado por duas partes distintas. Na primeira, linhas 3-22, o algoritmo realiza uma busca sequencial na solução T para encontrar caminhos viáveis e subótimos para o problema CVRP. Na segunda parte, linhas 23-34, usa as informações geradas na primeira parte para extrair as rotas do CVRP.

Inicialmente, no SPLIT-CVRP, as entradas do vetor P são inicializadas como ∞, exceto o primeiro elemento que é definido como nulo, nas linhas 1-2. O vetor P é usado para registrar o custo da rota sendo construída. Após isso, o algoritmo passa para dois laços aninhados que correspondem a primeira parte do SPLIT-CVRP. No laço externo, linha 3, o procedimento percorre do primeiro ao último elemento de T. Aqui, uma invariante é considerada no algo-

ritmo: uma rota gerada até o elemento $i-1$ é ótima independente de i, ou seja, a inclusão ou não de i na rota $i-1$ não impacta se essa rota é ótima ou não. No laço interno, linha 7, o procedimento tenta expandir a rota gerada até i com os elementos posteriores $i+1, \cdots, n$, desde que a inclusão desses elementos não torne a rota inviável. Assim, nesse laço interno, os nós $(n_i, n_{i+1}, \cdots, n_j)$ de T são analisados para verificar se $(0, n_i, n_{i+1}, \cdots, n_j, 0)$ é viável. Se sim, o caminho é modelado como um conjunto de arcos, usando o vetor P para armazenar o custo e o vetor W para armazenar os índices dos nós (linhas 16-17). Na segunda parte do SPLIT-CVRP, os valores dos vetores P e W são usados para extrair as rotas do CVRP. Essas rotas correspondem ao caminho de custo mínimo do nó 0 até o nó n em T. As informações dos índices armazenados em W são usadas para determinar o número e a sequência de cada rota. O algoritmo retorna como resultado o conjunto gerado de rotas.

O procedimento CONCAT é definido como o enfileiramento das rotas $r_0, r_1, \cdots, r_i$ como descrito a seguir. Seja $R = \{r_0, r_1, \cdots, r_i\}$ um conjunto de rotas em S, em que $r_k = (n_{k,1}, n_{k,2}, \cdots, n_{k,j_k})$, então o resultado do procedimento CONCAT é igual a $(n_{1,1}, n_{1,2}, \cdots, n_{1,j_1}, n_{2,1}, n_{2,2}, \cdots, n_{i,j_i-1}, n_{i,j_i})$. O pseudocódigo do procedimento CONCAT é mostrado no Algoritmo 24.

5.7 Método de Validação

O método VALIDO-CVRP é usado para impedir que o algoritmo construtivo HEURISTICA-CONSTRUTIVA crie soluções inviáveis para alguma restrição do problema CVRP. Em relação ao CVRP com soluções no espaço $\Omega(\mathrm{TSP})$, não há restrições e o método VALIDO-CVRP (Algoritmo 25) sempre retorna TRUE como resultado.

5.8 Método de Restrição

O método RESTRICAO-CVRP (Algoritmo 26) recebe duas rotas r e v de uma solução qualquer de $\Omega(\mathrm{VRP})$ e sempre retorna 0 (zero) como resultado.

5.9 Considerações Finais

Nesse capítulo foi apresentado o método MGV-C, uma matheurística com abordagem hierárquica, hibridizações e aplicação dos modelos matemáticos PCC e PPC para resolver problemas do tipo CVRP. A inclusão desses modelos permite explorar melhor o espaço de busca e gerar uma solução final que incorpore as melhores rotas geradas durante o GRASP.

Algoritmo 23: SPLIT-CVRP(T)

entrada: O *tour* $T \in \Omega(TSP)$
saída : Uma solução $S \in \Omega(VRP)$

$P_0 \leftarrow 0$;
$P_i \leftarrow +\infty, i = 1, 2, \cdots, n$;
para $i \leftarrow 1, \cdots, n$ **faça**
  $load \leftarrow 0$;
  $c \leftarrow 0$;
  $j \leftarrow i$;
  enquanto $j \leq n$ *and* $load \leq Q$ **faça**
    $load \leftarrow load + q_i$;
    se $i = j$ **então**
      $c \leftarrow c_{0,i} + c_{i,0}$;
    senão
      $c \leftarrow c - c_{j-1,0} + c_{j-1,j} + c_{j,0}$;
    fim
    se $load \leq Q$ **então**
      se $P_{i-1} + c < P_j$ **então**
        $P_j \leftarrow P_{i-1} + c$;
        $W_j \leftarrow i - 1$;
      fim
      $j \leftarrow j + 1$;
    fim
  fim
fim
$t \leftarrow 0$;
$j \leftarrow 0$;
$r_i \leftarrow \emptyset, i = 1, 2, \cdots, n$;
enquanto $i = 0$ **faça**
  $t \leftarrow t + 1$;
  $i \leftarrow W_j$;
  para $k \leftarrow i + 1, \cdots, j$ **faça**
    $r_i \leftarrow r_i \cup \{k\}$;
  fim
  $j \leftarrow i$;
fim
$S = \{r_1, \cdots, r_n\}$;
retorne S;

Algoritmo 24: CONCAT(S)

entrada: Uma solução $S \in \Omega(\text{VRP})$
saída : O *tour* $T \in \Omega(\text{TSP})$

```
1 T ← ∅;
2 para r ∈ S faça
3 |   para t ∈ r faça
4 |   |   T ← T ∪ {t};
5 |   fim
6 fim
7 retorne T;
```

Algoritmo 25: VALIDO-CVRP(m, T)

entrada: Um *tour* partial T em $\Omega(\text{TSP})$ e um movimento m sobre T
saída : TRUE ou FALSE

```
1 retorne TRUE;
```

Algoritmo 26: RESTRICAO-CVRP(r, v)

entrada: Duas rotas r e v de uma solução qualquer em $\Omega(VRP)$
saída : O número 0 ou 1

```
1 retorne 0;
```

6 Metodologia de Resolução do HRP com Matheurística

Neste capítulo é apresentado o método proposto para resolver os modelos HRPS e HRPM, chamado Matheurística GRASP+VNS para o HRP (MGV-H), a partir da adaptação do método MGV-C.

6.1 Introdução

A matheurística MGV-H é a adaptação do MGV-C para o HRPS e para o HRPM. Dado que HRPS e HRPM possuem diversas características comuns, neste capítulo usa-se a sigla HRP como referência a ambos modelos. Quando necessário, características exclusivas do HRPS ou do HRPM são ressaltadas no texto e nos algoritmos.

Nessa reformulação, as seguintes alterações são incorporadas no MGV-H:

- Soluções inviáveis são permitidas no espaço $\Omega(\mathrm{VRP})$. As inviabilidades permitidas se referem ao combustível máximo (Restrições 3.24), ao peso máximo (Restrições 3.22), ao tempo máximo de voo (Restrições 3.25), ao número máximo de assentos disponíveis (Restrições 3.18) e à janela de tempo (Restrições 3.30), esse último somente para o HRPM;
- A busca local opera em duas fases. Inicialmente, eliminam-se todas as inviabilidades da solução. Posteriormente, encontra-se um mínimo local da solução viável. Para isso, cada tipo de inviabilidade é tratada por vez. Por isso, é necessário que uma ordem de eliminação de inviabilidade seja definida;
- A função de *splitting* proposta é uma heurística gulosa, pois o procedimento de Bellman (GOLDBERG; RADZIK, 1993) não é aplicável a problemas de *pickup* e *delivery*. Nesse *splitting* proposto é permitido gerar soluções inviáveis;
- As restrições de precedência (Restrições 3.27) são sempre atendidas e não podem ser violadas em nenhuma parte do método;
- As soluções geradas pelos algoritmos construtivos e a busca local são rotas com uma única viagem. Para gerar soluções com múltiplas viagens, um procedimento específico é proposto;
- Dado que o HRP possui diversas requisições com mesma origem, mesmo destino e mesma janela de tempo, um procedimento de agrupamento de requisições é proposto para reduzir o tamanho do espaço de busca.

6.2 Método Proposto

O pseudocódigo do MGV-H é definido no Algoritmo 27. Ele recebe o grafo $G(N, E)$ representando o HRP, o número máximo de iterações M_i, o número $M_\alpha \in [0, 1]$ que controla o grau de aleatoriedade nas heurísticas construtivas, o número de iterações M_v do VNS, o número máximo de iterações M_s sem melhoria no VNS, o número máximo de viagens M_r numa rota, a estratégia de ordenação ϕ, um número real M_k não negativo que flexibiliza a restrição de tempo no *splitting*, a ordem M_γ que as restrições de interesse devem ser analisadas em S e o tamanho máximo M_c do agrupamento de nós. O resultado do algoritmo é uma solução $S \in \Omega(\mathrm{VRP})$.

Os parâmetros M_k, M_r, M_γ e M_c são específicos da adaptação MGV-H. O fator de tempo (M_k) define o percentual máximo de inviabilidade para a restrição de tempo no algoritmo *splitting*. O tamanho do agrupamento (M_c) restringe o tamanho máximo do agrupamento gerado. O objetivo do agrupamento é transformar um conjunto de nós com mesmas propriedades (origem, destino e janela de tempo) num único nó. Dessa forma, o espaço de busca é reduzido. O número máximo de viagens (M_r) é um parâmetro que define o número de viagens permitidas para cada rota. No caso do HRPS, esse parâmetro deve ser $M_r = 1$, pois nele é obrigatório que cada rota tenha somente uma viagem.

A busca local para o HRP não necessita gerar cada nova rota para avaliar o custo da nova solução, semelhante ao MGV-C. Isso permite reduzir o esforço computacional. Entretanto, usando essa estratégia, os movimentos que geram uma solução vizinha só podem avaliar o ganho ou a perda em relação a um tipo de restrição inviável. Por isso, uma ordem de restrições (M_γ) é definida. Assim, a busca local usa essa ordem de restrições para viabilizar uma solução.

De maneira geral, poucas linhas são incluídas ou alteradas no Algoritmo 27 do MGV-H. As inclusões são o método AGRUPAMENTO (linha 5) que agrupa os nós do grafo $G(N, E)$ e o bloco de controle para selecionar a solução final (linha 35). Além disso, um novo método chamado TRIP é usado para gerar uma solução no espaço $\Omega(\mathrm{VRP})$ com múltiplas viagens. Ele é usado na avaliação do custo da solução e para criar a solução final do método. As alterações realizadas são as substituições dos métodos VALIDO-CVRP, SPLIT-CVRP, BL-CVRP, RESTRICAO-CVRP pelos equivalentes VALIDO-HRP, SPLIT-HRP, BL-HRP, RESTRICAO-HRP do HRP.

Note, também, que o Algoritmo 27 utiliza os espaços $\Omega(\mathrm{TSP})$ e $\Omega(\mathrm{VRP})$. Apesar de ser uma aplicação no HRP, manteve-se a designação das soluções do HRP como $\Omega(\mathrm{VRP})$. A seguir, os novos métodos são apresentados em maiores detalhes. Entretanto, antes definem-se novas convenções e representações adotadas para o MGV-H.

Algoritmo 27: MGV-H(G, M_i, M_α, M_v, M_s, M_r, ϕ, M_k, M_γ, M_c)

entrada: Grafo $G(N, E)$, o número máximo M_i de iterações do GRASP, um número $M_\alpha \in [0, 1]$, o número máximo M_v de iterações do VNS, o número máximo M_s de iterações sem aprimoramento, o número máximo de viagens M_r para cada rota, estratégia de ordenamento ϕ, um número real M_k não negativo, estratégia da buscal local M_γ e tamanho máximo M_c do agrupamento

saída : Uma solução $S \in \Omega(\text{VRP})$

$i_c \leftarrow 0$;
$\pi_{PCC} \leftarrow \emptyset$;
$\pi_{PPC} \leftarrow \emptyset$;
$\bar{S} \leftarrow \emptyset$;
$G \leftarrow \text{AGRUPAMENTO}(G, M_c)$;
para $i = 0, \cdots, M_i$ **faça**
  $T \leftarrow \text{PCC-SOLUCAO}(G, \pi_{PCC}, \phi)$;
  $T \leftarrow \text{HEURISTICA-CONSTRUTIVA}(G, T, M_\alpha, \text{VALIDO-HRP})$;
  $S \leftarrow \text{SPLIT-HRP}(T, M_k)$;
  $S \leftarrow \text{BL-HRP}(S)$;
  $\pi_S \leftarrow \emptyset$;
  para $k = 1, \cdots, M_v$ **faça**
    $i_c \leftarrow i_c + 1$;
    $T^* \leftarrow \text{CONCAT}(S)$;
    $T^* \leftarrow \text{PERTUBA-HRP}(T^*)$;
    $S^* \leftarrow \text{SPLIT-HRP}(T^*, M_k)$;
    $S^* \leftarrow \text{BL-HRP}(S, M_\gamma)$;
    $\pi_S \leftarrow \pi_S \cup \{S^*\}$;
    se $\text{cost}(\text{TRIP}(S^*, M_r)) < \text{cost}(\text{TRIP}(S, M_r))$ **então**
      $S \leftarrow S^*$;
    fim
    se $\text{cost}(\text{TRIP}(S^*, M_r)) < \text{cost}(\text{TRIP}(\bar{S}, M_r))$ **então**
      $i_c \leftarrow 0$;
      $\bar{S} \leftarrow S$;
    fim
  fim
  $\pi_{PCC} \leftarrow \pi_{PCC} \cup \pi_S$;
  $\pi_{PPC} \leftarrow \pi_{PPC} \cup \pi_S$;
  se $i_c \geq M_s$ **então**
    $\pi_{PCC} \leftarrow \emptyset$;
    $i_c \leftarrow 0$;
  fim
fim
$S \leftarrow \text{PPC-SOLUCAO}(G, \pi_{PPC}, \text{RESTRICAO-HRP})$;
se $\text{cost}(\text{TRIP}(S, M_r)) < \text{cost}(\text{TRIP}(\bar{S}, M_r))$ **então**
  $\bar{S} \leftarrow S$;
fim
retorne $\text{TRIP}(\bar{S}, M_r)$;

6.3 Convenções e Representação das Soluções

O HRP é um problema com requisições de *pickup* e de *delivery*. Por isso, é necessário o acréscimo de novas convenções às definidas na Seção 4.2. Dado um grafo $G(N, E)$ retratando um rede de transporte do HRP, define-se os seguintes subconjuntos de N. Os conjuntos $N_p, N_d \subset N$ designam os nós de *pickup* e de *delivery*. Os conjuntos $N_p^0 \subset N_p$ e $N_d^0 \subset N_d$ designam os nós de *pickup* e *delivery* realizados no aeroporto, respectivamente. Além disso, dada uma requisição $r = (a, a')$, ela é dita uma requisição *pickup-delivery*, em que $a \in N_p$ e $a' \in N_d$.

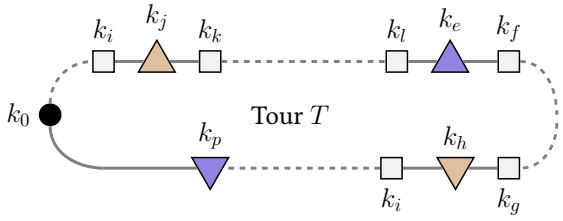
Nos problemas com *pickup* e *delivery* é necessário que as soluções atendam às restrições de precedência, ou seja, dada uma requisição (a, a'), o nó de *pickup* a deve estar localizado antes do nó a' em qualquer solução. Essas restrições de precedências são sempre atendidas nos espaços de solução $\Omega(\mathrm{TSP})$ e $\Omega(\mathrm{VRP})$. Entretanto, os problemas HRP com atendimento no aeroporto demandam o atendimento dos nós *pickup* no aeroporto (conjunto N_p^0) e nós *delivery* no aeroporto (conjunto N_d^0) no início e no fim da rota, respectivamente. No espaço de solução $\Omega(\mathrm{TSP})$, essa exigência é relaxada e os nós dos conjuntos N_p^0 e N_d^0 podem ocorrer em qualquer posição de $T \in \Omega(\mathrm{TSP})$. Para uma solução $S \in \Omega(\mathrm{VRP})$, as rotas $r_i \in S$ devem seguir a seguinte sequência de nós:

$$r_i = \left[k_0, k_1^0, k_2^0, \cdots, k_i^0, k_{i+1}, k_{i+2} \cdots, k_j, k_{j+1}^0, k_{j+2}^0, \cdots, k_h^0, k_0\right]$$

em que $k_1^0, k_2^0, \cdots, k_i^0 \in N_p^0$ são os nós de *pickup* no aeroporto e devem ocorrer logo após o nó inicial k_0, os nós internos $k_{i+1}, k_{i+2} \cdots, k_j \in (N_p \cup N_d) \setminus (N_p^0 \cup N_d^0)$ designando o atendimento em plataformas e os nós $k_{j+1}^0, k_{j+2}^0, \cdots, k_h^0 \in N_d^0$ designando o *delivery* executado no aeroporto.

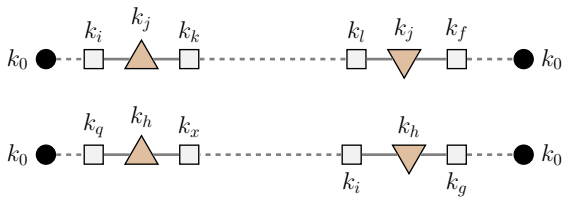
Em relação à representação gráfica dos nós *pickup* e *delivery* para a requisição (a, a'), o nó de *pickup* é representado como um triângulo com ápice para cima, enquanto o nó de *delivery* é desenhado como uma triângulo com ápice para baixo. Caso não seja necessária a informação do tipo de nó, ele é representado como um quadrado. A Figura 13 mostra um exemplo de *tour* para o HRP. Nela, são representados os pares de nós *pickup-delivery* (k_j, k_h) e (k_e, k_p). De forma semelhante, a Figura 14 apresenta o caso de uma solução no espaço $\Omega(\mathrm{VRP})$.

Figura 13 – Representação gráfica do *tour* T no espaço $\Omega(\mathrm{TSP})$ para o HRP.



Além disso, para o HRPM, é permitido que uma rota tenha mais de uma viagem. No espaço de solução $\Omega(\mathrm{VRP})$, um rota $r = r_1 \cup r_2 \cup \cdots \cup r_k$ com k viagens é representada como

Figura 14 – Representação gráfica da solução S no espaço $\Omega(\mathrm{VRP})$ para o HRP.



a concatenação das várias rotas de viagem única $r_1, r_2, \cdots r_k$ em que se mantém o nó garagem (k_0) internamente para demarcar o início e o fim de cada viagem. Assim, as rotas r_1 e r_2 com viagem única

$$r_1 = [k_0, k_1, k_2, k_0]$$

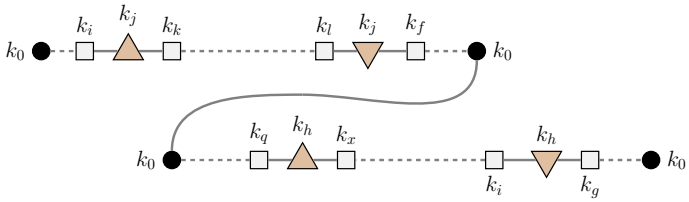
$$r_2 = [k_0, k_3, k_4, k_0]$$

podem ser utilizadas para formar a rota $r_3 = r_1 \cup r_2$, a qual é definida como:

$$r_3 = [k_0, k_1, k_2, k_0, k_0, k_3, k_4, k_0]$$

Note que na representação das rotas com várias viagens a repetição dos nós garagens k_0, k_0 delimita o fim de uma viagem e o início da próxima viagem. A Figura 15 mostra um exemplo de rota com duas viagens para o HRP formada a partir das rotas da Figura 14.

Figura 15 – Representação gráfica da solução S no espaço $\Omega(\mathrm{VRP})$ para o HRP com várias viagens.



6.4 Representação Computacional

De forma semelhante à matheurística MGV-C, o procedimento MGV-H também utiliza uma representação de solução como vetores de inteiros. Em relação ao espaço $\Omega(\mathrm{TSP})$, não há diferença entre MGV-C e MGV-H. Entretanto, em relação ao espaço $\Omega(\mathrm{VRP})$, novos vetores são definidos. De maneira análoga, esses vetores são utilizados para diminuir o tempo dispendido na busca local BL-HRP (Algoritmo 31).

Seja $S = \{r_1, s_2, \cdots, s_k\}$ uma solução com k rotas no espaço $\Omega(\mathrm{VRP})$ e $G(N, E)$ o grafo do problema HRP. Então para cada rota $r_i \in S$ são definidos os vetores auxiliares para o número máximo de assentos utilizados ($Q^{\mathrm{s}}_{r_i}$), o peso máximo utilizado ($Q^{\mathrm{p}}_{r_i}$), o combustível máximo disponível ($Q^{\mathrm{f}}_{r_i}$), folga mínima no atendimento da janela inferior ($Q^{\mathrm{L}}_{r_i}$), folga mínima no atendimento da janela superior ($Q^{\mathrm{U}}_{r_i}$).

O vetor auxiliar de número máximo de assentos utilizados ($Q^{\mathrm{s}}_{r_i}$) registra na posição i o maior número de assentos utilizados entre o início da rota e a posição i:

$$Q^{\mathrm{s}}_{r_i}[j] = \begin{cases} 0 & \text{se } r_i[j] = k_0 \\ max\left\{Q^{\mathrm{s}}_{r_i}[j-1],\ \text{assentos utilizados em } r_i[j]\right\} & \text{se } r_i[j] \neq k_0 \end{cases}$$

O vetor auxiliar de peso máximo ($Q^{\mathrm{p}}_{r_i}$) registra na posição i o maior peso registrado entre o início da rota e a posição i:

$$Q^{\mathrm{p}}_{r_i}[j] = \begin{cases} \text{peso em } k_0 & \text{se } r_i[j] = k_0 \\ max\left\{Q^{\mathrm{p}}_{r_i}[j-1],\ \text{peso em } r_i[j]\right\} & \text{se } r_i[j] \neq k_0 \end{cases}$$

O vetor auxiliar de combustível máximo ($Q^{\mathrm{p}}_{r_i}$) registra na posição i o máximo de combustível disponível entre o início da rota e a posição i:

$$Q^{\mathrm{f}}_{r_i}[j] = \begin{cases} \text{combustível em } k_0, & \text{se } r_i[j] = k_0 \\ max\left\{Q^{\mathrm{f}}_{r_i}[j-1],\ \text{combustível disponível em } r_i[j]\right\}, & \text{se } r_i[j] \neq k_0 \end{cases}$$

O vetor auxiliar de folga mínima no atendimento da janela inferior ($Q^{\mathrm{L}}_{r_i}$) registra na posição i a menor folga entre a posição i e o final da rota:

$$Q^{\mathrm{L}}_{r_i}[j] = \begin{cases} T_{r_i[j]} - L_{r_i[j]}, & \text{se } r_i[j] \text{ é o nó aeroporto no final da rota} \\ min\left\{Q^{\mathrm{L}}_{r_i}[j+1], T_{r_i[j]} - L_{r_i[j]}\right\}, & \text{caso contrário} \end{cases}$$

em que $T_{r_i[j]}$ representa o horário de atendimento do nó $r_i[j]$ na solução S e $L_{r_i[j]}$ é a janela de atendimento inferior do nó $r_i[j]$.

O vetor auxiliar de folga mínima no atendimento da janela superior ($Q^{\mathrm{U}}_{r_i}$) registra na posição i a menor folga entre a posição i e o final da rota:

$$Q^{\mathrm{U}}_{r_i}[j] = \begin{cases} U_{r_i[j]} - T_{r_i[j]}, & \text{se } r_i[j] \text{ é o nó aeroporto no final da rota} \\ min\left\{Q^{\mathrm{U}}_{r_i}[j+1], U_{r_i[j]} - T_{r_i[j]}\right\}, & \text{caso contrário} \end{cases}$$

em que $U_{r_i[j]}$ é a janela de atendimento superior do nó $r_i[j]$.

