UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

FLAVIO ROBERTO BAYER

PROPOSTA DE ALGORITMO DE BUSCA LARGA EM VIZINHANÇA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM SUPORTE A MÚLTIPLAS JANELAS DE TEMPO

JOINVILLE
2021

FLAVIO ROBERTO BAYER

PROPOSTA DE ALGORITMO DE BUSCA LARGA EM VIZINHANÇA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM SUPORTE A MÚLTIPLAS JANELAS DE TEMPO

Dissertação apresentada ao Programa de Pós–Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Fabiano Baldo

JOINVILLE
2021

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Setorial do CCT/UDESC,

com os dados fornecidos pelo(a) autor(a)

Bayer, Flavio Roberto
Proposta de algoritmo de busca larga em vizinhança para o problema de roteamento de veículos com suporte a múltiplas janelas de tempo / Flavio Roberto Bayer. -- 2021.
115 p.

Orientador: Fabiano Baldo
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2021.

1. Problema do roteamento de veículos. 2. Problema de agendamento. 3. Múltiplas janelas de tempo. 4. Meta-heurística. 5. Busca adaptativa em vizinhança de larga escala. I. Baldo, Fabiano. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. III. Título.

FLAVIO ROBERTO BAYER

PROPOSTA DE ALGORITMO DE BUSCA LARGA EM VIZINHANÇA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM SUPORTE A MÚLTIPLAS JANELAS DE TEMPO

Dissertação apresentada ao Programa de Pós–Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Fabiano Baldo

BANCA EXAMINADORA:

Dr. Fabiano Baldo
CCT/UDESC

Membros:

Dr. Marcone Jamilson Freitas Souza
DECOM/UFOP

Dr. Rafael Stubs Parpinelli
CCT/UDESC

Joinville, 26 de julho de 2021

Com muita satisfação, dedico este trabalho de pesquisa aos meus pais, família e amigos. Pelo apoio e suporte que me deram durante todo o curso e pelas incontáveis horas de ajuda dedicadas neste monografia.

AGRADECIMENTOS

Agradeço aos meus pais, Ana e Haroldo, a minha irmã, Elaine, minha namorada Mariana, e a toda minha família por todo o apoio fornecido em todos os momentos e em minhas decisões.

Ao meu orientador Fabiano Baldo, por seu suporte, incentivo, paciência e orientações no decorrer do desenvolvimento deste trabalho.

Aos colegas de universidade, pela parceria, momentos de diversão, alegria, suporte e por estarem sempre dispostos a ajudar e cooperar.

Agradeço a todos os professores da UDESC, pelas emoções, jornadas e pelo conhecimento fornecido no decorrer destes anos de estudo. Agradeço aos meus amigos, aos que se encontram próximos e aos que se encontram distantes, que me deram suporte e incentivo, pelos momentos bons que tivemos de aventura e felicidade.

Por fim, agradeço a todos que dedicam seu valioso tempo em ler e apreciar este trabalho.

"Não tenho certeza de nada, mas a visão das estrelas me faz sonhar." (Vincent van Gogh)

RESUMO

Em suas operações diárias, empresas de logística frequentemente precisam realizar o planejamento das rotas de seus veículos para atender seus clientes em horários programados. Muitas vezes esses atendimentos devem ser encaixados em uma rota de vários dias, ou então tolerar múltiplos intervalos para atendimento de um cliente no mesmo dia. As abordagens mais comuns para solução do Problemas de Roteamento de Veículos (VRP) geralmente não são capazes de lidar com esse tipo de situação, pois o tratamento de múltiplas janelas de atendimento por si só já é considerado um problema de otimização. Para suprir esta necessidade, o Problema de Roteamento de Veículos com Múltiplas Janelas de Tempo, também conhecido como VRPMTW, é uma especialização do Problema de Roteamento de Veículos em que cada serviço possui uma ou mais janelas de tempo em que podem ser atendidos. Embora o VRP seja um problema bem explorado, poucos estudos abordam as características do VRPMTW. Este trabalho apresenta a proposta de um método de otimização para o VRPMTW baseado na meta-heurística ALNS, associado a uma técnica de programação dinâmica para cálculo incremental (Δ-*Evaluation*) dos tempos de atendimento dos serviços e minimização do tempo de viagem. O objetivo do método proposto é realizar a minimização do tamanho da frota necessária para atender a todos os serviços, ao mesmo tempo que minimiza a jornada dos veículos, englobando tanto o tempo que o veículo está em deslocamento quanto o tempo que o veículo se encontra parado. Com base nos testes realizados foi possível constatar, com significância estatística, uma redução média de 0,93% no custo das soluções geradas em comparações aos resultados relatados por outros trabalhos na literatura. Ainda, o método proposto obteve melhor resultado em 31 das 48 instâncias de *benchmark* avaliadas. Além disso, por meio da avaliação incremental implementada foi possível constatar uma redução de 26,05% no tempo computacional necessário para execução do algoritmo, quando comparado a técnicas habituais para avaliação das múltiplas janelas de tempo. Portanto, é possível dizer que o método proposto apresenta resultados promissores.

Palavras-chave: Problema do roteamento de veículos, Problema de agendamento, Otimização, Múltiplas janelas de tempo, Meta-heurística, Busca adaptativa em vizinhança de larga escala.

ABSTRACT

Logistics companies often need to plan their vehicle routes to serve their customers into scheduled times in their daily operations. These plannings must often span across of several days, or allow for multiple service intervals for the same customer. The most common approach for solving the Vehicle Routing Problem (VRP) are usually unable to treat these types of constraints, since handling multiple time windows is already considered an optimization problem by itself. In order to meet these needs, the Vehicle Routing Problem with Multiple Time Windows, also known as VRPMTW, is a specialization of the Vehicle Routing Problem in which each customer may have one or more time windows available. Although the VRP is a well explored problem, few studies address the characteristics of VRPMTW. This work presents the proposal of an optimization method for the VRPMTW based on the ALNS meta-heuristic, associated with a dynamic programming technique for incremental computation (Δ-*Evaluation*) of service times and minimization of travel time. The objective of the proposed method is to minimize the size of the fleet required to attend all services, while minimizing the work time of vehicles, including both moving and idle vehicle time. Based on the tests carried out it was possible to verify, with statistical significance, an average of 0.93% cost reduction of the solutions generated in comparison with the results reported by other works in the literature. Also, the proposed method obtained the best result for 31 out of the 48 benchmark instances evaluated. Furthermore, through the implemented incremental computation, it was possible to verify a reduction of 26.05% in the computational time needed to run the algorithm, when compared to the usual techniques for evaluating multiple time windows. Therefore, it is possible to say that the proposed method presents promising results.

Keywords: Vehicle routing problem, Scheduling problem. Optimization. Multiple time windows. Meta-heuristic, Adaptive large neighborhood search.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

VRP	Vehicle Routing Problem
TSP	Traveling Salesman Problem
VRPTW	Vehicle Routing Problem with Time Window
VRPPD	Vehicle Routing Problem with Pickup and Delivery
CVRP	Capacited Vehicle Routing Problem
HFVRP	Heterogeneous Fleet Vehicle Routing Problem
OVRP	Open Vehicle Routing Problem
DVRP	Dynamic Vehicle Routing Problem
VRPMTW	Vehicle Routing Problem with Multiple Time Windows
TS	Tabu Search
MTDP	Minimum Tour Duration Problem
VRPSTW	Vehicle Routing Proble with Soft Time Windows
SVRP	Stochastic Vehicle Routing Proble
MDVRP	Multi-Depot Vehicle Routing Proble
MCTP	Minimum Completion Time Problem
GA	Genetic Algorithm
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
NS	Neighborhood Search
ALNS	Adaptive Large Neighborhood Search
GRASP	Greedy Randomized Adaptive Search Procedure
SA	Simulated Anealing
LNS	Large Neighborhood Search
TOP	Team Orienteering Problem
VNS	Variable Neighborhood Search
TSPTW	Traveling Salesman Problem with Time Windows
ILS	Iterated Local Search
GRILS	Greedy Randomized Iterated Local Search
OP	Orienteering Problem
TSPMTW	Traveling Salesman Problem with Multiple Time Windows

AVNS	Adaptive Variable Neighborhood Search
JHM	Java Microbenchmark Harness
CSV	Comma Separated Calues
RPD	Relative Percent Difference