Note que os vetores de número máximo de assentos utilizados ($Q^{\mathrm{s}}_{r_i}$), do peso máximo ($Q^{\mathrm{p}}_{r_i}$) e de combustível máximo ($Q^{\mathrm{f}}_{r_i}$) são construídos recursivamente a partir do início da rota, enquanto os vetores de folgas da janela ($Q^{\mathrm{L}}_{r_i}$ e $Q^{\mathrm{U}}_{r_i}$) são construídos recursivamente a partir do final da rota.

6.5 Métodos de Transformação de Espaço

O Algoritmo 28 apresenta o método SPLIT-HRP que faz o *split* do *tour* $T \in \Omega(\mathrm{TSP})$ num conjunto de rotas em $\Omega(\mathrm{VRP})$. Dado que o problema de definir a divisão e a sequência

de *pickups* e *deliveries* não pode ser resolvido de maneira eficiente pelo algoritmo de Bellman (DESROSIERS; DUMAS, 1988; BOUROS et al., 2011), então uma heurística gulosa baseada no SPLIT-CVRP do CVRP é proposta. O procedimento recebe como parâmetro o *tour* $T \in \Omega(\text{TSP})$ e o número não negativo k. Ele retorna como resultado uma solução $S \in \Omega(\text{VRP})$.

O SPLIT-HRP opera somente com duas restrições dos modelos matemáticos HRP na geração das rotas. A primeira restrição atendida é a de precedência de *pickups* e *deliveries* (Restrições 3.27), ou seja, caso um *pickup* seja feito na rota r, então o seu *delivery* também é feito nessa rota, sempre, após o *pickup*. A segunda restrição usada em SPLIT-HRP é a restrição de tempo máximo de voo (Restrições 3.25). Entretanto, nesse caso, o procedimento permite que essa restrição seja relaxada em k por cento. O restante das restrições do modelo relacionadas a peso, ao número de assentos, ao número máximo de combustível e à janelas de tempo são completamente relaxadas.

De forma geral, o SPLIT-HRP é um algoritmo recursivo. Inicialmente, nas linhas 1-2, os nós *pickup* e *delivery* no aeroporto $(N_p^0 \cup N_d^0)$ são removidos de T. Isso é feito para permitir que rotas maiores sejam construídas, dado que os nós em $(N_p^0 \cup N_d^0)$ devem estar sempre no começo ou no final da rota. Após, na linha 5, é verificado se T é vazio. Se sim, o resultado também é uma solução vazia. Caso contrário, o algoritmo entra em dois laços aninhados (linhas 8-17). O objetivo desse trecho é encontrar todos os *subtours* P de T que atendam as restrições de precedência e de tempo máximo de voo relaxada em k por cento (linha 13). Antes de avaliar se P é viável, os nós *pickup* e *delivery* no aeroporto necessários em P são readicionados nas linhas 11 e 12. Assim, cada ocorrência de P é salva no conjunto L de *subtours* de T. Após o término desses dois laços, é verificado se há, pelo menos, um *subtour* em L (linha 18). Em caso positivo, então uma rota R é gerada a partir do maior *subtour*, em termos de número de nós, de L. Essa rota é inserida no conjunto de rotas S, o *subtour* usado é removido de T e o procedimento é chamado outra vez com o novo T para gerar o restante das rotas (linhas 19-21) Entretanto, se não há *subtour* válido (linha 23), ou seja, $L = \emptyset$, então o procedimento HALVE é chamado para dividir T em dois *tours* T_1 e T_2 de tamanhos similares (linha 24). Após isso, o procedimento SPLIT-HRP é chamados duas vezes, um para T_1 e outra vez para T_2 (linha 25), para gerar as rotas em função de T_1 e T_2.

O Algoritmo 29 apresenta o procedimento HALVE usado por SPLIT-HRP para a divisão de um rota T em duas rotas T_1 e T_2 de tamanhos similares. O procedimento recebe uma solução $T \in \Omega(\text{TSP})$ e retorna duas soluções em $\Omega(\text{TSP})$. A ideia base do procedimento é dividir *pickups* e *deliveries* em dois *subtours* T_1 e T_2, respeitando a restrição de precedência (Restrições 3.27).

O procedimento HALVE itera num laço de tamanho igual a $size(T)$ (linhas 4-35). Nas primeiras h iterações, o objetivo é preencher T_1, enquanto no restante das iterações T_2 é construído, majoritariamente. No bloco referente a T_1 (linhas 6-19), todos os nós são incluídos em T_1. Entretanto, é necessário controlar se está sendo feito um *pickup* ou um *delivery*, pois

Algoritmo 28: SPLIT-HRP(T, M_k)

entrada: Uma solução $T \in \Omega(\text{TSP})$ e um número M_k não negativo.
saída : Uma solução $S \in \Omega(\text{VRP})$.

```
1  Remova nós de pickup no aeroporto (N_p^0) de T;
2  Remova nós de delivery no aeroporto (N_d^0) em T;
3  S ← ∅;
4  L ← ∅;
5  se T = ∅ então
6  |  retorne ∅;
7  fim
8  para k = 1, ··· , size(T) faça
9  |  para l = k, ··· , size(T) faça
10 |  |  P ← T[k, ··· , l];
11 |  |  Adicione, se necessário, os nós de pickup (N_p^0) no aeroporto em T;
12 |  |  Adicione, se necessário, os nós de delivery (N_d^0) no aeroporto em T;
13 |  |  se time(P) ≤ f^k * (1 + M_k) e T atende as restrições de precedência então
14 |  |  |  L ← L ∪ {P};
15 |  |  fim
16 |  fim
17 fim
18 se L ≠ ∅ então
19 |  R ← obtenha o subtour de maior tamanho em L;
20 |  Remova os nós em R de T;
21 |  S ← {R} ∪ SPLIT-HRP(T, M_k);
22 fim
23 senão
24 |  T_1, T_2 ← HALVE(T);
25 |  S ← SPLIT-HRP(T_1, M_k) ∪ SPLIT-HRP(T_2, M_k);
26 fim
27 retorne S;
```

o *delivery* de um *pickup* pode acontecer além das primeiras h iterações. Para isso, nas linhas 7-12, é verificado se o nó corrente a ser incluído em T_1 é *pickup* e se o seu *delivery* não é feito no aeroporto. Em caso positivo, a variável de controle h é diminuída de uma unidade (linha 10), supondo que o seu *delivery* correspondente está além das primeiras h iterações. Essa suposição pode ser revertida nas linhas 13-18, em que para cada *delivery* encontrado no intervalo reservado a T_1, a variável h é incrementada de uma unidade (linha 16). Assim, caso k torne-se maior que h com $h < size(T)/2$ (linha 6), significa que existem ainda $size(T)/2 - h$ nós *delivery* que precisam ser incluídos em T_1, mas que só serão encontrados quando for analisado o trecho correspondente ao preenchimento de T_2.

Por fim, o procedimento HALVE preenche o *subtour* T_2 nas linhas 21-34. Nesse trecho, é verificado se o nó é um *delivery* (linha 22). Em caso positivo, é necessário verificar se o *pickup* está em T_1. Se esse for o caso, então o nó corrente deve ser incluído em T_1. Em todos os outros

casos, o nó corrente deve ser incluído em T_2.

Algoritmo 29: HALVE(T)

entrada: O *tour* $T \in \Omega(\text{TSP})$.
saída : Dois *tours* $T_1, T_2 \in \Omega(\text{TSP})$.

$T_1 \leftarrow \emptyset$;
$T_2 \leftarrow \emptyset$;
$h \leftarrow \text{size(T)}\,/2$;
para $k = 1, \cdots, size(T)$ **faça**
  $s \leftarrow T[k]$;
  se $k < h$ **então**
    se s *é um nó pickup* **então**
      $d \leftarrow$ *delivery* de s;
      se d *não é delivery no aeroporto* **então**
        $h = h - 1$;
      fim
    fim
    se s *é um nó delivery* **então**
      $p \leftarrow$ *pickup* de s;
      se p *não é pickup no aeroporto* **então**
        $h = h + 1$;
      fim
    fim
    $T_1 \leftarrow T_1 \cup \{s\}$;
  fim
  senão
    se s *é um nó delivery* **então**
      $p \leftarrow$ *pickup* de s;
      se $p \in T_1$ **então**
        $T_1 \leftarrow T_1 \cup \{s\}$;
      fim
      senão
        $T_2 \leftarrow T_2 \cup \{s\}$;
      fim
    fim
    senão
      $T_2 \leftarrow T_2 \cup \{s\}$;
    fim
  fim
fim
retorne $[T_1, T_2]$;

6.6 Heurística de Busca Local

A busca local proposta para o MGV-H opera com soluções inviáveis. As seguintes restrições, denominadas de restrições de interesse, do modelo HRPS e HRPM 3.8-3.42 podem ser

inviáveis numa solução corrente da busca local:

- Combustível máximo (Restrições 3.24);
- Peso máximo (Restrições 3.22);
- Tempo máximo de voo (Restrições 3.25);
- Número máximo de assentos (Restrições 3.18);
- Janela de tempo (Restrições 3.30).

O objetivo da busca local é, então, remover as inviabilidades e reduzir o custo da solução, encontrando uma solução viável e localmente ótima. Isso é realizado em dois momentos distintos. Inicialmente, os movimentos da busca local reduzem o número de inviabilidades numa solução $S \in \Omega(\mathrm{VRP})$. Nessa fase, também, pode ser reduzida a distância total percorrida na solução S ou o número de aeronaves usadas. Finalizada essa parte, a solução S é viável, mas não necessariamente é uma solução localmente ótima. Então, nesse ponto, começa a segunda fase. Nesse momento, a busca local realiza movimentos que reduzam a distância percorrida ou o número de aeronaves, sem a possibilidade de incluir inviabilidades na solução S.

No processo de redução do número de nós com inviabilidades para as restrições de interesse, uma ordem de preferência é definida previamente como parâmetro. O objetivo é que as inviabilidades sejam removidas da solução S por tipo de restrição de interesse. Para isso, seja o conjunto finito Υ designando as restrições de interesse para o HRPS ou HRPM. O conjunto Υ possui os seguintes elementos:

- $\upsilon_F \in \Upsilon$, elemento que representa a restrição de combustível no HRP;
- $\upsilon_S \in \Upsilon$, elemento que representa a restrição de capacidade de assentos no HRP;
- $\upsilon_T \in \Upsilon$, elemento que representa a restrição de tempo no HRP;
- $\upsilon_W \in \Upsilon$, elemento que representa a restrição de peso no HRP;
- $\upsilon_J \in \Upsilon$, elemento que representa a restrição de janela de tempo no HRPM.

A partir do conjunto Υ, define-se o vetor M_υ, que determina a ordem em que as restrições de interesse são viabilizadas. Para cada elemento $\gamma \in M_\upsilon$, os movimentos da busca local minimizam a quantidade de nós ou magnitude das inviabilidades de cada nó. Em relação ao restante das restrições, os movimentos não podem tornar um nó viável inviável.

Na matheurística MGV-C para o CVRP três movimentos são utilizados na busca local: 2-OPT, CROSSOVER e OR-OPT. No caso de problemas de *pickup* e *delivery*, a utilização

clássica do 2-OPT e do CROSSOVER tem pouca aplicabilidade, pois gera soluções inviáveis com frequência para a restrição de precedência. Assim, na adaptação do MGV-H utiliza-se somente uma versão modificada do OR-OPT chamada OR-OPT-HRP. Nessa variante, o movimento busca remover pares de nós *pickup-delivery* de uma rota r_i e reinseri-los noutra rota r_j.

O Algoritmo 30 mostra o movimento OR-OPT-HRP adaptado para o problema HRP. Ele recebe uma solução $S \in \Omega(\mathrm{VRP})$ e uma restrição para viabilizar γ. No laço externo (linha 1), todos os pares de rotas r_i, r_j de S são selecionados para testar o movimento. O objetivo é selecionar um par de nós de *pickup-delivery* de r_i para inseri-los em r_j. No primeiro laço interno (linha 2), percorre-se os pares de *pickup-delivery* (a, a') de r_i. A partir disso, no laço mais interno (linha 3), testa-se a inserção de (a, a') em cada posição possível de r_j. A função $\mathrm{BL\text{-}HRP\text{-}CUSTO}$ verifica se essa reinserção de nós melhora a qualidade da solução (linha 9), seja reduzindo o número de inviabilidades ou diminuindo o custo da solução. Se esse for o caso, então o movimento é realizado e a função retorna TRUE. Veja que nesse caso, o movimento OR-OPT-HRP escolhe sempre o primeiro movimento que melhore a solução atual. Caso nenhum movimento se encaixe nesse critério, a função retorna FALSE após a execução de todos as iterações do laço externo.

A função $\mathrm{BL\text{-}HRP\text{-}CUSTO}$ é a função que, utilizando os vetores auxiliares apresentados na Seção 6.4, determina se a reinserção de um par de nós *pickup-delivery* gera uma solução vizinha melhor que a corrente. Em geral, o objetivo é garantir que a magnitude da inviabilidade diminua considerando as duas rotas utilizadas no movimento ou, caso não tenha nenhum nó inviável, que o movimento reduza o custo do roteamento. Os detalhes da função $\mathrm{BL\text{-}HRP\text{-}CUSTO}$ podem ser encontrados no Apêndice D.

No Algoritmo 31 é apresentado o pseudocódigo da busca local BL-HRP. Ele recebe uma solução S em $\Omega(\mathrm{VRP})$ e um vetor M_γ definindo a sequência na qual as restrições são viabilizadas durante a busca local. O resultado do procedimento é uma solução S localmente ótima. De maneira geral, o procedimento aplica, repetidamente, o procedimento OR-OPT-HRP (Algoritmo 30) até encontrar um mínimo local para S. Inicialmente, o conjunto θ é inicializado para registrar quais restrições já foram ou estão sendo viabilizadas (linha 1). A partir daí, inicia um laço externo (linha 2) que representa as iterações sobre o vetor de restrições M_γ. No laço interno, linha 5, aplica-se o procedimento OR-OPT-HRP até que a solução corrente S não tenha nenhuma inviabilidade em relação às restrições em θ. Entretanto, pode acontecer que o movimento realizado pelo OR-OPT-HRP não seja suficiente para viabilizar a restrição em consideração. Se esse for caso, as rotas $r \in S$ que tenham alguma restrição em θ violada são divididas em duas novas rotas. Para isso, nas linhas 10-14, transforma-se a rota $r \in S$ numa solução T, a qual é aplicada o procedimento HALVE (Algoritmo 29) e o procedimento SPLIT-HRP (Algoritmo 28). Isso gera duas novas rotas que são incluídas na solução S e a exclusão de r de S. O algoritmo opera assim até que todas as restrições em S sejam viáveis. Nesse

Algoritmo 30: OR-OPT-HRP(S, γ)

entrada: Uma solução $S \in \Omega(\text{VRP})$ e a restrição para viabilizar γ
saída : O valor verdade TRUE ou FALSE

```
1  para cada r_i, r_j ∈ S faça
2      para cada par pickup-delivery (a, a') de r_i faça
3          para cada nó b, c de r_j = [k_0, ··· , b, ··· , c, ··· , k_0] faça
4              Registre o número máximo de assentos entre b e c;
5              Registre o peso máximo entre b e c;
6              se b ocorre após c em r_j então
7                  Vá para o próximo par de nós do laço;
8              fim
9              se BL-CVRP-CUSTO(r_i*, r_j*, a, a', b, c, γ) < 0 então
10                 Remova a e a' de r_i;
11                 Insira a imediatamente antes de b em r_j;
12                 Insira a' imediatamente antes de c em r_j;
13                 retorne TRUE
14             fim
15         fim
16     fim
17 fim
18 retorne FALSE;
```

momento, futuras aplicações do OR-OPT-HRP somente podem reduzir o custo da solução, seja em relação ao número de helicópteros da solução ou a distância percorrida. Uma vez que OR-OPT-HRP não consiga gerar nenhuma solução vizinha melhor, tanto o laço interno quanto externo terminam. O resultado é a solução localmente ótima S.

6.7 Heurística de Pertubação

A heurística de pertubação dos modelos HRP é similar a do CVRP (Seção 5.5) com duas classes de movimentos: SWAP-HRP e REINSERTION-HRP. Entretanto, diferente do CVRP, nos modelos HRP é necessário considerar a restrição de precedência de *pickups* e *deliveries* nesses movimentos. Por isso, os movimentos são realizados sempre usando um ou dois pares de nós *pickup-delivery*.

O Algoritmo 32 apresenta o procedimento SWAP-HRP que realiza a permuta entre dois pares de nós *pickup-delivery*. Ele recebe uma solução $T \in \Omega(\text{TSP})$. Nas linhas 1-4, dois pares distintos de *pickup-delivery* são selecionados em T. Em seguida, nas linhas 5-6, os pares de nós são permutados em T.

Na Figura 16 é mostrado um exemplo do procedimento SWAP-HRP. Nesse esquema, dois pares de nós *pickup-delivery* (a, a') e (b, b') são trocados em T. Note que no esquema os nós (a, a') encontram-se antes de (b, b'), mas isso não é uma condição do procedimento.

Algoritmo 31: BL-HRP(S, M_γ)

entrada: Uma solução S em $\Omega(\text{VRP})$ e a ordem de restrições M_γ.
saída : Uma solução S em $\Omega(\text{VRP})$.

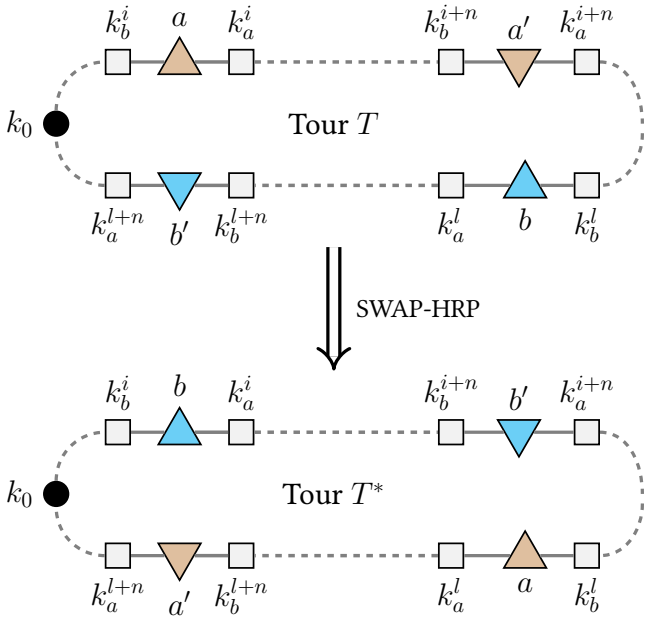
$\theta \leftarrow \emptyset$;
para cada $\upsilon \in M_\gamma$ **faça**
  $it \leftarrow$ TRUE;
  $\theta \leftarrow s \cup \{\upsilon\}$;
  enquanto $it =$ TRUE **faça**
    $r \leftarrow$ OR-OPT-HRP(S, γ);
    se S *é inviável para alguma restrição em* θ *e* $r =$ FALSE **então**
      $h \leftarrow \emptyset$;
      para cada *rota* $r \in S$ *inviável para restrições em* θ **faça**
        Remova r de S;
        $T \leftarrow \text{CONCAT}(\{r\})$;
        $[T_1, T_2] \leftarrow \text{HALVE}(T)$;
        $h \leftarrow h \cup \text{SPLIT-HRP}(T_1)$;
        $h \leftarrow h \cup \text{SPLIT-HRP}(T_2)$;
      fim
      $S \leftarrow S \cup h$;
    fim
    senão
      $it \leftarrow$ false;
    fim
  fim
fim
retorne S;

Algoritmo 32: SWAP-HRP(T)

entrada: O *tour* $T \in \Omega(\text{TSP})$
saída : O *tour* $T \in \Omega(\text{TSP})$

Selecione um nó *pickup* a, aleatoriamente, de T;
Selecione um nó *pickup* b, aleatoriamente, de T em que $b \neq a$;
$a' \leftarrow$ *delivery* de a;
$b' \leftarrow$ *delivery* de b;
Permute os nós a e b em T;
Permute os nós a' e b' em T;
retorne T;

Figura 16 – Pertubação via SWAP-HRP.



O Algoritmo 33 apresenta o procedimento REINSERTION-HRP que realiza a reinserção de um par de nós *pickup-delivery* em T. Ele recebe uma solução $T \in \Omega(\text{TSP})$. Na linha 1, um par de nós (a, a') de *pickup* e *delivery* são selecionados. Em seguida, linha 2, uma nova posição para o *pickup* a é escolhida aleatoriamente e, na linha 3, a é removido de T e inserido na sua nova posição. Após isso, na linha 4, uma nova posição para o *delivery* a' é escolhida após o nó *pickup* a em T, no qual a' é inserido (linha 5). O procedimento retorna a solução alterada T como resultado.

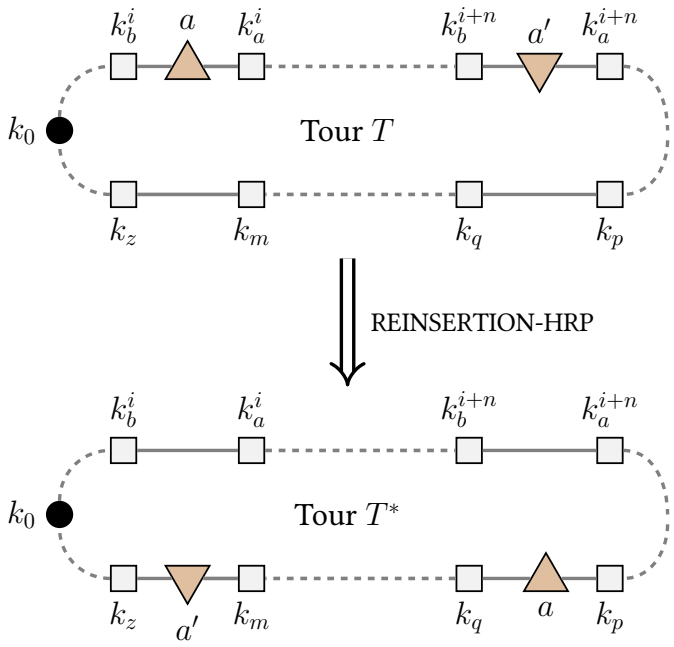
Algoritmo 33: REINSERTION-HRP(T)

entrada: O *tour* $T \in \Omega(\text{TSP})$
saída : O *tour* $T \in \Omega(\text{TSP})$
1 Selecione o par (a, a') de *pickup-delivery* aleatoriamente de T;
2 Selecione, aleatoriamente, dois nós adjacentes $\{k_p, k_q\}$, tal que $a, a' \notin \{k_p, k_q\}$, em T;
3 Remova a de T e o insira entre os nós $\{k_p, k_q\}$;
4 Selecione, aleatoriamente, dois nós adjacentes $\{k_z, k_m\}$ após a nova posição de a em T ;
5 Remove a' de T e o insira entre os nós $\{k_z, k_m\}$;
6 **retorne** T;

Na Figura 17 é mostrado um exemplo do procedimento REINSERTION-HRP. Nessa representação, um par de nós *pickup-delivery* (a, a') são reinseridos em T. Note que no esquema os nós (a, a') aparecem antes de (k_p, k_q) e (k_z, k_m), mas isso não é uma condição do procedimento.

O Algoritmo 34 apresenta o pseudocódigo PERTUBA-HRP para o MGV-H. Ele recebe uma solução $T \in \Omega(\text{TSP})$ e retorna uma nova solução $T \in \Omega(\text{TSP})$ com alguma pertubação

Figura 17 – Pertubação via REINSERTION-HRP.



aplicada. Nesse procedimento, escolhe-se, aleatoriamente, a aplicação de um REINSERTION-HRP ou um SWAP-HRP em T. Então, a solução alterada T é retornada como solução final.

Algoritmo 34: PERTUBA-HRP(T)

entrada: O *tour* $T \in \Omega(\text{TSP})$
saída : O *tour* $T \in \Omega(\text{TSP})$

1 Selecione o operador de pertubação m, aleatoriamente, de $\{\text{SWAP-HRP}, \text{REINSERTION-HRP}\}$;
2 $T^* \leftarrow m(T)$;
3 **retorne** T^*;

6.8 Método de Validação

O método VALIDO-HRP (Algoritmo 35) impede que o algoritmo construtivo HEURISTICA-CONSTRUTIVA crie soluções inviáveis para alguma restrição do HRP. Em relação a isso, com soluções no espaço $\Omega(\text{TSP})$, a única exigência é que a restrição de precedência de *pickup-delivery* sempre seja atendida.

6.9 Método de Restrição

O método RESTRICAO-HRP (Algoritmo 36) recebe duas rotas r e v de uma solução qualquer de $\Omega(\text{VRP})$. Se as rotas r e v são relativas ao problema HRPM e elas atendem a mesma plataforma t com um intervalo de tempo entre elas menor ou igual ao tempo de segurança

Algoritmo 35: VALIDO-HRP(m, T)

entrada: Um *tour* partial T em $\Omega(\text{TSP})$ e um movimento m sobre T
saída : TRUE ou FALSE

```
1 T* ← T;
2 Aplique m a T* ;
3 para cada par de nós pickup-delivery (a, a') em T* faça
4     se nó a' é anterior ao nó a em T* então
5         retorne FALSE ;
6     fim
7 fim
8 retorne TRUE ;
```

para plataformas, então o método retorna o valor igual a 1. Como consequência, na geração da matriz A do procedimento PPC-SOLUCAO (Algoritmo 14) uma linha é incluída impedindo que as rotas r e v participem simultaneamente da solução do modelo PPC. No caso de rotas r e v relativas ao problema HRPS, o procedimento sempre retorna um valor igual a 0, pois não existe restrição relacionada a pouso seguro no HRPS.

Algoritmo 36: RESTRICAO-HRP(r, v)

entrada: Duas rotas r e v de uma solução qualquer em $\Omega(\text{VRP})$
saída : O número 0 ou 1

```
1 se O problema pertence ao modelo HRPM então
2     se v e r atendem a mesma plataforma com intervalo de tempo menor ou igual ao
        tempo de segurança e v ≠ r então
3         retorne 1
4     fim
5 fim
6 retorne 0;
```

6.10 Método de Agrupamento

O método AGRUPAMENTO (Algoritmo 37) recebe o grafo $G(N, E)$ representando o HRP e o tamanho máximo do agrupamento M_c. Inicialmente, o procedimento cria janelas de tempo artificiais caso G represente um problema de HRPS (linha 2), visto que o procedimento espera que exista janela de tempo para cada nó. Em seguida, inicia-se o laço principal do algoritmo que realiza o agrupamento das requisições (linha 5). O objetivo aqui é aglutinar requisições a partir de grupos com, no máximo, M_c elementos. Para fazer isso, é necessário que essas requisições tenham a mesma origem, o mesmo destino e a mesma janela de tempo. No início do laço (linha 6), seleciona-se uma requisição (a, a') qualquer do problema. Ela serve como parâmetro para o agrupamento. Na próxima linha, forma-se o conjunto P que tem todas as requisições compatíveis com (a, a'), ou seja, requisições que tenham origem, destino e janela de tempo iguais a (a, a'). Posteriormente, seleciona-se um subconjunto C de P de tamanho

$|C| = min\{|P|, M_c\}$. Os pares de nós em C são utilizados para formar uma nova requisição. Essa nova requisição (b, b') (linha 9) representa o agrupamento dos nós em C. O nó b tem a mesma origem e a mesma janela de tempo de a. Além disso, o número de assentos e peso do nó b é definido como a soma dessas propriedades para o conjunto C considerando somente nós *pickup*, respectivamente. Em relação ao nó b', isso é feito de forma análoga. Por fim, os nós em C são removidos de N (linha 16) e os nos nós b, b' (linha 17) são adicionados ao conjunto N^*. O laço do procedimento é executado enquanto houver nós em N. Uma vez finalizado, se o problema é HRPS, as janelas artificiais são removidas do problema e o conjunto E é ajustado para refletir o novo conjunto de nós N^*. O resultado do procedimento é o grafo $G(N^*, E)$.

Algoritmo 37: AGRUPAMENTO(G, M_c)

entrada: Grafo $G(N, E)$ e um número inteiro positivo M_c
saída : Grafo $G(N^*, E)$

1 $N^* \leftarrow \emptyset$;
2 **se** *O problema pertence ao modelo HRPS* **então**
3 | Para cada nó $(a, a') \in N$ defina a janela de tempo como $[0, +\infty]$;
4 **fim**
5 **enquanto** $N \neq \emptyset$ **faça**
6 | Selecione, aleatoriamente, um par de nós *pickup-delivery* (a, a') de N;
7 | Seja $P \subset N \times N$ o conjunto de pares de nós *pickup-delivery* (p, d) em que p e d tenham a mesma localidade e janela de tempo de a e a', respectivamente;
8 | Selecine o conjunto $C \subset P$ de tal forma que $|C| = min\{|P|, M_c\}$;
9 | Crei o nós *pickup-delivery* (b, b') em que b e b' tenham a mesma localidade de a e a', respectivamente;
10 | Defina o número de assentos de b como a soma dos assentos dos nós *pickup* em C;
11 | Defina o número de assentos de b' como a soma dos assentos dos nós *delivery* em C;
12 | Defina o peso de b como a soma dos pesos dos nós *pickup* em C;
13 | Defina o peso de b' como a soma dos pesos dos nós *delivery* em C;
14 | Defina a janela de tempo de b como a janela de a;
15 | Defina a janela de tempo de b' como a janela de a';
16 | Remova os nós que ocorrem em C de N;
17 | $N^* \leftarrow N^* \cup \{b, b'\}$;
18 **fim**
19 **se** *O problema pertence ao modelo HRPS* **então**
20 | Para cada nó $(a, a') \in N$ remova a janela de tempo;
21 **fim**
22 Ajuste o conjunto E para refletir o novo conjunto de nós N^*;
23 **retorne** $G(N^*, E)$;

6.11 Método TRIP

O Algoritmo 38 apresenta o método TRIP. Ele recebe uma solução S em $\Omega(\text{VRP})$ e um número inteiro M_r. O objetivo é formar rotas com até M_r viagens a partir das rotas $r \in S$.

Inicialmente, linha 1, o procedimento ordena as rotas de S de maneira crescente em relação ao início da rota no aeroporto. Após isso, ele inicia um laço (linhas 3-16) que é executado enquanto é possível juntar duas rotas quaisquer r_i, r_j numa nova rota r_t (linha 7), em que o número de rotas de r_t é igual à soma do número de rotas de r_i e r_j. A rota r_t deve ter um número de viagens inferior ou igual ao parâmetro M_r e deve ser válida segundo o modelo HRPS ou HRPM (linha 9). Se isso for verdade, então as rotas r_i, r_j são removidas de S e a nova rota r_t é incluída em S. O algoritmo retorna a solução S em que rotas podem ter até M_r viagens.

Algoritmo 38: $\mathrm{TRIP}(S, M_r)$

entrada: Uma solução S em $\Omega(\mathrm{VRP})$ e o número máximo M_r de viagens.
saída : Uma solução S em $\Omega(\mathrm{VRP})$.

```
1  Ordene as rotas de S de forma crescente em função do horário de início da rota;
2  continue ← TRUE;
3  enquanto continue = TRUE faça
4      continue = FALSE;
5      para cada r_i ∈ S faça
6          para cada r_j ∈ S, r_i ≠ r_j faça
7              r_t = r_i ∪ r_j;
8              Recalcule as informações da rota r_t;
9              se r_t é viável e o número de viagens é inferior ou igual a M_r então
10                 Remova r_i, r_j de S;
11                 Adicione r_t a S;
12                 continue ← TRUE;
13             fim
14         fim
15     fim
16 fim
17 retorne S;
```

6.12 Considerações Finais

Neste capítulo descreveu-se a matheurística MGV-H, uma adaptação do MGV-C para resolver os problemas HRPS e HRPM. Em geral, os passos macros das matheurísticas MGV-C e MGV-H são semelhantes. As diferenças se encontram, principalmente, nos procedimentos de *splitting*, na busca local que opera com soluções inviáveis, no agrupamento de requisições e na formação de várias viagens por rota.

7 Resultados Computacionais

Neste capítulo são apresentados os resultados computacionais para os procedimentos propostos do MGV-C para o CVRP e do MGV-H para os modelos HRPS e HRPM. As matheurísticas MGV-C e MGV-H foram codificadas em C++ e os testes computacionais foram executados num AMD Ryzen 5 2600 Six-Core Processor de 1,5 GHz e 16 GB de RAM, rodando em um computador Linux 4.4.0 x86_64. A matheurística foi executado 10 vezes para cada instância de teste e o pacote CPLEX (versão 12.8) foi usado para resolver os modelos RL-PCC (4.9)-(4.13) e PI-PPC (4.5)-(4.8). Além disso, a execução do CPLEX foi limitada a 4 minutos por causa do PI-PPC, visto que, para algumas instâncias, o tempo para fechar o GAP da solução é excessivo.

7.1 Resultados experimentais do MGV-C para o CVRP

A primeira bateria de teste do MGV-C foi realizada em 91 instâncias *benchmarks* do CVRP denominadas A, B, E, P e M, disponíveis no sítio `http://vrp.galgos.inf.puc-rio.br/`. Os conjuntos de instâncias A, B e P foram propostos por Augerat et al. (1995). O conjunto E foi proposto por Christofides e Eilon (1969). O conjunto M foi elaborado por Christofides, Mingozzi e Toth (1979). O nome das instâncias segue o padrão X-na_n-k-b_n, em que X representa o *benchmark* de origem da instância, a_n o número de requisições e b_n o número de veículos disponíveis. A instância A-n32-k5, por exemplo, pertence ao *benchmark* A, possui 32 requisições e 5 veículos disponíveis.

Além disso, o MGV-C também foi testado em 12 instâncias do *benchmark* Golden proposto por Golden et al. (1998) e 7 instâncias do *benchmark* CMT definido por Christofides, Mingozzi e Toth (1979). O *benchmark* Golden é formado por instâncias que contêm entre 240 e 480 requisições, enquanto o CMT possui instâncias contendo entre 50 e 100 requisições. Esses *benchmarks* também estão disponíveis no endereço `http://vrp.galgos.inf.puc-rio.br/`.

De modo a realizar o ajuste fino do MGV-C, usou-se o pacote *Iterated Racing Package* (Irace) (LÓPEZ-IBÁÑEZ et al., 2016), uma ferramenta de configuração automática de algoritmos baseada no *Iterated F-Race* (BALAPRAKASH; BIRATTARI; STÜTZLE, 2007). Dado um conjunto representativo de instâncias de um problema, o Irace encontra a melhor configuração de parâmetros de um algoritmo através de uma série de iterações, na qual um conjunto de configurações candidatas são aplicadas para refinar os resultados de um algoritmo. Se evidência suficiente é recolhida durante os testes, configurações estatisticamente piores nos testes são removidas até que um conjunto de configurações, chamado elite, seja formado. As instâncias A-n45-k6, A-n80-k10, B-n57-k7 e B-n68-k9 foram usadas para o treinamento por serem representativas do primeiro conjunto de instâncias. Os valores dos parâmetros para o MGV-C

usado no Irace e os melhores candidatos obtidos são mostrados na Tabela 10.

Tabela 10 – Ajuste fino dos parâmetros do MGV-C pelo Irace.

Parâmetro	Tipo	Valor	Melhor Resultado
M_i	categórico	$\{20000, 30000, 40000, 50000\}$	40000
M_v	categórico	$\{25, 50, 100, 200, 300\}$	25
M_α	real	$[0,01; 0,1]$	0,0865
M_t	real	$[0,01; 0,05]$	0,0477
M_s	categórico	$\{50, 100, 200, 300, 400, 500, 1000\}$	500
ϕ	categórico	$\{\phi_w, \phi_{sw}, \phi_b, \phi_{sb}\}$	ϕ_{sw}

No intuito de avaliar o desempenho do MGV-C nos *benchmarks* A, B, P, E e M, uma comparação é conduzida com alguns outros métodos publicados para a resolução do CVRP. Esses algoritmos foram escolhidos, principalmente, porque eles são propostas que apresentam resultados para os *benchmarks* selecionados. Os algoritmos escolhidos são o DELS (TEOH; PONNAMBALAM; KANAGARAJ, 2015), o LNS-ACO (AKPINAR, 2016), o CVRP-FA (ALTABEEB; MOHSEN; GHALLAB, 2019), OHGA (LIN et al., 2019) e o HGA-VNS (SBAI; KRICHEN; LIMAM, 2020).

As Tabelas 11 a 15 exibem as comparações dos algoritmos selecionados nos conjuntos de dados utilizados. As colunas **Instância** e **BKS** indicam o nome da instância e o valor da melhor solução conhecida. As colunas **Best** e **AVG** representam a melhor solução encontrada e o valor médio das execuções para cada algoritmo apresentado. As colunas DELS, LNS-ACO e OHGA designam os melhores valores encontrados por esses algoritmos, dado que os valores médios não foram apresentados no trabalho original. Se algum método não mostrou o resultado para alguma instância, então um hífen ($-$) é colocado na respectiva célula da tabela e, caso um método apresente uma solução que usa mais veículos que disponíveis na instância, então um diamante ($\diamond$) é mostrado na sua respectiva célula da tabela. Se presente, as últimas duas linhas da tabela $\bar{X_1}$ e $\bar{X_2}$ mostram a média dos valores de cada método para o conjunto de dados usado. Dado que alguns algoritmos não possuem resultados para todas as instâncias disponíveis naquele conjunto de dados, a linha $\bar{X_1}$ representa o valor médio para as instâncias comuns entre os algoritmos, enquanto $\bar{X_2}$ exibe o valor médio somente para os algoritmos com resultados para todas as instâncias no conjunto de dados. Em negrito estão os melhores valores encontrados entre os algoritmos.

Para o *benchmark* A, em relação à melhor solução encontrada, a Tabela 11 revela que MGV-C sempre encontra o BKS (coluna **AVG**) em 21 das 27 instâncias, enquanto ele acha, pelo menos, um BKS (coluna **Best**) para cada instância. Além disso, o MGV-C tem o menor valor X_1 entre os algoritmos mostrados para as colunas de melhor resultado e média. O algoritmo HGA-VNS e o método proposto MGV-C conseguem resultados similares e o melhor desempenho entre os procedimentos apresentados. Entretanto, enquanto o MGV-C obtém os melhores resultados para as instâncias pequenas, o HGA-VNS, normalmente, encontra os melhores re-

sultados para as instâncias maiores.

Tabela 11 – Resultados para o conjunto A.

Instância	BKS	DELS	LNS-ACO	OHGA	CVRP-FA		HGA-VNS		MGV-C	
					Best	Avg	Best	Avg	Best	Avg
A-n32-k5	784	**784**	**784**	**784**	796	798,1	**784**	802	**784**	**784**
A-n33-k5	661	**661**	**661**	**661**	**661**	**661**	**661**	679	**661**	**661**
A-n33-k6	742	**742**	**742**	**742**	**742**	742,7	**742**	760	**742**	**742**
A-n34-k5	778	**778**	**778**	**778**	**778**	**778**	779	796	**778**	**778**
A-n36-k5	799	**799**	**799**	**799**	**799**	804,2	**799**	**799**	**799**	**799**
A-n37-k5	669	**669**	**669**	**669**	**669**	**669**	**669**	**669**	**669**	**669**
A-n37-k6	949	**949**	**949**	**949**	**949**	955,5	**949**	**949**	**949**	**949**
A-n38-k5	730	**730**	**730**	**730**	**730**	730,7	**730**	**730**	**730**	**730**
A-n39-k5	822	**822**	**822**	**822**	**822**	**822**	**822**	**822**	**822**	**822**
A-n39-k6	831	**831**	**831**	833	**831**	834,6	**831**	**831**	**831**	**831**
A-n44-k6	937	**937**	**937**	**937**	**937**	**937**	**937**	**937**	**937**	**937**
A-n45-k6	944	**944**	958	953	953	959,4	**944**	**944**	**944**	945,2
A-n45-k7	1146	**1146**	**1146**	**1146**	1147	1153,3	**1146**	**1146**	**1146**	**1146**
A-n46-k7	914	**914**	**914**	**914**	**914**	**914**	919	931	**914**	**914**
A-n48-k7	1073	**1073**	1084	**1073**	**1073**	**1073**	**1073**	1090	**1073**	**1073**
A-n53-k7	1010	**1010**	**1010**	1017	1011	1014,8	**1010**	**1010**	**1010**	**1010**
A-n54-k7	1167	**1167**	**1167**	**1167**	1172	1172	**1167**	**1167**	**1167**	**1167**
A-n55-k9	1073	**1073**	**1073**	1074	1074	1078,6	**1073**	**1073**	**1073**	**1073**
A-n60-k9	1354	**1354**	**1354**	1355	1355	1364,7	**1354**	**1354**	**1354**	**1354**
A-n61-k9	1034	1035	1067	1035	1039	1048,4	**1034**	**1034**	**1034**	**1034**
A-n62-k8	1288	**1288**	1308	1308	1298	1312,2	**1288**	**1288**	**1288**	1289,4
A-n63-k10	1314	1316	1329	1329	**1314**	1333,4	**1314**	**1314**	**1314**	1315,8
A-n63-k9	1616	1624	1649	1630	1630	1644,8	**1616**	**1616**	**1616**	1616,2
A-n64-k9	1401	1416	1415	1416	1420	1424,9	**1401**	**1401**	**1401**	1402
A-n65-k9	1174	1181	1185	1184	1178	1180,7	**1174**	**1174**	**1174**	1175,6
A-n69-k9	1159	1165	1170	1170	1162	1174	**1159**	**1159**	**1159**	**1159**
A-n80-k10	1763	1779	1815	1790	1773	1787,2	**1763**	**1763**	**1763**	**1763**
$\bar{X}_1$	1041,93	1043,96	1049,85	1046,85	1045,44	1050,67	1042,15	1045,85	**1041,93**	**1042,19**

Para o *benchmark* B, a Tabela 12 mostra que o MGV-C sempre encontra o BKS (coluna **AVG**) em 18 das 23 instâncias, enquanto o MGV-C sempre encontra, pelo menos, um BKS (coluna **Best**) para cada instância. Além disso, MGV-C tem os melhores valores para X_1 e X_2. Novamente, os algoritmos HGA-VNS e MGV-C exibem resultados similares e o melhor desempenho em relação aos outros procedimentos. Além disso, o MGV-C tem melhores resultados para instâncias pequenas, médias e duas grandes, enquanto o HGA-VNS alcança resultados ótimos para instâncias grandes.

Para o *benchmark* E, a Tabela 13 mostra que o MGV-C sempre encontra o BKS (coluna **AVG**) em 11 das 13 instâncias, enquanto ele obtém, pelo menos, um BKS (coluna **Best**) para cada instância do *benchmark*. Também, o MGV-C tem os melhores valores para X_1 e X_2. Mais uma vez, os algoritmos HGA-VNS e MGV-C exibem os melhores desempenhos quando comparados com os outros métodos. Entretanto, dessa vez, o MGV-C consegue resultados melhores que o HGA-VNS, visto que o MGV-C consegue encontrar o BKS das três maiores instâncias, mas o HGA-VNS não.

Para o *benchmark* P, em termos de melhores soluções encontradas, a Tabela 14 mostra

Tabela 12 – Resultados para o conjunto B.

Instância	BKS	DELS	LNS-ACO	OHGA	CVRP-FA		HGA-VNS		MGV-C	
					Best	Avg	Best	Avg	Best	Avg
B-n31-k5	672	**672**	**672**	**672**	**672**	**672**	**672**	**672**	**672**	**672**
B-n34-k5	788	**788**	**788**	**788**	**788**	789,7	**788**	**788**	**788**	**788**
B-n35-k5	955	**955**	**955**	**955**	**955**	955,1	**955**	**955**	**955**	**955**
B-n38-k6	805	**805**	**805**	**805**	806	806,2	807	822	**805**	**805**
B-n39-k5	549	**549**	**549**	**549**	550	553,9	552	566	**549**	**549**
B-n41-k6	829	**829**	**829**	**829**	**829**	829,8	**829**	**829**	**829**	**829**
B-n43-k6	742	**742**	**742**	**742**	**742**	**742**	**742**	**742**	**742**	**742**
B-n44-k7	909	**909**	**909**	**909**	**909**	913,3	**909**	921	**909**	**909**
B-n45-k5	751	**751**	**751**	**751**	**751**	754,2	**751**	763	**751**	**751**
B-n45-k6	678	**678**	**678**	680	686	692,8	**678**	690	**678**	**678**
B-n50-k7	741	**741**	**741**	**741**	**741**	744,8	**741**	753	**741**	**741**
B-n50-k8	1312	**1313**	1319	1315	1318	1329,6	1315	1324	**1313**	**1313**
B-n51-k7	1032	1033	**1032**	—	**1032**	◇	1037	1053	**1032**	**1032**
B-n52-k7	747	**747**	**747**	**747**	**747**	747,6	751	768	**747**	**747**
B-n56-k7	707	**707**	**707**	711	709	713,9	710	728	**707**	**707**
B-n57-k7	1153	1166	**1153**	—	**1153**	1162,5	**1153**	**1153**	**1153**	1155
B-n57-k9	1598	1599	**1598**	1603	1610	1615,2	**1598**	**1598**	**1598**	**1598**
B-n63-k10	1496	1504	1514	1531	1503	1540,6	**1496**	**1496**	**1496**	1496,6
B-n64-k9	861	**861**	874	867	862	887,8	**861**	**861**	**861**	861,4
B-n66-k9	1316	1322	1330	1324	1319	1324,8	**1316**	**1316**	**1316**	1316,1
B-n67-k10	1032	**1032**	1050	1042	1042	1067,6	**1032**	**1032**	**1032**	1033,1
B-n68-k9	1272	1281	1290	1290	1278	1288,1	**1272**	**1272**	**1272**	**1272**
B-n78-k10	1221	1230	1228	1245	1224	1248	**1221**	**1221**	**1221**	**1221**
$\bar{X}_1$	951,48	953,10	955,99	956,95	954,33	962,71	952,19	957,95	**951,52**	**951,63**
$\bar{X}_2$	963,74	965,83	967,86	-	966,35	-	964,61	970,57	**963,78**	**963,97**

Tabela 13 – Resultados para o conjunto E.

Instância	BKS	DELS	LNS-ACO	OHGA	CVRP-FA		HGA-VNS		MGV-C	
					Best	Avg	Best	Avg	Best	Avg
E-n13-k4	247	375	**247**	—	—	—	**247**	**247**	**247**	**247**
E-n22-k4	375	**375**	**375**	**375**	**375**	**375**	**375**	**375**	**375**	**375**
E-n23-k3	569	**569**	**569**	**569**	**569**	**569**	**569**	**569**	**569**	**569**
E-n30-k3	534	**534**	**534**	—	**534**	◇	**534**	**534**	**534**	**534**
E-n31-k7	379	—	**379**	—	—	—	**379**	**379**	**379**	**379**
E-n33-k4	835	**835**	**835**	**835**	**835**	845,2	**835**	**835**	**835**	**835**
E-n51-k5	521	—	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**
E-n76-k7	682	695	**682**	692	683	683	**682**	**682**	**682**	**682**
E-n76-k8	735	744	**735**	740	—	—	**735**	**735**	**735**	**735**
E-n76-k10	830	—	**830**	843	835	844,1	**830**	**830**	**830**	**830**
E-n76-k14	1021	1030	1022	1038	1029	1029	1022	1028	**1021**	**1021**
E-n101-k8	815	—	818	22	—	—	818	821	**815**	**816,6**
E-n101-k14	1067	1082	1069	1095	—	—	1069	1075	**1067**	**1068,3**
$\bar{X}_1$	696,40	700,80	755,29	701,80	698,20	695,19	696,60	697,80	**696,40**	**690,43**
$\bar{X}_2$	662,31	-	662,77	-	-	-	662,77	663,92	**662,31**	**662,53**

que o MGV-C sempre encontra o BKS (coluna **AVG**) em 19 das 24 instâncias, enquanto ele sempre acha, pelo menos, um BKS (coluna **Best**) para cada instância. Além disso, o MGV-C tem os menores valores de X_1 e X_2 entre os algoritmos. Os algoritmos HGA-VNS e MGV-C alcançam resultados similares, mas o HGA-VNS não obtém soluções boas com a mesma frequência que o MGV-C como mostra a coluna **AVG**.