SUMÁRIO

1 INTRODUÇÃO

Em suas operações diárias, empresas de serviço logístico frequentemente se deparam com a necessidade de programar os deslocamentos de seus veículos. Essa tarefa é complexa e de suma importância para o sucesso e eficiência de uma operação logística. Entretanto, ao mesmo tempo que ela é uma tarefa corriqueira e repetitiva, ela se torna extremamente desgastante quando realizada de forma manual. A partir desta motivação, a ciência passou a dedicar esforços no estudo desse tipo de problema, dando origem ao Problema de Roteamento de Veículo (*Vehicle Routing Problem* – VRP), que visa formular o cenário de otimização da definição de rotas de veículos para atendimento de clientes distribuídos geograficamente.

De forma geral, no VRP é realizada a minimização da soma do comprimento das rotas de um conjunto finito de veículos, com origem e destino final em uma localização definida (comumente um depósito ou garagem). Esses veículos devem percorrer um conjunto de roteiros, de forma que todos os clientes devem ser visitados uma única vez.

Por sua aplicação prática, a solução deste problema é um componente chave no gerenciamento de distribuição de itens e operações logísticas (WEI et al., 2018). O estudo do VRP permite otimizar e reduzir os custos operacionais e aumentar o nível de satisfação dos clientes (HASLE; KLOSTER, 2007). Esses problema possui aplicações em diferentes áreas de logística, em geral, distribuição e coleta de cargas e pessoas, atendimento de manutenção programada, entre outros (YU; YANG, 2011). Aumentar a eficiência do planejamento das rotas de operação logística é essencial para tornar o serviço de transporte atrativo e competitivo (LIU et al., 2013).

O problema de roteamento de veículos, introduzido por Dantzig e Ramser (1959), é um problema NP-Difícil clássico de otimização combinatória (WEI et al., 2018). Ele generaliza o problema do caixeiro viajante (*Traveling Salesman Problem* – TSP), pois realiza a otimização do percurso não apenas de um caixeiro, mas sim de vários veículos simultaneamente para o mesmo conjunto de clientes a serem visitados (LARSEN; MADSEN, 2000). Portanto, diferentemente do problema do caixeiro viajante clássico, no VRP pode haver mais de um ciclo Hamiltoniano a ser percorrido (MIRHASSANI; ABOLGHASEMI, 2011a).

Além de possuir um importante papel para a área de pesquisa operacional, no mundo real é possível correlacionar o VRP com diversas aplicações práticas de operações logísticas (YU; YANG, 2011). Empresas de serviço de logística frequentemente possuem características peculiares às suas operações, necessários para concretizar suas operações diárias, englobando aspectos do meio físico, regulamentação e especificidades da empresa, fazendo com que seja necessário que métodos avaliem diferentes restrições e objetivos (CORDEAU et al., 2007). Nos últimos anos, visando aproximar os estudos em VRP às necessidades e restrições encontradas em cenários reais, extensivos estudos vêm sendo

realizados para incorporar novas restrições à definição clássica de VRP, expandindo sua abrangência de aplicação.

Dentre as principais variantes do VRP clássico existe o problema de minimização da quantidade de veículos utilizados. Essa variante faz com que seja necessário minimizar também o número de veículos, mesmo que acarrete em aumento do comprimento das rotas. Ainda, outras variantes comumente encontradas lidam com a otimização de métricas de satisfação de cliente que englobam aspectos como: redução do tempo de espera, redução de atrasos, aumento da confiabilidade, redução do tempo de viagem, entre outros (CORDEAU et al., 2007).