Tabela 14 – Resultados para o conjunto P.

Instância	BKS	DELS	LNS-ACO	OHGA	CVRP-FA		HGA-VNS		MGV-C	
					Best	Avg	Best	Avg	Best	Avg
P-n16-k8	450	**450**	**450**	**450**	**450**	**450**	**450**	**450**	**450**	**450**
P-n19-k2	212	**212**	**212**	**212**	**212**	**212**	**212**	**212**	**212**	**212**
P-n20-k2	216	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**
P-n21-k2	211	**211**	**211**	**211**	**211**	**211**	**211**	**211**	**211**	**211**
P-n22-k2	216	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**
P-n22-k8	603	**603**	—	—	**603**	◇	**603**	**603**	**603**	**603**
P-n23-k8	529	—	**529**	**529**	**529**	**529**	**529**	**529**	**529**	**529**
P-n40-k5	458	**458**	**458**	**458**	**458**	459,9	**458**	**458**	**458**	**458**
P-n45-k5	510	**510**	**510**	**510**	**510**	**510**	**510**	**510**	**510**	**510**
P-n50-k7	554	**554**	**554**	556	**554**	557,3	555	557	**554**	**554**
P-n50-k8	631	641	643	—	**631**	633,4	**631**	**631**	**631**	631,3
P-n50-k10	696	**696**	**696**	700	◇	◇	**696**	706	**696**	**696**
P-n51-k10	741	742	747	**741**	742	750,3	745	759	**741**	**741**
P-n55-k7	568	**568**	**568**	**568**	**568**	573,5	**568**	571	**568**	**568**
P-n55-k10	694	**694**	**694**	698	697	702,9	**694**	**694**	**694**	**694**
P-n55-k15	989	**989**	**989**	**989**	◇	◇	**989**	**989**	**989**	**989**
P-n60-k10	744	**744**	755	749	749	752,6	**744**	**744**	**744**	**744**
P-n60-k15	968	**968**	977	985	**968**	983,2	970	979	**968**	**968**
P-n65-k10	792	**792**	800	797	**792**	799,3	795	803	**792**	**792**
P-n70-k10	827	**827**	837	841	**827**	827,9	**827**	**827**	**827**	**827**
P-n76-k4	593	**593**	598	600	**593**	598,8	596	596	**593**	**594,3**
P-n76-k5	627	629	645	630	628	630,7	**627**	**627**	**627**	627,5
P-n101-k4	681	685	—	696	**681**	684,7	**681**	**681**	**681**	681,1
$\bar{X}_1$	550,56	552,00	554,28	553,72	551,11	554,47	551,28	553,28	**550,56**	**550,66**
$\bar{X}_2$	587,39	-	-	-	-	-	587,96	589,96	**587,39**	**587,49**

Para o *benchmark* M, em relação às melhores soluções encontradas, a Tabela 15 mostra que o MGV-C sempre acha a melhor solução BKS (coluna **AVG**) em 2 das 5 instâncias, enquanto ele consegue obter pelo menos um BKS (coluna **Best**) para 3 instâncias. Além do mais, o MGV-C é capaz de encontrar boas soluções para as instâncias M-n200-k16 e M-n200-k17. Note que as instâncias M-n200-k16 e M-n200-k17 são iguais, exceto pelo número de veículos. Assim, a instância M-n200-k16 é mais difícil de resolver que a M-n200-k17 e nem todas as heurísticas encontram resultados para M-n200-k16.

Além disso, para realizar uma comparação entre os múltiplos algoritmos, o teste não paramétrico de Friedman (BRAZDIL; SOARES, 2000) foi aplicado para, estatisticamente, validar os resultados obtidos. No caso de heurísticas, a hipótese nula H_0 do teste de Friedman pode ser descrita como: *"não há nenhuma diferença de resultados entre as heurísticas"*. Já a hipótese alternativa H_a é definida como: *"há, pelo menos, uma heurística com resultados conflitantes com*

Tabela 15 – Resultados para o conjunto M.

Instância	BKS	LNS-ACO	CVRP-FA		MGV-C	
			Best	Avg	Best	Avg
M-n101-k10	820	**820**	—	—	**820**	**820**
M-n121-k7	1034	**1034**	—	—	**1034**	**1034**
M-n151-k12	1015	1033,3	—	—	**1015**	**1018**
M-n200-k16	1274	—	—	—	**1279**	**1303,3**
M-n200-k17	1275	1324,3	1328	1338,7	**1277**	**1283,8**
X_1	956,33	962,42	-	-	**956,33**	**957,33**

o restante das heurísticas". A hipótese é testada nas instâncias que possuem resultados para todos os algoritmos, o que representa 71 instâncias. Para uma melhor comparação, os custos das melhores soluções conhecidas (BKS) são incluídos no teste junto às outras heurísticas.

Tabela 16 – O p-valor obtido pelo teste de Friedman nas instâncias de testes.

χ^2	df	p-valor
46,84	6	$2,01 \times 10^{-08}$

Os resultados para o primeiro teste de Friedman são apresentados na Tabela 16. A coluna **df** designa o grau de liberdade. O nível de significância é preestabelecido como 95%. O p-valor $= 2,01 \times 10^{-08}$ indica que o teste é altamente significativo e a hipótese nula H_0 pode ser descartada. Isso implica que pelo menos um dos métodos analisados é inconsistente com os resultados dos outros algoritmos. Para identificar o procedimento destoante, o teste de Nemenyi (NEMENYI, 1962) é aplicado como um teste *post-hoc* para efetuar uma comparação por pares de métodos. Os resultados dessa comparação estão exibidos na Tabela 17. Os p-valores, em negrito, designam que a hipótese H_0 é rejeitada para o par de métodos representados pela linha e pela coluna. Note que três grupos são formados quando H_0 é rejeitado. O primeiro, é composto pelos algoritmos OHGA, LNS-ACO e CVRP-FA. O segundo, é formado pelos métodos DELS, LNS-ACO, CVRP-FA e HGA-VNS. O terceiro, é constituído dos elementos BKS, DELS, HGA-VNS e MGV-C.

Tabela 17 – Os p-valores da análise *post-hoc* de Nemenyi para o teste de Friedman com nível de significância de 95%.

	BKS	OHGA	DELS	LNS-ACO	CVRP-FA	HGA-VNS	MGV-C
BKS	n/a	**0,000**	0,395	**0,001**	**0,002**	0,602	1,000
OHGA	**0,000**	n/a	**0,025**	1,000	1,000	**0,008**	**0,000**
DELS	0,395	**0,025**	n/a	0,208	0,395	1,000	0,395
LNS-ACO	**0,001**	1,000	0,208	n/a	1,000	0,091	**0,001**
CVRP-FA	**0,002**	1,000	0,395	1,000	n/a	0,157	**0,002**
HGA-VNS	0,602	**0,008**	1,000	0,091	0,157	n/a	0,602
MGV-C	1,000	**0,000**	0,395	**0,001**	**0,002**	0,602	n/a

Dado que o BKS está agrupado com DELS, HGA-VNS e MGV-C, o teste de Friedman

é repetido, mas dessa vez, restrito a este grupo. A hipótese é avaliada nas 82 instâncias com resultados para esses algoritmos. Os resultados são apresentados na Tabela 18. Novamente, o p-valor $< 0,05$ e a hipótese H_0 é rejeitada, visto que existe, pelo menos, um algoritmo destoante em relação aos resultados do grupo. Dessa vez, o método DELS é descartado como mostrado na Tabela 19, a qual apresenta os p-valores da análise *post-hoc* de Nemenyi para o teste de Friedman.

Tabela 18 – Teste de Friedman para BKS, DELS, HGA-VNS e MGV-C nas instâncias de teste.

χ^2	df	p-valor
11,80	3	$8,10 \times 10^{-03}$

Tabela 19 – Os p-valores da análise *post-hoc* de Nemenyi para o teste de Friedman para BKS, DELS, HGA-VNS e MGV-C com nível de significância de 95%.

	BKS	DELS	HGA-VNS	MGV-C
BKS	n/a	**0,021**	0,178	0,855
DELS	**0,021**	n/a	0,602	**0,021**
HGA-VNS	0,178	0,602	n/a	0,178
MGV-C	0,855	**0,021**	0,178	n/a

Por fim, o teste de Friedman é repetido incluindo somente BKS, HGA-VNS e MGV-C. A hipótese desta vez não é rejeitada (p-valor $> 0,05$), portanto não há nenhuma diferença na qualidade das soluções fornecidas pelos métodos HGA-VNS, MGV-C e as melhores soluções conhecidas (BKS) como mostrado na Tabela 20.

Tabela 20 – Teste de Friedman para BKS, HGA-VNS e MGV-C nas instâncias de teste.

χ^2	df	p-valor
4,23	2	0,1208

Para avaliar o desempenho do MGV-C nos *benchmarks* CMT e Golden, uma avaliação é conduzida com outros métodos recentemente publicados para o CVRP. Os algoritmos escolhidos são o RRTR (GROËR; GOLDEN; WASIL, 2011), o HGSADC (VIDAL et al., 2012), o GRELS (PRINS, 2009), o VTS (KWON et al., 2007) e o PEO (AMMI; CHIKHI, 2016). As Tabelas 21 e 22 mostram a comparação para os algoritmos selecionados para os *benchmarks* CMT e Golden. Como nenhum valor médio está disponível para os procedimentos RRTR, HGSADC, GRELS, PEO e VTS, as colunas deles nas tabelas representam a melhor solução encontrada para cada instância.

Para o *benchmark* CMT, em relação às melhores soluções encontradas, a Tabela 21 revela que o MGV-C encontrada o BKS (coluna **MGV-C**) em 5 das 7 instâncias. Além disso, o MGV-C tem a média X_1 similar ao BKS, RRTR e HGSADC. Entretanto, o MGV-C somente supera, em termos de melhores soluções encontradas, os métodos VTS e HGSADC. Quando

o teste de Friedman é repetindo usando os valores apresentados na Tabela 21, o p-valor encontrado é igual a 0,113. Infere-se, portanto, que não há diferença de qualidade nas soluções geradas pelas heurísticas MGV-C, RRTR, HGSADC, GRELS e VTS.

Tabela 21 – Resultados para o conjunto CMT.

Instância	BKS	RRTR	HGSADC	GRELS	VTS	MGV-C
CMT1	**524,61**	**524,61**	**524,61**	**524,61**	**524,61**	**524,61**
CMT2	**835,26**	**835,26**	**835,26**	**835,26**	844,45	**835,26**
CMT3	**826,14**	**826,14**	**826,14**	**826,14**	828,74	**826,14**
CMT4	**1028,42**	**1028,42**	**1028,42**	1029,48	1040,22	1029,65
CMT5	**1291,29**	1291,45	1291,45	1294,09	1324,01	1292,54
CMT11	**1042,11**	**1042,11**	**1042,11**	**1042,11**	1043,90	**1042,11**
CMT12	**819,56**	**819,56**	**819,56**	**819,56**	**819,56**	**819,56**
$\bar{X}_1$	909,63	909,65	909,65	910,18	917,93	909,98

Para o *benchmark* Golden, em relação às melhores soluções encontradas, a Tabela 21 mostra que o MGV-C não foi capaz de encontrar nenhuma solução conhecida (coluna **BKS**), enquanto três métodos distintos encontram pelo menos uma. Em geral, o MGV-C obtém soluções que são superiores, em média, a 3,8% do valor da coluna **BKS**. Além disso, ele supera somente o algoritmo VTS. Uma explicação para esse baixo desempenho é que, para instâncias como as da Golden, os modelos matemáticos PPC e PCC não são capazes de encontrar boas soluções no tempo apropriado.

Tabela 22 – Resultados para o conjunto Golden.

Instância	BKS	RRTR	HGSADC	PEO	VTS	MGV-C
Golden9	**579,71**	**579,71**	**579,71**	**579,71**	589,60	604,52
Golden10	**735,43**	737,28	736,26	736,26	749,78	779,46
Golden11	**911,98**	913,35	912,84	912,84	961,64	978,06
Golden12	**1101,5**	1102,76	1102,69	1102,87	1173,37	1194,28
Golden13	**857,19**	**857,19**	**857,19**	**857,19**	904,86	866,703
Golden14	**1080,55**	**1080,55**	**1080,55**	1081,5	1143,47	1093,37
Golden15	**1337,27**	1338,19	1337,92	1338,9	1439,71	1391,99
Golden16	**1611,28**	1613,66	1612,50	1612,5	1706,25	1697,59
Golden17	**707,76**	**707,76**	**707,76**	**707,76**	717,26	710,95
Golden18	**995,13**	**995,13**	**995,13**	**995,13**	1023,64	1016,43
Golden19	**1365,60**	**1365,60**	**1365,60**	**1365,4**	1403,39	1392,29
Golden20	**1817,59**	1818,25	1818,32	1818,75	1879,44	1874,43
$\bar{X}_1$	1091,75	1092,45	1092,21	1092,40	1141,03	1133,34

A Tabela 23 mostra o tempo computacional médio (em segundos) para os métodos considerados. Para realizar uma comparação mais justa dos tempos computacionais entre os diversos métodos, um fator de conversão foi usado para o tipo de CPU, conforme apresentado na Tabela 24. Usou-se o método *Single Thread Rating* conforme disponível em `https://www.cpubenchmark.net/`. Em relação à Tabela 23, as heurísticas DELS e CVRP-FA não

publicaram esses valores, portanto, nenhum dado está disponível. Para o grupo de instâncias A, B, P e E, o tempo do MGV-C é inferior às outras heurísticas. Em relação aos *benchmarks* Golden e CMT, o tempo é superior, pelo menos, entre dez e cem vezes em relação ao GRELS e VTS. Um dos motivos desse comportamento é devido ao tempo demandado para resolver o PPC no final do MGV-C. Para algumas instâncias, ele usa todo o tempo configurado. Isso ocorre porque para instâncias maiores há mais rotas, o que torna mais difícil de encontrar uma solução exata. A linha X_1 denota a média dos tempos, excluindo o *benchmark* M.

Tabela 23 – Tempo computacional médio (em segundos) para as instâncias de teste.

Benchmark	OHGA	LNS-ACO	HGA-VNS	MGV-C
A	379,24	1541,10	29,36	16,6
B	335,92	1642, 74	18,72	18,0
E	880,08	2562,12	20,88	16,4
P	412,68	1498,2	29,92	16,5
M	-	-	-	150,5
X_1	501,98	1811,04	24,74	16,88

Benchmark	GRELS	VTS	HGSADC	MGV-C
CMT	7,85	22,75	192,52	906,90
Golden	101,25	438,43	2151,60	1061,30
X_1	54,55	230,60	1172,06	984,1

Tabela 24 – Fator de conversão de CPU.

OHGA[a]	LNS-ACO[b]	HGA-VNS[c]	MGV-C [d]
0,76	0,66	0,40	1,0

[a] Intel Core i7-3615QM @ 2.30GHz
[b] Core(TM) i7-2640 CPU 2.80 GHz
[c] Intel Core i3-4005U @ 1.70GHz
[d] AMD Ryzen 5 2600

GRELS[e]	VTS[f]	HGSADC [g]	MGV-C[d]
0,18	0,31	0,24	1,0

[e] Intel Pentium 4 2.80GHz
[f] Intel Xeon 3.20GHz
[g] Pentium 4 3.00 GHz
[d] AMD Ryzen 5 2600

Para analisar a robustez dos resultados obtidos pelo MGV-C, a Tabela 25 apresenta o desvio padrão não nulo para as instâncias de teste. A coluna **Instância** representa o nome da instância e a coluna **DP** designa o desvio padrão. Analisando os resultados, nota-se que a maioria dos desvios padrões são inferiores a 10, exceto para as instâncias M-n200-k16 e Golden20. A primeira representa uma instância que é reconhecida pela dificuldade de encontrar o ótimo, enquanto a Golden20 é a maior instância do seu *benchmark*.

Tabela 25 – Desvio padrão para os resultados obtidos nas instâncias do CVRP.

Instância	DP	Instância	DP	Instância	DP
A-n45-k6	2,452	P-n50-k8	1,300	Golden9	4,172
A-n62-k8	0,943	P-n76-k4	2,558	Golden10	4,702
A-n63-k10	0,100	P-n76-k5	0,527	Golden11	5,011
A-n63-k9	3,302	P-n101-k4	0,527	Golden12	6,609
A-n64-k9	2,234	M-n151-k12	2,014	Golden13	4,216
A-n65-k9	0,516	M-n200-k16	16,379	Golden14	5,767
B-n57-k7	1,703	M-n200-k17	8,774	Golden15	8,497
B-n63-k10	0,471	CMT2	0,007	Golden16	9,324
B-n64-k9	1,135	CMT3	0,649	Golden17	1,229
B-n66-k9	0,100	CMT4	0,537	Golden18	5,815
B-n67-k10	1,581	CMT5	4,618	Golden19	5,393
E-n101-k8	1,269	CMT11	0,491	Golden20	12,621
E-n101-k14	1,932				

Para uma análise da contribuição dos diversos componentes do MGV-C em termos de tempo e qualidade de solução gerada, o Apêndice C pode ser consultado.

7.2 Resultados experimentais do MGV-H para o HRPS

O MGV-H foi avaliado no conjunto de instâncias proposto por Rosa et al. (2016) a partir de dados reais de operações em plataformas marítimas no Brasil. No total, foram geradas 23 instâncias de 5 a 160 requisições de transporte com a disponibilidade de 12, 25 ou 100 helicópteros. As primeiras 22 instâncias foram construídas usando dados reais da Petrobras. A última instância tem 1000 requisições com 100 helicópteros e foi gerada para avaliar a efetividade do método CS. O nome das instâncias segue o padrão Ha_n, em que a_n é o número de requisições, conforme mostrado na Tabela 26. A coluna **Instância** designa o nome da instância, as colunas **NR** e **NH** denotam o número de requisições e o número de helicópteros disponíveis, respectivamente.

A Tabela 27 apresenta os valores usados no ajuste fino do MGV-H ao problema HRPS usando o Irace. O conjunto Π_{HRPS} representa todas as permutações possíveis do vetor (γ_F, γ_S, γ_T, γ_W). No caso, a combinação eleita pelo Irace para o parâmetro M_γ foi $(\gamma_S, \gamma_W, \gamma_T, \gamma_F)$. Assim, a busca local BL-HRP remove as indisponibilidades conforme a ordem definida pela variável M_γ, ou seja, remove primeiro as indisponibilidades relacionadas a assentos (γ_S), peso (γ_W), tempo (γ_T) e, por último, combustível (γ_F). As instâncias usadas para o ajuste foram a H30 e a H50. Além disso, no método PEN-PROPRIEDADE, a constante Y_1 tem valor igual a 10^5.

A Tabela 28 apresenta os resultados experimentais da aplicação do MGV-H ao HRPS. As colunas **Instância** e **Id** designam o nome e o número identificador da instância. As colunas **CS** e **MGV-H** designam o método CS apresentado por Rosa et al. (2016) e a matheurística

Tabela 26 – Instâncias HRPS.

Instância	Id	NR	NH	Instância	Id	NR	NH
H05	1	5	15	H70	13	70	25
H10	2	10	15	H80	14	80	25
H15	3	15	15	H90	15	90	25
H20	4	20	15	H100	16	100	25
H25	5	25	15	H110	17	110	25
H30	6	30	15	H120	18	120	25
H35	7	35	15	H130	19	130	25
H40	8	40	15	H140	20	140	25
H45	9	45	15	H150	21	150	25
H50	10	50	15	H160	22	160	25
H55	11	55	15	H1000	23	1000	100
H60	12	60	25				

Tabela 27 – Ajuste fino dos parâmetros do MGV-H via Irace para o HRPS.

Parâmetro	Tipo	Valor	Melhor Resultado
M_i	categórico	$\{250, 500, 750, 1000, 2000\}$	1000
M_v	categórico	$\{25, 50, 75, 100, 200\}$	50
M_α	real	$[0{,}01, 0{,}1]$	0,0521
M_s	categórico	$\{5, 10, 20, 50, 100\}$	20
M_r	categórico	$\{1\}$	1
M_c	categórico	$\{0, 2, 3, 4, 6, 8\}$	4
M_k	categórico	$\{0{,}01, 0{,}05, 0{,}10, 0{,}20\}$	0,10
M_γ	categórico	Π_{HRPS}	$(\gamma_S, \gamma_W, \gamma_T, \gamma_F)$
ϕ	categórico	$\{\phi_w, \phi_{sw}, \phi_b, \phi_{sb}\}$	ϕ_{sw}

MGV-H. As colunas **Best**, **Worst** e **T(s)** mostram o melhor resultado, o pior resultado e o tempo médio de execução (em segundos) para os métodos apresentados. A coluna **R(%)** exibe o valor relativo entre CS e MGV-H para a coluna **Best**. A coluna **CPLEX** apresenta os dados da resolução do modelo HRPS feita pelo pacote CPLEX. A coluna **OF** apresenta o valor da função objetivo, a coluna **LB** apresenta o *lower bound*, a coluna **UB** mostra o valor do *upper bound* e a coluna **GAP** exibe o *gap* da solução. Dado que o modelo exato é difícil de resolver, quando algum valor não é encontrado, um hífen (-) é mostrado no seu lugar. O método *Single Thread Rating* foi usado para comparação de tempos e um fator multiplicativo de 0,60 foi aplicado ao tempo do CS. Em negrito, estão os melhores valores obtidos pelos métodos.

Analisando os dados da Tabela 28, nota-se que os dois métodos apresentam resultados similares para as primeiras cinco instâncias. Entretanto, a partir da sexta instância, o método adaptado MGV-H sempre obtém as melhores soluções, tanto para a coluna **Best** quanto para a coluna **Worst**. Por fim, o melhor resultado relativo para o MGV-H é para a instância H1000, a maior instância, na qual o valor relativo é de 55,17%. Isso significa que o MGV-H encontrou uma solução com valor próximo à metade da encontrada pelo CS. Em relação ao tempo, o método CS foi projetado para sempre executar por cinco minutos. Por isso, o tempo apresentado é a

média das execuções até encontrar a melhor solução. Considerando isso, note que para todas as instâncias, exceto a última, o tempo de execução do MGV-H é inferior ao CS. Além disso, ambos métodos utilizam agrupamento de requisições com o CS agrupando até 11 requisições, enquanto o MGV-H o faz com até quatro requisições.

Tabela 28 – Resultados experimentais para o HRPS.

Instância	Id	CPLEX					CS[a]			MGV-H[b]			R(%)
		OF	LB	UB	GAP(%)	T(s)	Best	Worst	T(s)	Best	Worst	T(s)	
H05	1	**1.135**	**1.135**	**1.135**	0	1,54	**1.135**	**1.135**	1,15	**1.135**	**1.135**	0,11	100,00
H10	2	-	1.184	2.126	44,31	28.800	**2.126**	**2126**	1,386	**2.126**	2.128	0,33	100,00
H15	3	-	1.088	2.128	48,88	28.800	**2.128**	**2.128**	11,00	**2.128**	**2.128**	0,59	100,00
H20	4	-	1.064	2454	56,64	28.800	**2.158**	**2.170**	106,52	**2.158**	2.177	0,88	100,00
H25	5	-	954	-	-	28.800	**2.170**	2.177	100,51	**2.170**	**2.171**	1,162	100,00
H30	6	-	1.072	-	-	28.800	2.273	2.284	191,47	**2.171**	**2.180**	1,55	95,51
H35	7	-	1.036	-	-	28.800	3.083	3.094	209,60	**2.177**	**2.201**	2,77	70,61
H40	8	-	1.036	-	-	28.800	3.257	3.264	128,62	**2.181**	**2.210**	2,88	66,96
H45	9	-	954	-	-	28.800	3.291	3.365	197,45	**3.105**	**3.122**	4,89	94,35
H50	10	-	906	-	-	28.800	4.126	4.194	174,61	**3.284**	**3.302**	5,40	79,59
H55	11	-	906	-	-	28.800	4.158	4.221	196,22	**3.286**	**3.310**	5,27	79,37
H60	12	-	1.036	-	-	28.800	4.149	4.164	164,67	**4.112**	**4.134**	6,20	99,11
H70	13	-	906	-	-	28.800	5.035	5.046	157,74	**4.125**	**4.145**	5,19	81,93
H80	14	-	906	-	-	28.800	6.487	6.591	214,13	**4.154**	**4.182**	5,58	82,50
H90	15	-	906	-	-	28.800	7.144	7.157	188,38	**5.214**	**5.243**	9,04	72,98
H100	16	-	1.036	-	-	28.800	8.036	8.058	153,51	**5.233**	**5.333**	7,32	65,12
H110	17	-	906	-	-	28.800	9.153	9.184	167,63	**6.164**	**6.222**	13,47	67,34
H120	18	-	906	-	-	28.800	10.121	10.133	153,52	**6.237**	**6.280**	12,00	61,62
H130	19	-	906	-	-	28.800	11.004	11.024	169,76	**7.139**	**7.309**	15,88	64,88
H140	20	-	906	-	-	28.800	11.058	11.082	221,11	**7.336**	**7.556**	17,72	66,34
H150	21	-	906	-	-	28.800	12.095	12.124	148,91	**7.339**	**8.245**	17,75	60,68
H160	22	-	906	-	-	28.800	12.115	12.156	179,21	**8.174**	**8.301**	20,43	67,47
H1000	23	-	-	-	-	28.800	96.926	97.038	170,14	**53.469**	**54.580**	390,79	55,17
X_1	-	-	-	-	-	-	9.705,57	9.735,43	148,14	6.287,70	6.417,13	23,79	79,63

[b] Intel Core i7-2720QM @ 2.20GHz

[b] AMD Ryzen 5 2600

Para analisar a robustez dos resultados obtidos pelo MGV-H para o HRPS, a Tabela 29 apresenta o desvio padrão para cada instância. A coluna **Instância** representa o nome da instância e a coluna **DP** designa o desvio padrão. Analisando os resultados, nota-se para as instâncias com até 100 requisições que o desvio padrão é inferior a 10. Em especial, as instâncias H150 e H1000 possuem os maiores desvios. Isso ocorre nelas porque algumas soluções obtidas usam mais helicópteros, o que implica num custo maior na função objetivo.

7.3 Resultados experimentais do MGV-H para o HRPM

O MGV-H foi avaliado no conjunto de instâncias proposto por Machado et al. (2019) a partir de dados reais de operações em plataformas marítimas no Brasil. No total, foram geradas 14 instâncias de 5 a 80 requisições de transporte com a disponibilidade de 10 helicópteros. O nome das instâncias segue o padrão Ka_n, em que a_n é o número de requisições, conforme mostrado na Tabela 30. A coluna **Instância** designa o nome da instância, as colunas **NR** e **NH** denotam o número de requisições e o número de helicópteros disponíveis, respectivamente.

Tabela 29 – Desvio padrão para os resultados obtidos nas instâncias do HRPS.

Instância	DP	Instância	DP	Instância	DP
H05	0,000	H45	5,228	H110	16,734
H10	0,966	H50	5,797	H120	14,622
H15	0,000	H55	7,637	H130	51,542
H20	8,208	H60	6,944	H140	67,413
H25	0,316	H70	7,008	H150	390,496
H30	3,479	H80	9,606	H160	41,580
H35	8,475	H90	9,663	H1000	428,642
H40	9,606	H100	28,503		

As instâncias do HRPM tem três tipos de janelas de tempo: no período da manhã, no período da tarde e diurnas.

Tabela 30 – Instâncias HRPM.

Instância	Id	NR	NH	Instância	Id	NR	NH
K05	1	5	10	K40	8	40	10
K10	2	10	10	K45	9	45	10
K15	3	15	10	K50	10	50	10
K20	4	20	10	K55	11	55	10
K25	5	25	10	K60	12	60	10
K30	6	30	10	K70	13	70	10
K35	7	35	10	K80	14	80	10

A Tabela 31 apresenta os valores usados no ajuste fino do MGV-H ao problema HRPM usando o Irace. O conjunto Π_{HRPM} representa todas as permutações possíveis do vetor (γ_F, γ_S, γ_T, γ_W, γ_J). No caso, a combinação eleita pelo Irace para o parâmetro M_γ foi $(\gamma_W, \gamma_S, \gamma_J, \gamma_F, \gamma_T)$. Assim, a busca local BL-HRP remove as indisponibilidades conforme a ordem definida pela variável M_γ, ou seja, remove primeiro as indisponibilidades relacionadas a peso (γ_W), assentos (γ_S), janelas de tempo (γ_J), combustível (γ_F) e, por último, tempo (γ_T). As instâncias usadas para o ajuste foram a K20 e a K50. Além disso, no método PEN-PROPRIEDADE, a constante Y_1 tem valor igual a 10^5.

Tabela 31 – Ajuste fino dos parâmetros do MGV-H via Irace para o HRPM.

Parâmetro	Tipo	Valor	Melhor Resultado
M_i	categórico	$\{250, 500, 750, 1000, 2000\}$	1000
M_v	categórico	$\{25, 50, 75, 100, 200\}$	50
M_α	real	$[0{,}01, 0{,}1]$	0,0902
M_s	categórico	$\{5, 10, 20, 50, 100\}$	10
M_r	categórico	$\{3\}$	3
M_c	categórico	$\{0, 2, 3, 4, 6, 8\}$	4
M_k	categórico	$\{0{,}01, 0{,}05, 0{,}10, 0{,}20\}$	0,20
M_γ	categórico	Π_{HRPM}	$(\gamma_W, \gamma_S, \gamma_J, \gamma_F, \gamma_T)$
ϕ	categórico	$\{\phi_w, \phi_{sw}, \phi_b, \phi_{sb}\}$	ϕ_{sw}

A Tabela 32 apresenta os resultados experimentais da aplicação do MGV-H ao HRPM. As colunas **Instância** e **Id** designam o nome e o número identificador da instância. As colunas **GRASP** e **MGV-H** designam o método GRASP apresentado por Machado et al. (2019) e a matheurística MGV-H. As colunas **Best**, **Worst** e **T(s)** mostram o melhor resultado, o pior resultado e o tempo médio de execução (em segundos) para os métodos apresentados. A coluna **R(%)** exibe o valor relativo entre GRASP e MGV-H para a coluna **Best**. A coluna **CPLEX** apresenta os dados da resolução do modelo HRPM pelo pacote CPLEX. A coluna **OF** apresenta o valor da função objetivo, a coluna **LB** apresenta o *lower bound*, a coluna **UB** mostra o valor do *upper bound* e a coluna **GAP** exibe o *gap* da solução. Dado que o modelo exato é difícil de resolver, quando algum valor não é encontrado, um hífen (-) é mostrado no seu lugar. O método *Single Thread Rating* não é necessário, pois ambos procedimentos executaram na mesma arquitetura. Em negrito, estão os melhores valores obtidos pelos métodos.

Tabela 32 – Resultados experimentais para o HRPM.

Instância	Id	CPLEX					GRASP[a]			MGV-H[a]			R(%)
		FO	LB	UB	T(s)	GAP(%)	Best	Worst	T(s)	Best	Worst	T(s)	
K05	1	**2.732**	2.732	2.732	17,02	0.00	**2.732**	**2.732**	≈ 1	**2.732**	**2.732**	0,20	100
K10	2	-	961	3.095	7.200	68.94	2.806	2.829	3	**2.763**	**2.763**	0,44	98,47
K15	3	-	280	-	7.200	-	3.086	3.264	12	**3.077**	**3.077**	1,16	99,70
K20	4	-	98	-	7.200	-	3.174	3.553	45	**3.085**	**3.085**	1,47	97,20
K25	5	-	-	-	7.200	-	3.749	4.773	88	**3.204**	**3.204**	2,13	85,46
K30	6	-	-	-	7.200	-	**4.488**	5.631	140	4.535	**4.587**	2,72	101,05
K35	7	-	-	-	7.200	-	6.060	6.918	211	**5.146**	**5.146**	4,14	84,92
K40	8	-	-	-	7.200	-	6.271	7.509	334	**5.146**	**5.146**	4,98	82,06
K45	9	-	-	-	7.200	-	7.264	7.957	478	**5.333**	**5.338**	5,80	73,42
K50	10	-	-	-	7.200	-	7.459	8.656	699	**5.341**	**6.055**	6,80	71,60
K55	11	-	-	-	7.200	-	8.803	9.491	938	**6.174**	**6.325**	9,48	70,14
K60	12	-	-	-	7.200	-	9.102	10.274	1.262	**6.353**	**6.472**	11,81	69,80
K70	13	-	-	-	7.200	-	11.742	12.587	1.845	**6.701**	**6.294**	27,16	57,07
K80	14	-	-	-	7.200	-	13.185	15.095	2.484	**7.540**	**7.641**	26,53	57,18
X_1	-	-	-	-	-	-	6.422,93	7.233,5	610	4.795,00	4.847,50	7,49	82,01

[a] AMD Ryzen 5 2600

[b] AMD Ryzen 5 2600

Analisando a Tabela 32, nota-se que o método MGV-H obtém resultados melhores que o GRASP para 12 das 14 instâncias com tempo sempre inferior. Em especial, em nenhuma das execuções o tempo passou de 60 segundos. Para a primeira instância, ambos métodos encontraram a solução exata conforme resolvido pelo CPLEX. Na instância K30 o MGV-H gerou uma solução inferior a do GRASP. Isso ocorreu porque o MGV-H usa uma estratégia de gerar rotas com múltiplas viagens somente após encontrar uma solução localmente mínima. Já o GRASP sempre forma rotas com várias viagens. Por isso, para a instância K30, o MGV-H não conseguiu gerar uma rota com três viagens, enquanto o GRASP foi capaz. Além disso, verifique que, exceto pela instância K30 e pela última instância, o valor da coluna **R(%)** decresce da menor para a maior instância. Em relação ao tempo de execução, o método GRASP trabalha com instâncias viáveis num algoritmo paralelizado. Ele não usa nenhuma estratégia para reduzir o custo computacional no cálculo dos movimentos na busca local como o MGV-H e também não faz agrupamento das requisições. Isso explica, em partes, a discrepância entre os tempos

Tabela 33 – Desvio padrão para os resultados obtidos nas instâncias do HRPM.

Instância	DP	Instância	DP	Instância	DP
K05	0,000	K30	16,444	K55	57,199
K10	0,000	K35	0,000	K60	45,706
K15	0,000	K40	0,000	K70	99,918
K20	0,000	K45	1,776	K80	31,311
K25	0,316	K50	341,318		

computacionais.

Para analisar a robustez dos resultados obtidos pelo MGV-H para o HRPM, a Tabela 33 apresenta o desvio padrão para cada instância. A coluna **Instância** representa o nome da instância e a coluna **DP** designa o desvio padrão. Analisando os resultados, nota-se que, para instâncias com até 45 requisições, o desvio padrão é inferior a 20. Em especial, a instância K50 possui o maior desvio padrão. Isso ocorre porque o método MGV-H, às vezes, gera uma solução que não utiliza todas as viagens disponíveis de uma rota. Como consequência, seria possível transferir viagens de uma rota à outra e reduzir o número de helicópteros necessários, o que diminuiria o custo da solução.

8 Conclusões e Trabalhos Futuros

Esta tese propôs uma nova formulação matemática do HRP chamada HRPM, a qual é uma extensão do HRPS definido por Rosa et al. (2016). O HRPM inclui novas restrições operacionais que garantem o pouso e decolagem seguros em plataformas marítimas. Além disso, ele adiciona também a possibilidade de múltiplas viagens por rota e janelas de tempo para as requisições. Para resolução dos modelos HRPS e HRPM, uma nova matheurística seguindo a abordagem hierárquica RFCS é proposta como método de resolução. Essa nova matheurística é avaliada nos problemas do tipo CVRP para validá-la. Em seguida, ela é adaptada para resolver os modelos HRPS e HRPM.

A matheurística é uma hibridização do GRASP e do VNS com a aplicação dos modelos matemáticos PCC e PPC. Na fase construtiva do GRASP, o modelo matemático do PCC e várias heurísticas construtivas são usados para construir uma solução inicial. Na fase de busca local do GRASP, utiliza-se a meta-heurística VNS. As soluções localmente ótimas geradas pelo VNS são armazenadas para serem usadas nos modelos PCC e PPC. Após a execução de todas as iterações do GRASP, o modelo PPC é aplicado para gerar a solução final.

A versão da matheurística para o CVRP é chamada MGV-C. Ela foi aplicada em sete *benchmarks* da literatura, totalizando 110 instâncias. Os resultados computacionais mostraram que o método proposto é competitivo para instâncias pequenas e médias. Nesses casos, uma análise estatística foi conduzida para avaliar a matheurística com outros algoritmos. Para instâncias grandes, o método foi capaz de encontrar soluções próximas às melhores conhecidas, mas com resultados inferiores a outras heurísticas.

A versão da matheurística para o HRPS e o HRPM é chamada MGV-H. Ela foi avaliada em 23 instâncias do HRPS e 14 instâncias do HRPM. Para o HRPS, usou-se o método CS proposto por Rosa et al. (2016) como comparação. Tanto o CS quanto o MGV-H obtiveram resultados similares para as cinco instâncias menores. Entretanto, para o restante das instâncias, o MGV-H conseguiu resultados superiores. Em especial, para a maior instância, obteve uma solução com valor inferior a 44% da obtida pelo CS. Para o HRPM, utilizou-se o método GRASP proposto por Machado et al. (2019) para avaliação. Das 14 instâncias, o MGV-H superou os resultados em 12 delas. As duas exceções são a menor instância, na qual ambos algoritmos atingiram o mesmo valor e uma instância média, no qual o MGV-H gerou uma solução com valor 1,05% superior à obtida pelo GRASP.

A partir desses resultados, são presentados como sugestão de continuidade a este trabalho as seguintes propostas. A matheurística proposta utiliza diversas heurísticas construtivas. Dado que há outros métodos disponíveis como o CW, propõe-se a utilização deles para avaliar o ganho de novas heurísticas na fase construtiva. Além disso, a integração do PCC relaxado

com as heurísticas construtivas é feita utilizando uma estratégia de ordenação. Durante os experimentos, evidenciou-se a preferência por um certo tipo de estratégia no ajuste fino do método. Assim, propõe-se que novas estratégias sejam avaliadas além das propostas.

A versão do MGV-H funciona com soluções inviáveis, enquanto o MGV-C sempre mantém a solução viável. Como sugestão, sugere-se a modificação do MGV-C para operar com soluções inviáveis, pois é possível que com isso consiga melhores resultados para instâncias grandes. Além disso, há outras opções de funções *splitting* para o MGV-C. Para o HRP e problemas de *pickup* e *delivery* em geral, ainda há poucos estudos relacionados ao *splitting*. Assim, propõe-se o desenvolvimento de novos métodos para transformar soluções entre diversos espaços de soluções para problemas PDVRP

A matheurística conseguiu bons resultados para os problemas CVRP e HRP. Assim, recomenda-se a sua adaptação para outras classes de roteamento. Além disso, a utilização do PCC é frequentemente acompanhada de GC. Dessa forma, uma alteração possível no método é aplicação de GC como ferramenta para resolução do problema. Nesse âmbito, sugere-se também que o subproblema seja resolvido por programação dinâmica. Em especial, as regras de dominância para tratar as diversas restrições do problema não foram encontradas na revisão da literatura.

A matheurística é uma hibridização das meta-heurísticas GRASP e VNS. Isso abre a possibilidade novas hibridizações, especialmente com o GRASP. Uma sugestão é substituir o método VNS por outra busca local como BT ou substituir a busca local do VNS pela BT. Além disso, dado que o método armazena soluções localmente ótimas, uma possibilidade de aprimoramento é aplicação de *path-relinking* com objetivo de explorar melhor o espaço de solução. Na representação da solução, no caso da aplicação no HRPM, permitir que a rota faça várias viagens durante todo o método pode ser o suficiente para que ele resolva todas as instâncias tão bem quanto o GRASP usado na comparação.

Em relação ao modelo HRPM, propõe-se a adoção de novas restrições de segurança. Atualmente, diversos estudos buscam avaliar cenários que aumentem esse requisito. Uma possibilidade é permitir que passageiros troquem de aeronaves numa plataforma em alto-mar. Isso, potencialmente, diminuiria o número de pousos e decolagens. No campo de roteamento de helicópteros, uma nova característica adotada no mundo real é a formação de corredores de voo. Isso tem como consequência a diminuição da complexidade do modelo matemático e permite a resolução de instâncias maiores. No caso de técnicas de resolução exata, a aplicação de BB, BP ou BC podem ser promissoras.

Propõem-se, também, aplicar nos problemas HRPS e HRPM os algoritmos que usam abordagem hierárquica já publicados na literatura. Em especial, os métodos propostos por Velasco et al. (2009) e Abbasi-Pooya e Kashan (2017). Além disso, as matheurísticas MGV-C e MGV-H possuem diversos parâmetros que precisam ser ajustados. Para automatizar essa tarefa de calibração, sugere-se a definição de um procedimento que determine cada um dos

parâmetros das matheurísticas a partir do tamanho da instância do problema.

Referências

ABBASI-POOYA, Amin; KASHAN, Ali Husseinzadeh. New mathematical models and a hybrid Grouping Evolution Strategy algorithm for optimal helicopter routing and crew pickup and delivery. **Computers & Industrial Engineering**, Elsevier, v. 112, p. 35–56, 2017.

ADARANG, Hesam; BOZORGI-AMIRI, Ali; KHALILI-DAMGHANI, Kaveh; TAVAKKOLI-MOGHADDAM, Reza. A robust bi-objective location-routing model for providing emergency medical services. **Journal of Humanitarian Logistics and Supply Chain Management**, Emerald Publishing Limited, 2020.

AGARWAL, Yogesh; MATHUR, Kamlesh; SALKIN, Harvey M. A set-partitioning-based exact algorithm for the vehicle routing problem. **Networks**, Wiley Online Library, v. 19, n. 7, p. 731–749, 1989.

AI, The Jin; KACHITVICHYANUKUL, Voratas. Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. **Computers & Industrial Engineering**, Elsevier, v. 56, n. 1, p. 380–387, 2009.

AKPINAR, Sener. Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. **Expert Systems with Applications**, Elsevier, v. 61, p. 28–38, 2016.

ALINEZHAD, Hamed; YAGHOUBI, Saeed; HOSSEINI-MOTLAGH, Seyyed-Mahdi; ALLAHYARI, Somayeh; SAGHAFI, Mojtaba. An improved particle swarm optimization for a class of capacitated vehicle routing problems. **International Journal of Transportation Engineering**, Tarrahan Parseh Transportation Research Institute, v. 5, n. 4, p. 331–347, 2018.

ALTABEEB, Asma M; MOHSEN, Abdulqader M; GHALLAB, Abdullatif. An improved hybrid firefly algorithm for capacitated vehicle routing problem. **Applied Soft Computing**, Elsevier, v. 84, p. 105728, 2019.

ALTINEL, İ K; ÖNCAN, Temel. A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. **Journal of the Operational Research Society**, Springer, v. 56, n. 8, p. 954–961, 2005.

AMMI, Meryem; CHIKHI, Salim. Cooperative parallel metaheuristics based penguin optimization search for solving the vehicle routing problem. **International Journal of Applied Metaheuristic Computing (IJAMC)**, IGI Global, v. 7, n. 1, p. 1–18, 2016.

AMOUS, Marwa; TOUMI, Said; JARBOUI, Bassem; EDDALY, Mansour. A variable neighborhood search algorithm for the capacitated vehicle routing problem. **Electronic Notes in Discrete Mathematics**, Elsevier, v. 58, p. 231–238, 2017.

ANDREEVA-MORI, Adriana; KOBAYASHI, Keiji; SHINDO, Masato. Particle Swarm Optimization/Greedy-Search Algorithm for Helicopter Mission Assignment in Disaster Relief. **Journal of Aerospace Information Systems**, American Institute of Aeronautics e Astronautics (AIAA), v. 12, n. 10, p. 646–660, out. 2015. DOI: 10.2514/1.i010362. Disponível em: <https://doi.org/10.2514/1.i010362>.

ARMSTRONG-CREWS, Nicholas; MOCK, Kenrick. Helicopter Routing for Maintaining Remote Sites in Alaska using a Genetic Algorithm. In: MENLO PARK, CA; CAMBRIDGE, MA; LONDON; AAAI PRESS; MIT PRESS; 1999, 4. PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE. [S.l.: s.n.], 2005. v. 20, p. 1586.

AUGERAT, Ph; BELENGUER, Jose Manuel; BENAVENT, Enrique; CORBERÁN, A; NADDEF, D; RINALDI, G. **Computational results with a branch and cut code for the capacitated vehicle routing problem**. [S.l.]: IMAG, 1995. v. 34.

BABU, B Ramesh; POOJARY, Manjula; RENUKA, B. Application of Hybrid Ant Colony Optimization (HACO) Algorithm for Solving Capacitated Vehicle Routing Problem (CVRP). **International Journal of Computer Science and Information Technologies**, Citeseer, v. 3, n. 2, p. 3540–3543, 2012.

BALAPRAKASH, Prasanna; BIRATTARI, Mauro; STÜTZLE, Thomas. Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In: SPRINGER. INTERNATIONAL workshop on hybrid metaheuristics. [S.l.: s.n.], 2007. P. 108–122.

BALINSKI, Michel L; QUANDT, Richard E. On an integer program for a delivery problem. **Operations research**, INFORMS, v. 12, n. 2, p. 300–304, 1964.

BARBAROSOĞLU, Gülay; ÖZDAMAR, Linet; CEVIK, Ahmet. An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. **European Journal of Operational Research**, Elsevier, v. 140, n. 1, p. 118–133, 2002.

BAZARAA, Mokhtar S; JARVIS, John J; SHERALI, Hanif D. **Linear programming and network flows**. [S.l.]: John Wiley & Sons, 2011.

BEAMON, Benita M; BALCIK, Burcu. Performance measurement in humanitarian relief chains. **International Journal of Public Sector Management**, Emerald Group Publishing Limited, v. 21, n. 1, p. 4–25, 2008.

BENTLEY, Jon Jouis. Fast algorithms for geometric traveling salesman problems. **ORSA Journal on computing**, INFORMS, v. 4, n. 4, p. 387–411, 1992.

BERGER, Jean; BARKAOUI, Mohamed. A hybrid genetic algorithm for the capacitated vehicle routing problem. In: SPRINGER. GENETIC and evolutionary computation conference. [S.l.: s.n.], 2003. P. 646–656.

BLOCHO, Miroslaw. Heuristics, metaheuristics, and hyperheuristics for rich vehicle routing problems. In: SMART Delivery Systems. [S.l.]: Elsevier, 2020. P. 101–156.