Além das variantes associadas às alterações na minimização do objetivo do VRP clássico, ainda é possível encontrar variantes que lidam com diversas restrições impostas pelo cenário de aplicação do VRP. Por exemplo, em determinados cenários, os consumidores esperam ser atendidos em um intervalo de tempo específico, caracterizando assim o VRP como com Janela de Tempo (*Vehicle Routing Problem with Time Window* – VRPTW) (LU; YANG, 2012). Em outros casos, os veículos precisam realizar as operações de carga e descarga no decorrer do trajeto de acordo com a necessidade individual de cada cliente, caracterizando o VRP como de Coleta e Entrega (*Vehicle Routing Problem with Pickup and Delivery* – VRPPD) (NACCACHE; CôTé; COELHO, 2018). Além disso, um cliente pode exigir um veículo com capacidades específicas para que suas necessidades sejam atendidas, caracterizando o VRP como com Capacidade (*Capacited Vehicle Routing Problem* – CVRP) (YU; YANG, 2011), ou ainda, veículos podem ter características distintas dentro de uma frota, caracterizando o VRP como com Frota Heterogênea (*Heterogeneous Fleet Vehicle Routing Problem* – HFVRP) (TASAN; GEN, 2012). Eventualmente, pode não haver necessidade do veículo iniciar ou finalizar a sua rota em um depósito, permitindo que a rota comece e/ou termine em lugares arbitrários, caracterizando o VRP como com Rotas Abertas (*Open Vehicle Routing Problem* – OVRP) (Sakakibara; Tamaki; Nishikawa, 2007). Também pode ocorrer a necessidade de otimização durante o dia de operação, onde novas requisições são recebidas durante a execução do roteamento, caracterizando assim o problema como VRP Dinâmico (*Dynamic Vehicle Routing Problem* – DVRP) (SCHMITT; BALDO; PARPINELLI, 2018). Dependendo das necessidades da operação logística, diferentes combinações dessas características de VRP podem ser necessárias.

Considerando aplicações reais do VRPTW, nem sempre um cliente está disponível para ser atendido em uma única janela de tempo, isso significa que ele pode informar ao entregador um conjunto de intervalos de tempo (janelas) em que ele está disponível para atendimento e, neste caso, o roteirizador deve identificar qual é a melhor janela para atendê-lo. Esse cenário dá origem ao problema chamado VRP com múltiplas janelas de tempo (*Vehicle Routing Problem with Multiple Time Windows* – VRPMTW). Segundo Bell et al. (1983), um exemplo de VRPMTW pode ser encontrado na distribuição recorrente de gases industriais. Nesse cenário, clientes estão disponíveis para atendimentos em diferentes

períodos do mesmo dia e também em diferentes dias da semana. Portanto, existem vários intervalos de tempo possíveis para realizar o atendimento ao cliente. Belhaiza, Hansen e Laporte (2014) retratam o cenário de entrega de móveis e eletrodomésticos de lojas varejistas, em que os clientes possuem opção de escolha de entrega entre vários períodos do dia em diferentes dias. Já Rancourt, Cordeau e Laporte (2013) abordam o caso de transporte de longa distância com vários dias de operação, também lidando com a questão da jornada de trabalho dos motoristas. Nesse cenário, é necessário decidir tanto qual é o melhor horário quanto o melhor dia para fazer uma entrega em um cliente específico. Li et al. (2020) justificam o uso de múltiplas janelas de tempo em casos como intervalo de trabalho, de forma que existam dois períodos disjuntos no mesmo dia que o cliente pode ser atendido (de manhã ou a tarde, por exemplo, mas não durante o horário de almoço). Lin e Yu (2015) e Souffriau, Vansteenwegen e Berghe (2013) trazem o caso do problema do turista que quer visitar o maior número possível de atrações em uma cidade, mas elas não estão abertas a todo momento. Por fim, Larsen e Pacino (2019) abordam o caso de distribuição de compras em que os clientes têm a possibilidade de fazer agendamento em sistema *on-line* para selecionar os horários disponíveis.

Portanto, no VRPMTW cada cliente possui uma ou mais possíveis janelas de atendimento, podendo estarem contidas no mesmo dia ou distribuídas entre múltiplos dias, sendo necessário que para cada cliente seu atendimento inicie dentro do período de qualquer uma de suas janelas de tempo (BELHAIZA et al., 2018). Para resolver esse tipo de problema, Bell et al. (1983) incorporaram ao VRP o problema de agendamento com múltiplas janelas de tempo.

Embora a área de estudo de roteamento de veículos seja bastante ativa, poucos estudos foram realizados abordando a característica de múltiplas janelas de tempo. Contudo, é possível encontrar estudos que abordam essa característica em problemas de otimização semelhantes, tais como, no TSP, com os trabalhos de Paulsen, Diedrich e Jansen (2015) e Pesant et al. (1999), no *Orientering Problem*, com os trabalhos de Lin e Yu (2015) e Souffriau, Vansteenwegen e Berghe (2013), e no *Team Orientering Problem*, com os trabalhos de Tricoire et al. (2013) e Tricoire et al. (2010). Portanto, é possível perceber que existem diversos problemas de otimização que lidam com múltiplas janelas de tempo, o que reforça a justificativa de sua aplicação ao contexto do VRP.