BLUM, Christian; ROLI, Andrea. Hybrid metaheuristics: an introduction. In: HYBRID Metaheuristics. [S.l.]: Springer, 2008. P. 1–30.

BONDY, John-Adrian; MURTY, U. S. R. **Graph theory**. New York, London: Springer, 2007. (Graduate texts in mathematics). OHX. ISBN 978-1-8462-8969-9. Disponível em: <http://opac.inria.fr/record=b1123512>.

BORGULYA, István. An algorithm for the capacitated vehicle routing problem with route balancing. **Central European Journal of Operations Research**, Springer, v. 16, n. 4, p. 331–343, 2008.

BOUROS, Panagiotis; SACHARIDIS, Dimitris; DALAMAGAS, Theodore; SELLIS, Timos. Dynamic pickup and delivery with transfers. In: SPRINGER. INTERNATIONAL Symposium on Spatial and Temporal Databases. [S.l.: s.n.], 2011. P. 112–129.

BRACHNER, Markus; HVATTUM, Lars Magnus. Combined emergency preparedness and operations for safe personnel transport to offshore locations. **Omega**, Elsevier, v. 67, p. 31–41, 2017.

BRAEKERS, Kris; RAMAEKERS, Katrien; VAN NIEUWENHUYSE, Inneke. The vehicle routing problem: State of the art classification and review. **Computers & Industrial Engineering**, Elsevier, v. 99, p. 300–313, 2016.

BRAMEL, Julien; SIMCHI-LEVI, David. On the effectiveness of set covering formulations for the vehicle routing problem with time windows. **Operations Research**, INFORMS, v. 45, n. 2, p. 295–301, 1997.

_____. Set-covering-based algorithms for the capacitated VRP. In: THE vehicle routing problem. [S.l.]: SIAM, 2002. P. 85–108.

BRAZDIL, Pavel B; SOARES, Carlos. A comparison of ranking methods for classification algorithm selection. In: SPRINGER. EUROPEAN conference on machine learning. [S.l.: s.n.], 2000. P. 63–75.

BRITO, Julio; MARTÍNEZ, F Javier; MORENO, JA; VERDEGAY, José L. A GRASP–VNS hybrid for the fuzzy vehicle routing problem with time windows. In: SPRINGER. INTERNATIONAL Conference on Computer Aided Systems Theory. [S.l.: s.n.], 2009. P. 825–832.

CABALLERO-MORALES, Santiago-Omar; MARTINEZ-FLORES, Jose-Luis. Helicopter routing model with non-deterministic failure rate for evacuation of multiple oil platforms. **Computers & Industrial Engineering**, Pergamon, v. 139, p. 105669, 2020.

CACERES-CRUZ, Jose; ARIAS, Pol; GUIMARANS, Daniel; RIERA, Daniel; JUAN, Angel A. Rich vehicle routing problem: Survey. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 47, n. 2, p. 1–28, 2014.

CAPRARA, Alberto; TOTH, Paolo; FISCHETTI, Matteo. Algorithms for the set covering problem. **Annals of Operations Research**, Springer, v. 98, n. 1-4, p. 353–371, 2000.

CHRISTIANSEN, Christian H; LYSGAARD, Jens. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. **Operations Research Letters**, Elsevier, v. 35, n. 6, p. 773–781, 2007.

CHRISTOFIDES, N; MINGOZZI, A; TOTH, P. Loading problems. **N. Christofides and al., editors, Combinatorial Optimization**, p. 339–369, 1979.

CHRISTOFIDES, Nicos; EILON, Samuel. An algorithm for the vehicle-dispatching problem. **Journal of the Operational Research Society**, Taylor & Francis, v. 20, n. 3, p. 309–318, 1969.

CLARKE, Geoff; WRIGHT, John W. Scheduling of vehicles from a central depot to a number of delivery points. **Operations research**, Informs, v. 12, n. 4, p. 568–581, 1964.

CORDEAU, Jean-François; GENDREAU, Michel; LAPORTE, Gilbert. A tabu search heuristic for periodic and multi-depot vehicle routing problems. **Networks**, Wiley Online Library, v. 30, n. 2, p. 105–119, 1997.

CORMEN, Thomas H. **Introduction to algorithms**. [S.l.]: MIT press, 2009.

DANTZIG, George B; RAMSER, John H. The truck dispatching problem. **Management science**, Informs, v. 6, n. 1, p. 80–91, 1959.

DAVENDRA, Donald. **Traveling Salesman Problem: Theory and Applications**. [S.l.]: BoD–Books on Demand, 2010.

DESROSIERS, Jacques; DUMAS, Yvan. The shortest path problem for the construction of vehicle routes with pick-up, delivery and time constraints. In: ADVANCES in optimization and control. [S.l.]: Springer, 1988. P. 144–157.

DÍAZ-PARRA, Ocotlán; VANOYE, Jorge Alberto Ruiz; FUENTES-PENNA, Alejandro; LORANCA, Beatriz Bernabe; PÉREZ-ORTEGA, Joaquín; BARRERA-CÁMARA, Ricardo A; VELEZ-DÍAZ, Daniel; PÉREZ-OLGUIN, Nubia B. Oil Platform Transport Problem (OPTP) is NP-hard. **International Journal of Combinatorial Optimization Problems and Informatics**, v. 8, n. 3, p. 2–19, 2017.

DIESTEL, Reinhard. **Graph Theory, 4th Edition**. [S.l.]: Springer, 2012. v. 173. (Graduate texts in mathematics). ISBN 978-3-642-14278-9.

FEO, Thomas A; RESENDE, Mauricio G. C. A probabilistic heuristic for a computationally difficult set covering problem. **Operations Research Letters**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 8, n. 2, p. 67–71, abr. 1989. ISSN 0167-6377. DOI: 10.1016/0167-6377(89)90002-3. Disponível em: <http://dx.doi.org/10.1016/0167-6377(89)90002-3>.

FERNÁNDEZ-CUESTA, Eirik; NORDDAL, Ida Kristine; ANDERSSON, Henrik; FAGERHOLT, Kjetil. Base location and helicopter fleet composition in the oil industry. **INFOR: Information Systems and Operational Research**, Taylor & Francis, v. 55, n. 2, p. 71–92, 2017.

FIALA TIMLIN, M. T.; PULLEYBLANK, W. R. Precedence Constrained Routing and Helicopter Scheduling: Heuristic Design. **Interfaces**, INFORMS, v. 22, 3 jun. 1992. DOI: 10.1287/inte.22.3.100.

GALVÃO, Roberto D; GUIMARÃES, Jesu. The control of helicopter operations in the Brazilian oil industry: Issues in the design and implementation of a computerized system. **European Journal of Operational Research**, Elsevier, v. 49, n. 2, p. 266–270, 1990.

GAVRILAS, Mihai. Heuristic and metaheuristic optimization techniques with application to power systems. **Technical University of Iasi, D. Mangeron Blvd., Iasi, Romania**, 2010.

GOLDBERG, Andrew; RADZIK, Tomasz. **A heuristic improvement of the Bellman-Ford algorithm**. [S.l.], 1993.

GOLDEN, BL; WASIL, EA; KELLY, JP; CHAO, IM. **Metaheuristics in vehicle routing, Fleet management and logistics, TG Crainic and G. Laporte**. [S.l.]: Kluwer, Boston, 1998.

GRIBKOVSKAIA, Irina; HALSKAU, Oyvind; KOVALYOV, Mikhail Y. Minimizing takeoff and landing risk in helicopter pickup and delivery operations. **Omega**, Elsevier, v. 55, p. 73–80, 2015.

GROËR, Chris; GOLDEN, Bruce; WASIL, Edward. A parallel algorithm for the vehicle routing problem. **INFORMS Journal on Computing**, INFORMS, v. 23, n. 2, p. 315–330, 2011.

HADDADENE, Syrine Roufaida Ait; LABADIE, Nacima; PRODHON, Caroline. A GRASP× ILS for the vehicle routing problem with time windows, synchronization and precedence constraints. **Expert Systems with Applications**, Elsevier, v. 66, p. 274–294, 2016.

HALSKAU, Øyvind. Offshore helicopter routing in a hub and spoke fashion: Minimizing expected number of fatalities. **Procedia Computer Science**, Elsevier, v. 31, p. 1124–1132, 2014.

HANSEN, Pierre; MLADENOVIĆ, Nenad. An introduction to variable neighborhood search. In: META-HEURISTICS. [S.l.]: Springer, 1999. P. 433–458.

_____. Variable neighborhood search. In: SEARCH methodologies. [S.l.]: Springer, 2014. P. 313–337.

HERNÁDVÖLGYI, István T. **Automatically generated lower bounds for search**. [S.l.]: Citeseer, 2004.

HOSSEINABADI, Ali Asghar Rahmani; ROSTAMI, Najmeh Sadat Hosseini; KARDGAR, Maryam; MIRKAMALI, Seyedsaeid; ABRAHAM, Ajith. A new efficient approach for solving the capacitated vehicle routing problem using the gravitational emulation local search algorithm. **Applied Mathematical Modelling**, Elsevier, v. 49, p. 663–679, 2017.

IRNICH, Stefan; FUNKE, Birger; GRÜNERT, Tore. Sequential search and its application to vehicle-routing problems. **Computers & Operations Research**, Elsevier, v. 33, n. 8, p. 2405–2429, 2006.

JIN, Jianyong; CRAINIC, Teodor Gabriel; LØKKETANGEN, Arne. A cooperative parallel metaheuristic for the capacitated vehicle routing problem. **Computers & Operations Research**, Elsevier, v. 44, p. 33–41, 2014.

KAMPKE, E.H.; ARROYO, J.E.C.; SANTOS, A.G. dos. Reactive GRASP with path relinking for solving parallel machines scheduling problem with resource-assignable sequence dependent setup times. In: NATURE Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on. [S.l.: s.n.], 2009. P. 924–929. DOI: 10.1109/NABIC.2009.5393873.

KASHAN, Ali Husseinzadeh; ABBASI-POOYA, Amin; KARIMIYAN, Sommayeh. A rig-based formulation and a league championship algorithm for helicopter routing in offshore transportation. In: SPRINGER. PROCEEDINGS of the 2nd international conference on data engineering and communication technology. [S.l.: s.n.], 2019. P. 23–38.

KELLY, James P; XU, Jiefeng. A set-partitioning-based heuristic for the vehicle routing problem. **INFORMS Journal on Computing**, INFORMS, v. 11, n. 2, p. 161–172, 1999.

KIR, Sena; YAZGAN, Harun Reşit; TÜNCEL, Emre. A novel heuristic algorithm for capacitated vehicle routing problem. **Journal of Industrial Engineering International**, Springer, v. 13, n. 3, p. 323–330, 2017.

KWON, Yong-Ju; KIM, Jun-Gyu; SEO, Jeongyeon; LEE, Dong-Ho; KIM, Deok-Soo. A Voronoi tabu search algorithm for the capacitated vehicle routing problem. **Journal of Korean Institute of Industrial Engineers**, Korean Institute of Industrial Engineers, v. 33, n. 4, p. 469–479, 2007.

LAPORTE, Gilbert. The vehicle routing problem: An overview of exact and approximate algorithms. **European journal of operational research**, Elsevier, v. 59, n. 3, p. 345–358, 1992.

LEE, Chou-Yuan; LEE, Zne-Jung; LIN, Shih-Wei; YING, Kuo-Ching. An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem. **Applied Intelligence**, Springer, v. 32, n. 1, p. 88–95, 2010.

LENSTRA, Jan Karel; KAN, AHG Rinnooy. Complexity of vehicle routing and scheduling problems. **Networks**, Wiley Online Library, v. 11, n. 2, p. 221–227, 1981.

LIN, Na; SHI, Yanjun; ZHANG, Tongliang; WANG, Xuping. An effective order-aware hybrid genetic algorithm for capacitated vehicle routing problems in Internet of Things. **IEEE Access**, IEEE, v. 7, p. 86102–86114, 2019.

LINFATI, Rodrigo; ESCOBAR, John Willmer. Reoptimization heuristic for the capacitated vehicle routing problem. **Journal of Advanced Transportation**, Hindawi, v. 2018, 2018.

LÓPEZ-IBÁÑEZ, Manuel; DUBOIS-LACOSTE, Jérémie; CÁCERES, Leslie Pérez; BIRATTARI, Mauro; STÜTZLE, Thomas. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, Elsevier, v. 3, p. 43–58, 2016.

LOURENÇO, Helena Ramalhinho; MARTIN, Olivier C; STÜTZLE, Thomas. Iterated local search: Framework and applications. In: HANDBOOK of metaheuristics. [S.l.]: Springer, 2019. P. 129–168.

LYSGAARD, Jens; LETCHFORD, Adam N; EGLESE, Richard W. A new branch-and-cut algorithm for the capacitated vehicle routing problem. **Mathematical Programming**, Springer, v. 100, n. 2, p. 423–445, 2004.

MACHADO, André Manhães; BOERES, Maria Claudia Silva; ALVARENGA ROSA, Rodrigo de; MAURI, Geraldo Regis. A New Hybridization of Evolutionary Algorithms, GRASP and Set-Partitioning Formulation for the Capacitated Vehicle Routing Problem. In: SPRINGER, CHAM. BRAZILIAN Conference on Intelligent Systems. [S.l.: s.n.], 2020. P. 3–17.

MACHADO, André Manhães; MAURI, Geraldo Regis; BOERES, Maria Claudia Silva; ALVARENGA ROSA, Rodrigo de. A MILP Model and a GRASP Algorithm for the Helicopter Routing Problem with Multi-Trips and Time Windows. In: SPRINGER. INTERNATIONAL Conference on Computational Logistics. [S.l.: s.n.], 2019. P. 204–218.

_____. A new hybrid matheuristic of GRASP and VNS based on constructive heuristics, set-covering and set-partitioning formulations applied to the capacitated vehicle routing problem. **Expert Systems with Applications**, Elsevier, p. 115556, 2021.

MARTINEZ, Clara Marina; CAO, Dongpu. **iHorizon-Enabled Energy management for electrified vehicles**. [S.l.]: Butterworth-Heinemann, 2018.

MENEZES, Fernanda; PORTO, Oscar; REIS, Marcelo L; MORENO, Lorenza; ARAGÃO, Marcus Poggi de; UCHOA, Eduardo; ABELEDO, Hernán; NASCIMENTO, Nelci Carvalho do. Optimizing helicopter transport of oil rig crews at Petrobras. **Interfaces**, INFORMS, v. 40, n. 5, p. 408–416, 2010.

MISHRA, Sashikala; SHAW, Kailash; MISHRA, Debahuti. A new meta-heuristic bat inspired classification approach for microarray data. **Procedia Technology**, Elsevier, v. 4, p. 802–806, 2012.

MORENO, Lorenza; POGGI DE ARAGAO, M; PORTO, Oscar; REIS, ML. Planning Offshore Helicopter Flights on the Campos Basin. **XXXVII Simpósio Brasileiro de Pesquisa Operacional (SBPO), Gramado, Brazil**, 2005.

MOSHEIOV, Gur. Vehicle routing with pick-up and delivery: tour-partitioning heuristics. **Computers & Industrial Engineering**, Elsevier, v. 34, n. 3, p. 669–684, 1998.

MOTTA, Allan; VIEIRA, Roberta; SOLETTI, João. Optimal routing offshore helicopter using Genetic Algorithm. In: IEEE. INFORMATION Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International. [S.l.: s.n.], 2011. v. 2, p. 6–9.

MUTAR, M; BURHANUDDIN, M; HAMEED, A; YUSOF, N; MUTASHAR, H. An efficient improvement of ant colony system algorithm for handling capacity vehicle routing problem. **International Journal of Industrial Engineering Computations**, v. 11, n. 4, p. 549–564, 2020.

NAFSTAD, Gaute Messel; HAUGSETH, Amund; HØYLAND, Vebjørn; STÅLHANE, Magnus. An exact solution method for a rich helicopter flight scheduling problem arising in offshore oil and gas logistics. **Computers & Operations Research**, Elsevier, v. 128, p. 105158, 2021.

NEMENYI, Peter. Distribution-free multiple comparisons. In: INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 2. BIOMETRICS. [S.l.: s.n.], 1962. v. 18, p. 263.

NIU, Yunyun; WANG, Shuo; HE, Juanjuan; XIAO, Jianhua. A novel membrane algorithm for capacitated vehicle routing problem. **Soft Computing**, Springer, v. 19, n. 2, p. 471–482, 2015.

OH, Byung Hoon; KIM, Kwangyeon; CHOI, Han-Lim; HWANG, Inseok. Cooperative multiple agent-based algorithm for evacuation planning for victims with different urgencies. **Journal of Aerospace Information Systems**, American Institute of Aeronautics e Astronautics, v. 15, n. 6, p. 382–395, 2018.

OZDAMAR, Linet. Planning helicopter logistics in disaster relief. **OR spectrum**, Springer, v. 33, n. 3, p. 655–672, 2011.

ÖZDAMAR, Linet; DEMIR, Onur. A hierarchical clustering and routing procedure for large scale disaster relief logistics planning. **Transportation Research Part E: Logistics and Transportation Review**, Elsevier, v. 48, n. 3, p. 591–602, 2012.

ÖZDAMAR, Linet; DEMIR, Onur; BAKIR, Erdinc; YILMAZ, Samet. Hierarchical optimization for helicopter mission planning in large-scale emergencies. **Handbook of emergency response: A human factors and systems engineering approach**, CRC Press, Taylor e Francis, p. 151–174, 2013.

PISINGER, David; ROPKE, Stefan. A general heuristic for vehicle routing problems. **Computers & operations research**, Elsevier, v. 34, n. 8, p. 2403–2435, 2007.

PRINS, Christian. A GRASP× evolutionary local search hybrid for the vehicle routing problem. In: BIO-INSPIRED algorithms for the vehicle routing problem. [S.l.]: Springer, 2009. P. 35–53.

PRINS, Christian; LACOMME, Philippe; PRODHON, Caroline. Order-first split-second methods for vehicle routing problems: A review. **Transportation Research Part C: Emerging Technologies**, Elsevier, v. 40, p. 179–200, 2014.

QIAN, Fubin; GRIBKOVSKAIA, Irina; HALSKAU SR, Øyvind. Helicopter routing in the Norwegian oil industry: Including safety concerns for passenger transport. **International Journal of Physical Distribution & Logistics Management**, Emerald Group Publishing Limited, v. 41, n. 4, p. 401–415, 2011.

QIAN, Fubin; GRIBKOVSKAIA, Irina; LAPORTE, Gilbert; HALSKAU SR, Øyvind. Passenger and pilot risk minimization in offshore helicopter transportation. **Omega**, Elsevier, v. 40, n. 5, p. 584–593, 2012.

REINELT, Gerhard. **The traveling salesman: computational solutions for TSP applications**. [S.l.]: Springer, 2003. v. 840.

RESENDE, Mauricio G. C.; RIBEIRO, Celso C. GRASP: Advances, Hybridizations and Applications. In: GENDREAU, Michel; POTVIN, Jean-Yves (Ed.). **Handbook of Metaheuristics**. 2nd. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 1441916636, 9781441916631.

ROMERO, Martín; SHEREMETOV, Leonid; SORIANO, Angel. A genetic algorithm for the pickup and delivery problem: An application to the helicopter offshore transportation. In: THEORETICAL advances and applications of fuzzy logic and soft computing. [S.l.]: Springer, 2007. P. 435–444.

ROSA, Rodrigo de Alvarenga; MACHADO, André Manhães; RIBEIRO, Glaydston Mattos; MAURI, Geraldo Regis. A mathematical model and a Clustering Search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas. **Computers & Industrial Engineering**, Elsevier, v. 101, p. 303–312, 2016.

ROSERO, VB; TORRES, Fidel. Ant colony based on a heuristic insertion for a family of helicopter routing problems. In: CITESEER. CONFERENCE Proceedings from the Third International Conference on Production Research-Amerericas Region. [S.l.: s.n.], 2006. P. 1–12.

SAVELSBERGH, Martin WP; SOL, Marc. The general pickup and delivery problem. **Transportation science**, INFORMS, v. 29, n. 1, p. 17–29, 1995.

SBAI, Ines; KRICHEN, Saoussen; LIMAM, Olfa. Two meta-heuristics for solving the capacitated vehicle routing problem: the case of the Tunisian Post Office. **Operational Research**, Springer, p. 1–43, 2020.

SIERKSMA, Gerard; TIJSSEN, Gert A. Routing helicopters for crew exchanges on off-shore locations. **Annals of Operations Research**, Springer, v. 76, p. 261–286, 1998.

SIPSER, Michael. **Introduction to the Theory of Computation**. [S.l.]: Cengage learning, 2012.

SÖRENSEN, Kenneth. Metaheuristics—the metaphor exposed. **International Transactions in Operational Research**, Wiley Online Library, v. 22, n. 1, p. 3–18, 2015.

SZETO, Wai Yuen; WU, Yongzhong; HO, Sin C. An artificial bee colony algorithm for the capacitated vehicle routing problem. **European Journal of Operational Research**, Elsevier, v. 215, n. 1, p. 126–135, 2011.

TAVAKKOLI-MOGHADDAM, Reza; SAFAEI, Nima; GHOLIPOUR, Y. A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length. **Applied Mathematics and Computation**, Elsevier, v. 176, n. 2, p. 445–454, 2006.

TEOH, Boon Ean; PONNAMBALAM, Sivalinga Govinda; KANAGARAJ, Ganesan. Differential evolution algorithm with local search for capacitated vehicle routing problem. **International Journal of Bio-Inspired Computation**, Inderscience Publishers (IEL), v. 7, n. 5, p. 321–342, 2015.

TOFFOLO, Túlio AM; VIDAL, Thibaut; WAUTERS, Tony. Heuristics for vehicle routing problems: Sequence or set optimization? **Computers & Operations Research**, Elsevier, v. 105, p. 118–131, 2019.

TORRES, Fidel. Algoritmo genético basado en una heurística de inserción para el problema de ruteo de helicópteros. PYLO, 2006.

TOTH, Paolo; VIGO, Daniele. Models, relaxations and exact approaches for the capacitated vehicle routing problem. **Discrete Applied Mathematics**, Elsevier, v. 123, n. 1-3, p. 487–512, 2002.

_____. The granular tabu search and its application to the vehicle-routing problem. **Informs Journal on computing**, INFORMS, v. 15, n. 4, p. 333–346, 2003.

_____. **The vehicle routing problem**. [S.l.]: SIAM, 2002.

_____. **Vehicle routing: problems, methods, and applications**. [S.l.]: SIAM, 2014.

VAN URK, Rick; MES, Martijn RK; HANS, Erwin W. Anticipatory routing of police helicopters. **Expert systems with applications**, Elsevier, v. 40, n. 17, p. 6938–6947, 2013.

VAN WASSENHOVE, Luk N. Humanitarian aid logistics: supply chain management in high gear. **Journal of the Operational research Society**, Springer, v. 57, n. 5, p. 475–489, 2006.

VANDERBEI, Robert J. **Linear programming: foundations and extensions**. [S.l.]: Springer Nature, 2020. v. 285.

VELASCO, Nubia; CASTAGLIOLA, Philippe; DEJAX, Pierre; GUÉRET, Christelle; PRINS, Christian. A memetic algorithm for a pick-up and delivery problem by helicopter. In: BIO-INSPIRED algorithms for the vehicle routing problem. [S.l.]: Springer, 2009. P. 173–190.

VIDAL, Thibaut; CRAINIC, Teodor Gabriel; GENDREAU, Michel; LAHRICHI, Nadia; REI, Walter. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. **Operations Research**, INFORMS, v. 60, n. 3, p. 611–624, 2012.

WANG, Chung-Ho; LU, Jiu-Zhang. A hybrid genetic algorithm that optimizes capacitated vehicle routing problems. **Expert Systems with Applications**, Elsevier, v. 36, n. 2, p. 2921–2936, 2009.

XAVIER, Iran R; BANDEIRA, Renata AM; BANDEIRA, Adriano PF; CAMPOS, Vania BG; SILVA, Leandro O. Planning the use of helicopters in distribution of supplies in response operations of natural disasters. **Transportation Research Procedia**, Elsevier, v. 47, p. 633–640, 2020.

XIAO, Yiyong; ZHAO, Qiuhong; KAKU, Ikou; XU, Yuchun. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. **Computers & operations research**, Elsevier, v. 39, n. 7, p. 1419–1431, 2012.

ZHOU, Yongquan; XIE, Jian; ZHENG, Hongqing. A hybrid bat algorithm with path relinking for capacitated vehicle routing problem. **Mathematical Problems in Engineering**, Hindawi, v. 2013, 2013.

ZHU, Wenbin; QIN, Hu; LIM, Andrew; WANG, Lei. A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP. **Computers & Operations Research**, Elsevier, v. 39, n. 9, p. 2178–2195, 2012.

APÊNDICE A – Definições e Conceitos Gerais

Neste apêndice são apresentados as definições e os conceitos gerais que são utilizados no trabalho. Inicialmente, define-se o conceito de grafo. Posteriormente, utilizando este objeto matemático, definem-se caminhos, ciclos, percursos e distâncias que podem ser associadas a este objeto. Em seguida, apresentam-se os modelos de programação. Por fim, definem-se as funções assintóticas para análise de algoritmos.

A.1 Grafos

Um grafo não orientado ou simplesmente um grafo, é um par de conjuntos $G = (V, E)$ tal que $E \subseteq V^2$. Os elementos de V são chamados de vértices (ou nós) do grafo G, enquanto os elementos de E são as arestas (ou arcos, quando o grafo é orientado) de G. O conjunto de vértices de G é denotado como $V(G)$ e o conjunto de arestas é designado como $E(G)$ (DIESTEL, 2012).

Os números de vértices e arestas de um grafo $G = (V, E)$ são definidos como $|V| = n$ e $|E| = m$, respectivamente. A aresta $e \in E$ é representada como $\{v_i, v_j\}$ ou simplesmente $v_i v_j$. Dois vértices v, w são adjacentes ou vizinhos se $e = \{v, w\} \in E$. O vértice v é incidente na aresta e se $v \in e$ (DIESTEL, 2012).

O conjunto de vizinhos de um vértice $v \in V$ no grafo $G = (V, E)$ é denotado como $N(v)$. De forma genérica, para $U \subseteq V$, os vizinhos dos vértices de U em V são definidos como (DIESTEL, 2012):

$$N(U) = \{w | \{w, u\} \in E \wedge u \in U \wedge w \notin U\}$$

O grau $d(v)$ do vértice $v \in V$ no grafo $G = (V, E)$ é o número de arestas que incidem em v. Por definição, o grau do vértice v é igual ao número de vizinhos de v (DIESTEL, 2012).

A Figura 18 apresenta um grafo $G = (V, E)$, em que $V = \{a, b, c, d, e, f, g\}$ e $E = \{\{a, b\}, \{b, c\}, \{c, a\}, \{c, d\}, \{d, e\}, \{e, f\}, \{e, g\}\}$.

Um grafo orientado (ou digrafo) é um par de conjuntos $G = (V, E)$ munido de duas funções $init : E \to V$ e $ter : E \to V$, as quais designam para todo arco $e \in E$ um vértice inicial $v_i = init(e) \in V$ e um vértice terminal $v_j = ter(e) \in V$. O arco e é dito ser direcionado do vértice v_i ao vértice v_j. Se $v_i = v_j$, o arco é chamado de laço (DIESTEL, 2012).

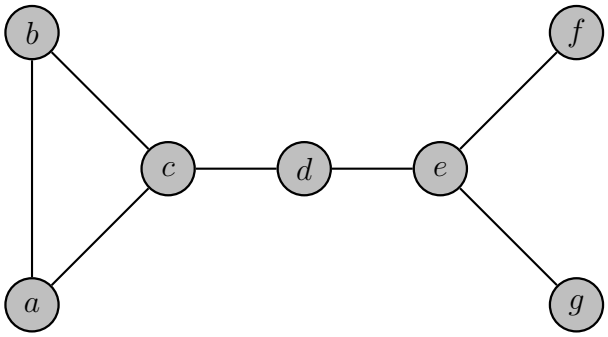


Figura 18 – Grafo $G = (V, E)$ não orientado.

Fonte: Autor

A Figura 19 apresenta um grafo orientado $G = (V, E)$, em que:

$$
\begin{aligned}
V &= \{a, b, c\} \\
E &= \{\{a, b\}, \{b, c\}, \{c, a\}\} \\
init &= \{(\{a, b\}, a), (\{b, c\}, b), (\{c, a\}, c)\} \\
ter &= \{(\{a, b\}, b), (\{b, c\}, c), (\{c, a\}, a)\}
\end{aligned}
$$

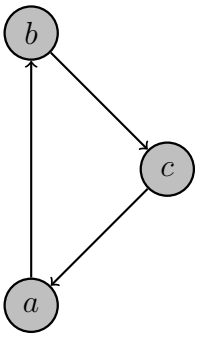


Figura 19 – Grafo $G = (V, E)$ orientado.

Fonte: Autor

O grafo $G = (V, E)$ é ponderado se para toda aresta $e \in E$ é designado um número $p(e) \in \mathbb{R}$. O número $p(e)$ é chamado peso da aresta. O grafo ponderado G é denotado como a tupla ordenada $G = (V, E, p)$. A função $p : E \to \mathbb{R}$ pode ser considerada como um vetor cujas coordenadas são indexadas pelo conjunto de arestas E (BONDY; MURTY, 2007).

A Figura 20 apresenta uma versão ponderada do grafo $G = (V, E)$ da Figura 18, em que a função p é igual a

$$
\begin{aligned}
\{(\{a, b\}, 1{,}0), (\{b, c\}, 3{,}2), (\{c, a\}, 4{,}5), (\{c, d\}, 2{,}1), \\
(\{d, e\}, 3{,}0), (\{e, f\}, 1{,}4), (\{e, g\}, 1{,}0)\}
\end{aligned}
$$

Um grafo $H = (V_H, E_H)$ é um subgrafo de $G = (V, E)$ se $V_H \subseteq V$ e $E_H \subseteq E$ (BONDY; MURTY, 2007).

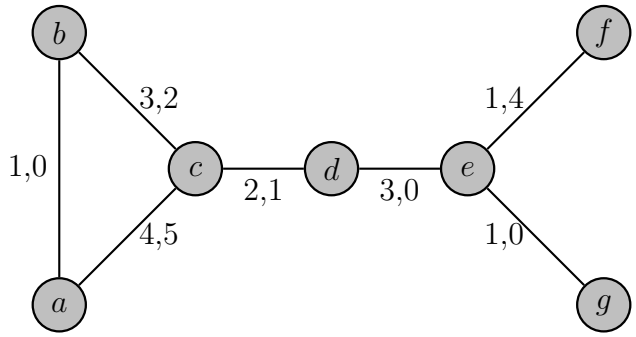


Figura 20 – Grafo $G = (V, E)$ ponderado.

Fonte: Autor

Um grafo $G = (V, E)$ é conexo se para cada par de vértices $v, w \in V$ existe um caminho entre v e w. Um grafo não conexo pode ser dividido em subgrafos (disjuntos e conexos) chamados componentes (BONDY; MURTY, 2007).

A.2 Caminhos, Percursos, Ciclos e Distância

Um percurso de tamanho t num grafo $G = (V, E)$ é uma sequência alternada e não vazia $v_0e_0v_1 \cdots e_{t-1}v_t$ de vértices e arestas de G tal que $e_i = v_iv_{i+1}$ para todo $i < t$. Se $v_0 = v_t$, o percurso é fechado. Se todos os vértices do percurso são distintos, então o percurso é um caminho em G (DIESTEL, 2012). Quando não há arestas paralelas, o caminho pode ser abreviado pela designação somente dos vértices $v_0v_1 \cdots v_{t-1}v_t$.

Um ciclo é um caminho alternado de vértices e arestas $v_0e_0v_1 \cdots e_{t-1}v_t$ em que o primeiro e último vértices são iguais, ou seja, $v_0 = v_t$. Se designa por $k - ciclo$ um ciclo que contenha k arestas.

A distância $d(v, w)$ em $G = (V, E)$ entre dois vértices $v, w \in V$ é o número de arestas do menor caminho entre v e w em G. Caso não exista um caminho entre v e w, então $d(v, w) = \infty$ (DIESTEL, 2012). O menor caminho entre dois vértices v e w também é designado como geodésica.

Um caminho p é dito hamiltoniano se ele é formado por todos os vértices V de um grafo $G = (V, E)$ e $|p| = |V|$, ou seja, o caminho p visita cada vértice de V uma única vez (DIESTEL, 2012).

A.3 Programação Matemática

Um modelo de PL é um método de otimização para funções lineares sujeita a equações ou inequações lineares. Dados o vetor x de variáveis reais de dimensão n, os vetores c e b de coeficientes e a matriz A de coeficientes, então o PL pode ser definido como (BAZARAA;

JARVIS; SHERALI, 2011):

$$\text{Maximizar } c^T x \tag{A.1}$$

$$\text{Sujeito a: } Ax \leq b \tag{A.2}$$

$$x \geq 0 \tag{A.3}$$

$$x \in \mathbb{R}^n \tag{A.4}$$

No caso de x ser um vetor definido somente sobre os conjuntos dos inteiros, o PL se transforma num modelo de PI. Se o vetor x contiver variáveis inteiras e reais, então o PL converte-se num modelo de PLIM.

A.4 Notação Assintótica e Problemas Computacionais

Na análise de algoritmos, as funções assintóticas são utilizadas para avaliar a complexidade de algoritmos. Estas funções impõem tetos e/ou pisos para o tempo de execução do algoritmo em termos de um ou mais números naturais, os quais representam a entrada deste algoritmo. As funções assintóticas utilizadas para a análise dos algoritmos apresentados nesta tese são: a notação assintótica Θ, a notação assintótica Ω e a notação assintótica O.

Sejam $f : \mathbb{N} \to \mathbb{R}$ e $g : \mathbb{N} \to \mathbb{R}$ funções do conjunto dos naturais no conjunto dos números reais. Afirma-se que $f(x)$ é $O(g(x))$ se existem constantes positivas $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{R}^+$ de forma que (CORMEN, 2009):

$$0 \leq f(x) \leq cg(x) \tag{A.5}$$

quando $x > n_0$. Dizemos que $f(x)$ é $\Omega(g(x))$ se existem constantes positivas $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{R}^+$ de forma que (CORMEN, 2009):

$$0 \leq cg(x) \leq f(x) \tag{A.6}$$

quando $x > n_0$. Dizemos que $f(x)$ é $\Theta(g(x))$ se existem constantes positivas $c_1 \in \mathbb{R}^+$, $c_2 \in \mathbb{R}^+$ e $n_0 \in \mathbb{R}^+$ de forma que (CORMEN, 2009):

$$c_1 g(x) \leq f(x) \leq c_2 g(x) \tag{A.7}$$

quando $x > n_0$.

Na teoria da computação (SIPSER, 2012), define-se P como o conjunto de problemas que podem ser resolvidos, em tempo polinomial, por uma máquina de Turing determinística. Além disso, define-se NP como o conjunto de problemas de decisão que podem ser resolvidos, em tempo polinomial, por uma máquina de Turing não determinística. Logo, $P \subset NP$. Dado um problema C, ele é designado como NP-completo se:

- C está em NP (condição de pertencimento);
- Todo problema em NP pode ser reduzido a C em tempo polinomial (condição de redutibilidade).

Por fim, um problema C é NP-difícil se ele atende somente a condição de redutibilidade, mas não a condição de pertencimento.

APÊNDICE B – Exemplo de aplicação da matheurística MGV-C

Nos próximos parágrafos é esboçado o comportamento da matheurística MGV-C para uma problema hipotético no grafo $G(V, E)$. Para os parâmetros de entrada do MGV-C, suponha $M_s = 50$, ou seja, o número máximo de iterações sem aprimoramento no VNS é fixado como 50. No início do algoritmo, as variáveis são inicializadas:

Variável	Valor
i_c	0
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	$\emptyset$

O algoritmo entra no laço exterior do GRASP (linha 5) e inicia a fase construtiva do GRASP. O procedimento PCC-SOLUCAO constrói uma solução inicial no espaço $\Omega(\text{TSP})$ usando as soluções no conjunto π_{PCC}. Na primeira iteração do laço, $\pi_{PCC} = \emptyset$, logo $T = \emptyset$:

Variável	Valor
i_c	0
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	$\emptyset$
T	$\emptyset$

A heurística construtiva é chamada para completar a solução T (linha 7). Na primeira iteração, como $T = \emptyset$, a heurística construtiva constrói uma solução do zero. O resultado é uma solução $T = T_1$ de $\Omega(\text{TSP})$:

Variável	Valor
i_c	0
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	$\emptyset$
T	T_1

A solução T_1 é convertida para uma solução CVRP usando a função SPLIT-CVRP na linha 8. Essa solução é armazenada em S. Esse é o último passo da fase construtiva do GRASP.

Variável	Valor
i_c	0
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	$\emptyset$
T	T_1
S	S_1

Inicia-se a fase de busca local do GRASP. Na linha 9, o método de busca local BL-CVRP é utilizado para encontrar uma solução localmente ótima S_1^* a partir de S_1. S_1^* é armazenada em S:

Variável	Valor
i_c	0
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	$\emptyset$
T	T_1
S	S_1^*

A partir da solução localmente ótima em S, começa a segunda parte da fase local do GRASP com a meta-heurística VNS. Inicializa-se a variável π_S e entra-se no laço interno do VNS na linha 11. A variável π_S armazena todas as soluções localmente ótimas geradas dentro desse laço interno:

Variável	Valor
i_c	0
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	$\emptyset$
T	T_1
S	S_1^*
π_S	$\emptyset$

No laço do VNS, a variável i_c é incrementada. A solução S é convertida para T^* via o método CONCAT. No espaço $\Omega(\mathrm{TSP})$ essa solução T^* é perturbada pela função PERTUBA-CVRP e convertida novamente para S^* com o método SPLIT-CVRP. Seja S_2 essa solução em S^*:

Variável	Valor
i_c	1
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	$\emptyset$
T	T_1
S	S_1^*
π_S	$\emptyset$
S^*	S_2

Na solução S_2 em S^* é aplicado a busca local BL-CVRP. A solução resultante é armazenada em S^* e adicionada ao conjunto π_S:

Variável	Valor
i_c	1
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	$\emptyset$
T	T_1
S	S_1^*
π_S	$\{S_2^*\}$
S^*	S_2^*

Em seguida, o laço do VNS realiza as atualizações das variáveis S, $\bar{S}$ e i_c se necessário. Supondo que sim, i_c é redefinida como zero e a solução S_2^* é armazenada em S e $\bar{S}$:

Variável	Valor
i_c	0
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	S_2^*
T	T_1
S	S_2^*
π_S	$\{S_2^*\}$
S^*	S_2^*

O laço do VNS irá executar M_v vezes. Sempre que ele encontra uma solução localmente ótima, ele adiciona ao conjunto π_S. Após o término do VNS, um possível estado das variáveis é:

Variável	Valor
i_c	5
π_{PCC}	$\emptyset$
π_{PPC}	$\emptyset$
$\bar{S}$	S_3^*
T	T_1
S	S_7^*
π_S	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
S^*	S_t^*

Veja que no caso anterior $i_c = 5$. Logo, nas últimas cinco iterações do VNS, ele não foi capaz de encontrar o valor de um novo limitante inferior para o problema. Terminado o VNS, finaliza-se o laço do GRASP com a atualização de π_{PPC} e π_{PCC}. Verifica-se também se $i_c > M_s$. No exemplo, essa condição não é atendida.

Variável	Valor
i_c	5
π_{PCC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
π_{PPC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
$\bar{S}$	S_3^*
T	T_1
S	S_7^*
π_S	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
S^*	S_t^*

A partir daí, começa-se o próximo laço do GRASP. Como $\pi_{PCC} \neq \emptyset$, o procedimento SOLUCAO-INICIAL constrói uma solução inicial não vazia usando os elementos do conjunto π_{PCC}:

Variável	Valor
i_c	5
π_{PCC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
π_{PPC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
$\bar{S}$	S_3^*
T	T_2
π_S	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
S^*	S_t^*

A heurística construtiva é chamada para completar a solução $T = T_2$ se necessário (linha 7). Supondo que sim, é gerada a solução T_3:

Variável	Valor
i_c	5
π_{PCC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
π_{PPC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
$\bar{S}$	S_3^*
T	T_3
π_S	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
S^*	S_t^*

A seguir, o algoritmo continua conforme relatado anteriormente com a aplicação de BL-CVRP e VNS. Dentro desse, as soluções localmente ótimas são armazenadas no conjunto π_S. Entretanto, o esboço anterior possui uma exceção após a execução do VNS, quando $i_c \geq M_s$. Nesse caso, π_{PCC} é zerado no final do laço do GRASP. Assim, suponha $i_c = 51$ ao final do VNS:

Variável	Valor
i_c	51
π_{PCC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_t^*\}$
π_{PPC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_t^*\}$
$\bar{S}$	S_{29}^*
T	T_8
π_S	$\{S_i^*, , \cdots, S_t^*\}$
S^*	S_t^*

Nesse estado de variáveis, a estrutura de controle ao final do GRASP (linha 28) zera i_c e π_{PCC}, pois $M_s = 50$ e $i_c > M_s$:

Variável	Valor
i_c	0
π_{PCC}	$\emptyset$
π_{PPC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
$\bar{S}$	S_3^*
T	T_3
π_S	$\{S_2^*, S_3^*, S_5^*, \cdots, S_k^*\}$
S^*	S_t^*

Isso é realizado para que os métodos PCC-SOLUCAO e HEURISTICA-CONSTRUTIVA não fiquem presos num mínimo local. Assim, nos laços seguintes a isso, o conjunto π_{PCC} começa a ser preenchido novamente do zero.

Após todas as M_i execuções do GRASP, o último passo é gerar a solução final S^* utilizando o conjunto π_{PPC} (linha 33):

Variável	Valor
i_c	26
$\vdots$	
π_{PPC}	$\{S_2^*, S_3^*, S_5^*, \cdots, S_u^*\}$
$\vdots$	

A solução S^* gerada usando o problema PPC é a solução final da matheurística MGV-C.

APÊNDICE C – Análise de Componentes do MGV-C

Este apêndice analisa a contribuição dos principais componentes do MGV-C em termos de tempo de execução e qualidade da solução gerada. Para essa avaliação, componentes da matheurística foram desativados e a heurística modificada foi reaplicada nas instâncias utilizadas no ajuste fino. A Tabela 34 apresenta essa análise. A coluna **MGV-C** apresenta os resultados da matheurística conforme exposto no Capítulo 7. As colunas **PPC**, **2-OPT**, **CROSSOVER** e **OR-OPT** representam os resultados quando os componentes PPC, o movimento 2-OPT, o movimento CROSSOVER e o movimento OR-OPT são excluídos do método. Esses componentes foram escolhidos porque eles demandam maior poder computacional (PPC) ou porque todos precisam executar para escolher uma nova solução (2-OPT, CROSSOVER e OR-OPT). As colunas **AVG** e **T(s)** designam o valor médio e o tempo médio da matheurística conforme apresentado no Capítulo 7. As colunas R_{avg} e R_T representam, respectivamente, o valor médio relativo e o tempo médio relativo da heurística com os componentes desativados em relação à matheurística MGV-C com todos os componentes ativos. A linha X designa a média dos valores apresentados nas colunas. Nos testes de desativação dos componentes, cada versão foi executada 10 vezes.

Tabela 34 – Análise de Componentes do MGV-C

Instância	MGV-C		PPC		2-OPT		CROSSOVER		OR-OPT	
	AVG	T(s)	R_{avg}	R_T	R_{avg}	R_T	R_{avg}	R_T	R_{avg}	R_T
A-n45-k6	945,2	8,51	102,04	97,18	100,44	112,34	100,04	122,91	100,00	105,72
A-n80-k10	1763	42,52	100,98	127,00	100,27	115,95	100,03	105,60	100,02	106,76
B-n57-k7	1155	13,57	178,56	92,11	100,10	103,98	100,11	137,73	100,00	108,73
B-n68-k9	1272	122,1	101,25	33,66	100,44	103,85	100,23	108,57	100,27	101,64
X	1283,8	46,675	120,71	87,49	100,31	109,03	100,102	118,70	100,07	105,71

Analisando a Tabela 34, nota-se que o componente que mais contribui para a qualidade da solução é o PPC, em especial para a instância B-n57-k7. No caso, a nova média da solução é 78,56% pior que a gerada pelo MGV-C. Em geral, a remoção do PPC diminui o tempo de execução, especialmente para a instância B-n68-k9. Entretanto, remover o PPC para reduzir o tempo computacional impacta o número de ótimos alcançados. No caso das instâncias A-n80-k10 e B-n68-k9, a sua remoção faria que o MGV-H não encontrasse o ótimo conhecido em todas as execuções. A desativação dos componentes 2-OPT e CROSSOVER da busca local, também aumentam o valor médio obtido, mas com uma piora menos significativa, inferior a 1%. Note que, entretanto, o tempo de execução aumenta. Isso pode ocorrer porque com menos movimentos é necessário produzir mais vizinhanças para obter um mínimo local. No caso do

MGV-C, ele trabalha com estruturas que otimizam o cálculo dos movimentos da busca local, mas em contrapartida tem-se o custo de calcular os vetores auxiliares toda vez que uma nova solução é produzida. O mesmo raciocínio pode ser aplicado ao movimento OR-OPT da busca local, sendo que há pouca contribuição para o valor da solução, mas uma média de tempo superior a 5% quando ele é desativado na heurística.

APÊNDICE D – Função BL-HRP-CUSTO

Neste apêndice apresenta-se a função BL-HRP-CUSTO, a qual avalia o ganho da reinserção de um par de nós *pickup-delivery* (a, a') em função da restrição de interesse γ. A ideia desse método é, sem a necessidade de recalcular todos os parâmetros das rotas envolvidas na mudança, determinar se a reinserção melhora o custo da solução. Para isso, usa-se os vetores auxiliares, conforme apresentado na Seção 6.4. O Algoritmo 39 mostra o método BL-HRP-CUSTO. Ele recebe as rotas r_i, r_j, o par de nós *pickup-delivery* (a, a') em r_i, os nós b, c em r_j e a restrição de interesse γ. O objetivo é avaliar se a remoção de (a, a') de r_i, a inserção de a após b em r_j e a inserção de a' após c em r_j diminuem o número de restrições nas rotas r_i, r_j ou o valor do custo das rotas, sempre considerando a restrição de interesse γ. Para determinar esse custo, para cada restrição que pode ser relaxada no HRP (tempo, janela de tempo, peso, combustível e assento) um método específico é definido para avaliação de uma penalização da reinserção. Esses métodos são o PEN-TEMPO para o tempo, PEN-JANELA para a janela de tempo, PEN-PESO para o peso, PEN-COMBUSTIVEL para o combustível e PEN-ASSENTO para o assento. Como regra, esses métodos retornam um valor negativo se alguma melhora é obtida (diminuição do valor absoluto de uma inviabilidades ou remoção de uma inviabilidade), positivo (aumento de uma inviabilidade) ou infinito, no caso que um nó que não tinha inviabilidade passa a ter com a reinserção. Além disso, existe o método DISTANCIA-AERONAVES que retorna a variação da distância e a variação no custo de aeronaves utilizadas. O retorno do método BL-HRP-CUSTO é a soma dos valores retornados pelas funções PEN-TEMPO, PEN-JANELA, PEN-PESO, PEN-COMBUSTIVEL, PEN-ASSENTO e DISTANCIA-AERONAVES.

Algoritmo 39: BL-HRP-CUSTO$(r_i, r_j, a, a', b, c, \gamma)$

entrada: Duas rotas r_i, r_j, uma par de nós (a, a') *pickup-delivery* em r_i, dois nós b, c em r_j e uma restrição γ de interesse
saída : Um número real

1 $v \leftarrow 0 +$ PEN-TEMPO$(r_i, r_j, a, a', b, c, \gamma)$;
2 $v \leftarrow v +$ PEN-JANELA$(r_i, r_j, a, a', b, c, \gamma)$;
3 $v \leftarrow v +$ PEN-PESO$(r_i, r_j, a, a', b, c, \gamma)$;
4 $v \leftarrow v +$ PEN-COMBUSTIVEL$(r_i, r_j, a, a', b, c, \gamma)$;
5 $v \leftarrow v +$ PEN-ASSENTO$(r_i, r_j, a, a', b, c, \gamma)$;
6 $v \leftarrow v +$ DISTANCIA-AERONAVES(r_i, r_j, a, a', b, c);
7 **retorne** v;

Os métodos PEN-TEMPO, PEN-JANELA, PEN-PESO, PEN-COMBUSTIVEL e PEN-ASSENTO utilizados em BL-HRP-CUSTO avaliam o impacto da reinserção do par de nós *pickup-delivery* em função de cada restrição relaxada do HRP. Para isso, o método PEN-PROPRIEDADE apresentado no Algoritmo 40 é utilizado para avaliar, genericamente, a alteração de cada res-

trição. O método PEN-PROPRIEDADE recebe como entrada um número real i, um número real m, um número real v e um valor verdade t. Na lógica do método, i representa o valor atual de alguma propriedade de interesse avaliada na reinserção dos nós *pickup-delivery* (a, a'), m representa o valor máximo que a propriedade pode ter em qualquer solução viável do HRP e v representa a alteração que é causada pela reinserção de (a, a') no valor de i na solução. O método possui três casos. No primeiro caso, linha 2, $i < m$, ou seja, a solução não possui inviabilidade para i e, após a alteração, continua viável, $i + v \leq m$. Nesse caso, o valor retornado pelo método é 0, pois a solução continua viável. No segundo caso, linha 5, i é viável antes da alteração, mais inviável após a reinserção. Assim, o valor retornado pelo método é ∞. Isso garante que nenhum movimento possa tornar um nó viável em inviável. O terceiro caso acontece quando $t = \text{TRUE}$. Isso é ativado quando a restrição de interesse é compatível com a função que está avaliando uma das propriedades relaxadas do HRP. Nesse caso, calcula-se o valor de retorno da função em função do produto de uma constante Y_1 e a variação v. Note que, caso $t \neq \text{TRUE}$, o método retorna 0, ou seja, nenhuma avaliação é realizada numa propriedade já inviável quando ela ainda não é de interesse.

Algoritmo 40: PEN-PROPRIEDADE(i, m, v, t)

entrada: Um número real i, um número real m, um número real v e um valor verdade t
saída : Um número real

```
1  c ← 0;
2  se i ≤ m e i + v ≤ m então
3  |  c ← 0;
4  fim
5  se então i ≤ m e i + v > m então
6  |  c ← +∞;
7  fim
8  se então t = TRUE então
9  |  se v < 0 então
10 |  |  v ← max(v, m − i);
11 |  fim
12 |  c ← Y_1 * v;
13 fim
14 retorne c;
```

O Algoritmo 41 apresenta o método PEN-TEMPO que determina a penalização da reinserção dos nós em relação às restrições de tempo. Ele recebe as rotas r_i, r_j, o par de nós *pickup-delivery* (a, a') em r_i, os nós b, c em r_j e a restrição de interesse γ. Entre as linhas 1 e 4, o método avalia o impacto da remoção de (a, a') no tempo da rota r_i. Inicialmente, registra-se o tempo t_i atual de r_i. Em seguida, a variação v_i do tempo de execução da rota r_i quando (a, a') é removido. Note que isso pode ser feito sem a necessidade de refazer a rota r_i. Após isso, registra-se o tempo máximo permitido m_i para a rota r_i. Em posse desses valores, utiliza-se o método PEN-PROPRIEDADE para calcular penalização da rota r_i com a remoção de (a, a'). O mesmo

raciocínio é aplicado para a inserção de (a, a') em r_j. O resultado final do método é a soma dos valores retornados por PEN-PROPRIEDADE.

Algoritmo 41: PEN-TEMPO$(r_i, r_j, a, a', b, c, \gamma)$

entrada: Duas rotas r_i, r_j, uma par de nós (a, a') *pickup-delivery* em r_i, dois nós b, c em r_j e uma restrição γ de interesse
saída : Um número real

1 Seja t_i o tempo de atendimento de r_i;
2 Seja v_i a variação do tempo total de atendimento de r_i ao remover (a, a') de r_i;
3 Seja m_i o tempo máximo permitido para a rota r_i;
4 $c_i \leftarrow$ PEN-PROPRIEDADE$(t_i, m_i, v_i, \gamma_T = \gamma)$;
5 Seja t_j o tempo de atendimento de r_j;
6 Seja v_j a variação do tempo total de atendimento de r_j ao inserir a' após c e a após b;
7 Seja m_j o tempo máximo permitido para a rota r_j;
8 $c_j \leftarrow$ PEN-PROPRIEDADE$(t_j, m_j, v_j, \gamma_T = \gamma)$;
9 **retorne** $c_i + c_j$;

O Algoritmo 42 apresenta o método PEN-ASSENTO que determina a penalização da reinserção dos nós em relação às restrições de ocupação de assentos. Ele recebe as rotas r_i, r_j, o par de nós *pickup-delivery* (a, a') em r_i, os nós b, c em r_j e a restrição de interesse γ. Entre as linhas 1 e 4, o método avalia o impacto da remoção de (a, a') no número de assentos utilizados entre a e a' em r_i. Na remoção de nós, há duas possibilidades para calcular a remoção. Na primeira, o número de assentos utilizados aumenta entre a e a'. Isso pode ser verificado usando o vetor auxiliar de número de assentos máximos $Q^s_{r_i}$. Entretanto, isso só é verdade se esse máximo entre a e a' também for o máximo entre o início da rota e a'. O objetivo dessa estratégia é permitir a remoção de nós (a, a') que não sejam inviáveis para assentos, mas que tenham alguma inviabilidade no caminho que conecta a e a'. No segundo caso, utiliza-se o número de assentos de a como referência. Assim, inicialmente, determina-se o nó k_{a-1} anterior a a em r_i. Verifica-se se entre k_{a-1} e a' ocorreu aumento de número de assentos utilizados. Se sim, então o valor $Q^s_{r_i}[k'_a]$ é registrado na variável s_i. Caso contrário, o valor de número de assentos de a é registrado em s_i. Após isso, registra-se o número máximo permitido de assentos m_i para a rota r_i. A variação de assentos é o valor negativo do número de assentos v necessário para atender a, pois esse é o número de assentos liberados ao remover a de r_i. Em posse desses valores, utiliza-se o método PEN-PROPRIEDADE para calcular a penalização da rota r_i com a remoção de (a, a'). No caso da inserção, verifica-se o número máximo de assentos utilizados entre b e c na rota r_j. Esse valor é mantido pelo método OR-OPT-HRP (Algoritmo 30) ao avaliar as possibilidades de inserção em r_j. Assim, não é necessário recalculá-lo no método PEN-ASSENTO. Além disso, registra-se o número máximo permitido de assentos m_j para a rota r_j e a variação do número de assentos como o número de assentos para atender a, pois é o caso de inserção de a, a' em r_j. A partir desses valores, calcula-se PEN-PROPRIEDADE para a inserção de (a, a') em r_j entre b e c. O valor final do método é a soma dos valores retornados

por PEN-PROPRIEDADE para r_i e r_j.

Algoritmo 42: PEN-ASSENTO$(r_i, r_j, a, a', b, c, \gamma)$

entrada: Duas rotas r_i, r_j, uma par de nós (a, a') *pickup-delivery* em r_i, dois nós b, c em r_j e uma restrição γ de interesse
saída : Um número real

Seja k_{a-1} o índice do nó anterior a a em r_i para o vetor $Q^s_{r_i}$;
Seja k'_a o índice de a' para o vetor $Q^s_{r_i}$;
Seja s_a o número de assentos ocupados no nó a em r_i;
Seja v o número de assentos necessário para atender a;
$s_i \leftarrow 0$;
se $Q^s_{r_i}[k_{a-1}] < Q^s_{r_i}[k'_a]$ **então**
    $s_i \leftarrow Q^s_{r_i}[k'_a]$;
fim
senão
    $s_i \leftarrow s_a$;
fim
Seja m_i a quantidade máxima de assentos do helicóptero que atende r_i;
$c_i \leftarrow$ PEN-PROPRIEDADE$(s_i, m_i, -v, \gamma_S = \gamma)$;
Seja s_j o número máximo de assentos usados entre b e c em r_j;
Seja m_j a quantidade máxima de assentos do helicóptero que atende r_j;
$c_j \leftarrow$ PEN-PROPRIEDADE$(s_j, m_j, v, \gamma_S = \gamma)$;
retorne $c_i + c_j$;

O Algoritmo 43 apresenta o método PEN-COMBUSTIVEL que determina a penalização da reinserção dos nós em relação às restrições de combustível. Ele recebe as rotas r_i, r_j, o par de nós *pickup-delivery* (a, a') em r_i, os nós b, c em r_j e a restrição de interesse γ. Entre as linhas 1 e 5, o método avalia o impacto da remoção de (a, a') no combustível usado na rota r_i até o atendimento de a. Inicialmente, registra-se o combustível máximo f^{max}_a entre o início da rota e o atendimento de a. Em seguida, a variação v_i de combustível usado na rota r_i quando (a, a') é removido. Note que isso pode ser feito sem a necessidade de refazer a rota r_i. Após isso, registra-se o máximo de combustível permitido m_i para a rota r_i. Em posse desses valores, utiliza-se o método PEN-PROPRIEDADE para calcular penalização da rota r_i com a remoção de (a, a'). O mesmo raciocínio é aplicado para a inserção de (a, a') em r_j. O final do método é a soma dos valores retornados por PEN-PROPRIEDADE.

Algoritmo 44 apresenta o método PEN-JANELA que determina a penalização da reinserção dos nós em relação às restrições de janela. Ele recebe as rotas r_i, r_j, o par de nós *pickup-delivery* (a, a') em r_i, os nós b, c em r_j e a restrição de interesse γ. Esse método utiliza os vetores auxiliares de folga Q^L_r e Q^U_r de uma rota $r \in S$. O vetor Q^L_r mantém na posição k a menor folga para a janela inferior entre k e o final de rota, em que a folga é definida como $T_k - L_k$ com T_k como horário de atendimento de k e L_k o limite inferior da janela de tempo de k. De forma semelhante, o vetor Q^U_r mantém a menor folga para a janela superior entre k e o final da rota, em que a folga é definida como $U_k - T_k$, no qual U_k representa a janela superior de

Algoritmo 43: PEN-COMBUSTIVEL$(r_i, r_j, a, a', b, c, \gamma)$

entrada: Duas rotas r_i, r_j, uma par de nós (a, a') *pickup-delivery* em r_i, dois nós b, c em r_j e uma restrição γ de interesse
saída : Um número real

1 Seja k_a o índice de a para o vetor $Q^f_{r_i}$;
2 $f^{max}_a \leftarrow Q^f_{r_i}[k_a]$;
3 Seja v_i a variação de combustível ao remover (a, a') de r_i;
4 Seja m_i a capacidade máxima de combustível para o helicóptero que atende r_i;
5 $c_i \leftarrow$ PEN-PROPRIEDADE$(f^{max}_a, m_i, v_i, \gamma_F = \gamma)$;
6 Seja k_c o índice de c para o vetor $Q^f_{r_j}$;
7 $f^{max}_c \leftarrow Q^f_{r_j}[k_c]$;
8 Seja v_j a variação de combustível ao inserir a' após c e a após b em r_j;
9 Seja m_j a capacidade máxima de combustível para o helicóptero que atende r_j;
10 $c_j \leftarrow$ PEN-PROPRIEDADE$(f^{max}_c, m_j, v_j, \gamma_F = \gamma)$;
11 **retorne** $c_i + c_j$;

k. Caso $Q^L_r[k] > 0$ para algum nó k então do nó k até o final da rota é possível adiantar o atendimento de todos o nós de um tempo equivalente a $Q^L_r[k]$ sem que nenhuma inviabilidade seja incluída nesse trecho. O mesmo raciocínio é válido para a janela superior Q^U_r, em que é permitido postegar o atendimento de cado nó de um tempo equivalente a $Q^U_r[k]$. Dado que PEN-PROPRIEDADE (Algoritmo 40) necessita de um valor máximo para os cálculos das inviabilidades, a estratégia adotada para PEN-JANELA é utilizar o valor negativo desses vetores e garantir que as modificações na rota produzam um valor inferior a zero. De forma semelhante as outros métodos, a avaliação da remoção e inserção de (a, a') é feita em duas partes. Para a remoção de (a, a') deve ser avaliada a alteração para as janelas de tempo que ocorrem desde o nó k_{a-1} anterior a a até o final da rota r_i. Seja a alteração de tempo t_a devido à remoção de a de r_i. Se $-(Q^L_{r_i}[k_{a-1}] + t_a)$ é negativo, então é garantido que a remoção de a não implica em nenhuma inviabilidade até o final da rota quando se considera somente a janela de tempo inferior. Note que soma-se $+t_a$, pois acréscimos de tempo aumentam a folga na parte inferior da janela de atendimento. De maneira semelhante, se $-(Q^U_{r_i}[k_{a-1}] - t_a)$ é negativo, nenhum nó é inviável entre k_{a-1} e o final da rota após a remoção de a quando se considera a janela superior. Nesse caso, é necessário usar o valor negativo $-t_a$, pois diminuir o tempo de atendimento dos nós k_{a-1} até o final da rota aumenta a folga da sua janela de atendimento superior. Portanto, $-Q^L_{r_i}[k_{a-1}]$ é o valor inicial e $-t_a$ a variação para a janela inferior e $-Q^U_{r_i}[k_{a-1}]$ é o valor inicial e $+t_a$ a variação para a janela superior para o nó de *pickup*. No caso de nó *delivery*, usa-se a alteração de tempo t_i ocorrida com a remoção de a, a' de r_i. Dada essas propriedades e explicações iniciais, a remoção é realizada da seguinte forma. Nas linhas 1-4, registra-se os valores das folgas inferiores e superiores do nó anterior k_{a-1} a a e do nó de *delivery* a'. Além disso, calcula-se a alteração de tempo na remoção de a e de a, a' de r_i. Nas linhas 5 e 6 a penalização é calculada para as janelas de tempo inferior e superior. A mesma lógica é aplicada

considerando os nós b e c para a inserção de (a, a') em r_j. O valor retornado por PEN-JANELA é a soma de todas as penalizações geradas pela função PEN-PROPRIEDADE.

Algoritmo 44: PEN-JANELA$(r_i, r_j, a, a', b, c, \gamma)$

entrada: Duas rotas r_i, r_j, uma par de nós (a, a') *pickup-delivery* em r_i, dois nós b, c em r_j e uma restrição γ de interesse
saída : Um número real

Seja k_{a-1} o índice do nó anterior a para os vetores de folga $Q^L_{r_i}$ e $Q^U_{r_i}$;
Seja k'_a o índice de a' para os vetores de folga $Q^L_{r_i}$ e $Q^U_{r_i}$;
Seja t_a a alteração de tempo em r_i ao remover a;
Seja t_i a alteração de tempo em r_i ao remover a e a';
$c_i \leftarrow 0 +$ PEN-PROPRIEDADE$(-Q^L_{r_i}[k_{a-1}], 0, -t_a, \gamma_J = \gamma)$;
$c_i \leftarrow c_i +$ PEN-PROPRIEDADE$(-Q^U_{r_i}[k_{a-1}], 0, +t_a, \gamma_J = \gamma)$;
$c_i \leftarrow c_i +$ PEN-PROPRIEDADE$(-Q^L_{r_i}[k'_a], 0, -t_i, \gamma_J = \gamma)$;
$c_i \leftarrow c_i +$ PEN-PROPRIEDADE$(-Q^U_{r_i}[k'_a], 0, +t_i, \gamma_J = \gamma)$;
Seja k_b o índice de b para os vetores de folga $Q^L_{r_j}$ e $Q^U_{r_j}$;
Seja k_c o índice de c para os vetores de folga $Q^L_{r_j}$ e $Q^U_{r_j}$;
Seja t_b a alteração de tempo em r_j ao adicionar a após b;
Seja t_j a alteração de tempo em r_j ao adicionar a após b e a' após c;
$c_j \leftarrow 0 +$ PEN-PROPRIEDADE$(-Q^L_{r_j}[k_b], 0, -t_b, \gamma_J = \gamma)$;
$c_j \leftarrow c_j +$ PEN-PROPRIEDADE$(-Q^U_{r_j}[k_b], 0, +t_b, \gamma_J = \gamma)$;
$c_j \leftarrow c_j +$ PEN-PROPRIEDADE$(-Q^L_{r_j}[k_c], 0, -t_j, \gamma_J = \gamma)$;
$c_j \leftarrow c_j +$ PEN-PROPRIEDADE$(-Q^U_{r_j}[k_c], 0, +t_j, \gamma_J = \gamma)$;
retorne $c_i + c_j$;

Algoritmo 45 apresenta o método PEN-PESO que determina a penalização da reinserção dos nós em relação às restrições de peso. Ele recebe as rotas r_i, r_j, o par de nós *pickup-delivery* (a, a') em r_i, os nós b, c em r_j e a restrição de interesse γ. Entre as linhas 1 e 12, o método avalia o impacto da remoção de (a, a') no peso total entre a e a' em r_i. Na remoção de nós, há duas possibilidades para calcular a remoção. Na primeira, o peso aumenta entre a e a'. Isso pode ser verificado usando o vetor auxiliar peso máximo $Q^w_{r_i}$. Entretanto, isso só é verdade se esse máximo entre a e a' também for o máximo entre o início da rota e a'. O objetivo dessa estratégia é permitir a remoção de nós (a, a') que não sejam inviáveis para à restrição de peso, mas que tenham alguma inviabilidade para ela entre a e a'. No segundo caso, utiliza-se o peso total ao atender o nó a como referência. Assim, inicialmente, determina-se o nó k_{a-1} anterior a a em r_i. Verifica-se se entre k_{a-1} e a' ocorreu aumento do peso total. Se sim, então o valor $Q^w_{r_i}[k_{a'}]$ é registrado na variável p_i. Caso contrário, o valor do peso de atendimento em a é registrado em p_i. Após isso, registra-se o peso máximo m_i permitido para a rota r_i. A variação do peso é o valor negativo do peso v necessário para atender a, pois esse é o decréscimo de peso ao remover a de r_i. Note que aqui não é computado o peso do combustível liberado. Em posse desses valores, utiliza-se o método PEN-PROPRIEDADE para calcular a penalização da rota r_i com a remoção de (a, a'). No caso da inserção, entre as linhas 13-18, registra-se o combustível

consumido f_j para atender os par de nós (a, a') inseridos em r_j e o consumo de combustível $f_{a'}$ decorrente da inserção do nó a' em r_j. Além disso, registra-se o peso máximo entre b e c na rota r_j. Esse valor é mantido pelo método OR-OPT-HRP (Algoritmo 30) ao avaliar as possibilidades de inserção em r_j. Assim, não é necessário recalculá-lo no método PEN-PESO. Além disso, registra-se o peso máximo m_j permitido para a rota r_j. A partir desses valores, aplica-se PEN-PROPRIEDADE (linha 19) para avaliar acréscimo de combustível f_j sobre o peso entre o início da rota e o nó b. Em seguida, linha 18, PEN-PROPRIEDADE é avaliada novamente com o peso máximo entre b e c em que o acréscimo de peso decorrente da inserção (a, a') é o peso para atender a e o combustível para atender a'. O valor final do método é a soma dos valores retornados por PEN-PROPRIEDADE para r_i e r_j.

Algoritmo 45: PEN-PESO$(r_i, r_j, a, a', b, c, \gamma)$

entrada: Duas rotas r_i, r_j, uma par de nós (a, a') *pickup-delivery* em r_i, dois nós b, c em r_j e uma restrição γ de interesse
saída : Um número real

1 Seja k_{a-1} o índice do nó anterior a a para o vetor de peso máximo $Q^w_{r_i}$;
2 Seja $k_{a'}$ o índice de a' para o vetor de peso máximo $Q^w_{r_i}$;
3 Seja p_a o peso total da aeronave ao atender o nó a em r_i;
4 Seja v o acréscimo de peso ao atender o nó a;
5 **se** $Q^w_{r_i}[k_{a-1}] < Q^w_{r_i}[k_{a'}]$ **então**
6 $\quad p_i \leftarrow Q^w_{r_i}[k_{a'}]$;
7 **fim**
8 **senão**
9 $\quad p_i \leftarrow p_a$;
10 **fim**
11 Seja m_i a capacidade máxima de peso do helicóptero que atende r_i;
12 $c_i \leftarrow$ PEN-PROPRIEDADE$(p_i, m_i, -v, \gamma_W = \gamma)$;
13 Seja k_b o índice do nó b para o vetor de peso máximo $Q^w_{r_j}$ em r_j;
14 Seja f_j a variação de combustível (em kg) ao adicionar (a, a') em r_j;
15 Seja $f_{a'}$ a variação de combustível (em kg) ao adicionar a' em r_j;
16 Seja p_j o peso máximo entre b e c em r_j;
17 Seja m_j a capacidade máxima de peso do helicóptero que atende r_j;
18 $c_j \leftarrow 0 +$ PEN-PROPRIEDADE$(Q^w_{r_j}[k_b], m_j, f_j, \gamma_W = \gamma)$;
19 $c_j \leftarrow c_j +$ PEN-PROPRIEDADE$(p_j, m_j, v + f_{a'}, \gamma_W = \gamma)$;
20 **retorne** $c_i + c_j$;

O Algoritmo 46 apresenta o método DISTANCIA-AERONAVES que calcula a variação da distância e a variação do custo de uso de aeronaves para uma reinserção de nós *pickup-delivery*. Ele recebe as rotas r_i, r_j, o par de nós *pickup-delivery* (a, a') em r_i, os nós b, c em r_j. Inicialmente, define-se o custo total retornado pelo método como $v = 0$. Nas linhas 1 e 2, calcula-se a variação de distância ao remover (a, a') de r_i e inseri-los em r_j. Note que isso pode ser calculado sem a necessidade de refazer as rotas r_i e r_j. Em seguida, na linha 4, verifica-se se a remoção de (a, a') torna r_i uma rota sem atendimento. Nesse caso, o custo de uso da

aeronave é subtraído de v. Na linha 7, de forma semelhante, avalia-se se a rota r_j era uma rota sem atendimento antes da inserção de (a, a'). Nesse caso, soma-se a v o custo de utilização da aeronave para a rota r_j. O custo da reinserção v é retornado como resultado final do método.

Algoritmo 46: DISTANCIA-AERONAVES(r_i, r_j, a, a', b, c)

entrada: Duas rotas r_i, r_j, uma par de nós (a, a') *pickup-delivery* em r_i, dois nós b, c em r_j
saída : Um número real

1 Seja d_i a variação da distância em r_i ao remover (a, a') de r_i;
2 Seja d_j a variação da distância em r_j a inserir a' após c e a após b;
3 $v \leftarrow d_i + d_j$;
4 **se** *A remoção* (a, a') *de* r_i *a torna sem atendimento* **então**
5 $\quad v \leftarrow v -$ custo de uso do helicóptero atendendo a rota r_i;
6 **fim**
7 **se** *A rota* r_j *não realiza atendimento antes da inserção de* (a, a') **então**
8 $\quad v \leftarrow v +$ custo de uso do helicóptero atendendo a rota r_j;
9 **fim**
10 **retorne** v;