Obter uma solução eficiente para o VRP clássico não é uma tarefa simples, haja visto que ele é um problema NP-Difícil (CORMEN, 2009). Isso significa que sua solução gera uma explosão combinatória de possibilidades que aumenta de forma exponencial de acordo com o aumento do tamanho do problema. Nesse contexto, incluir o problema de agendamento de atendimento, resultante da incorporação das múltiplas janelas de tempo, que por si só também é um problema de otimização combinatória, torna a busca pela solução ótima de um VRPMTW ainda mais custosa.

Existem diversas abordagens possíveis para obter uma solução para o VRP, tanto

aplicando métodos exatos, quanto utilizando heurísticas e meta-heurísticas (OYOLA; ARNTZEN; WOODRUFF, 2017; OYOLA; ARNTZEN; WOODRUFF, 2018). De acordo com o levantamento de Braekers, Ramaekers e Van Nieuwenhuyse (2016) acerca dos trabalhos publicados envolvendo VRP, cerca 70% abordam o problema por meio de meta-heurísticas, enquanto 17% utilizam métodos exatos. Embora métodos exatos podem garantir uma solução ótima, pelo fato de o VRP ser um problema NP-Difícil, o tempo e recursos computacionais necessários para resolver uma instância não trivial do problema com esse tipo de método se torna inviável (MIRHASSANI; ABOLGHASEMI, 2011a). Em virtude disso, grande parte das pesquisas feitas para a solução deste problema utilizam abordagens meta-heurística para produzir uma solução próxima da ótima em um tempo aceitável e utilizando recursos computacionais compatíveis (Sakakibara; Tamaki; Nishikawa, 2007).

Dentre os poucos estudos que abordam o VRPMTW encontrados na literatura, as soluções por eles propostas abrangem a utilização de heurísticas exatas (LI et al., 2020), *Tabu Search* (BELHAIZA; HANSEN; LAPORTE, 2014; Belhaiza, 2018; Belhaiza; M'Hallah, 2016), algoritmos genéticos (BEHESHTI; HEJAZI; ALINAGHIAN, 2015), algoritmos baseados em colônia de formiga (Bitao; Fei, 2010), algoritmos de busca variável em vizinhança (Belhaiza, 2018; Belhaiza; M'Hallah, 2016; BELHAIZA et al., 2018) e algoritmos de Busca Larga Adaptativa em Vizinhança (HOOGEBOOM et al., 2020). A maioria dos estudos mais recentes apontam a utilização de meta-heurísticas de melhoramento de solução única como alternativa promissora para a solução do VRPMTW. Ainda, para diminuir a complexidade da solução do subproblema de otimização de agendamento das múltiplas janelas de tempo, a grande maioria delas empregam procedimentos exatos otimizados para restringir a quantidade de cálculos necessários para a análise de factibilidade dos possíveis agendamentos.

Dadas as contribuições acerca da forma de modelagem do problema apresentadas pelos trabalhos relacionados, este trabalho também assume que a melhor estratégia de abordagem para a solução do VRPMTW é por meio da utilização de uma heurística adaptativa de melhoramento de solução única, conjugada com a aplicação de um método exato otimizado pelo uso de programação dinâmica para a solução do subproblema de agendamento das múltiplas janelas. Portanto, a pergunta de pesquisa que se pretende responder é: como resolver de forma eficiente e com uma boa aproximação à solução ótima o VRPMTW utilizando meta-heurística de melhoramento de solução única combinada com a utilização de programação dinâmica para otimizar a avaliação da factibilidade e cálculo dos tempos de atendimento das múltiplas janelas de tempo?

1.1 OBJETIVO GERAL

Este trabalho tem por objetivo desenvolver um método de otimização baseado em uma meta-heurística de melhoramento de solução única, combinada com a utilização de um método exato otimizado pelo aplicação de programação dinâmica, para a solução eficiente do VRPMTW, visando a diminuição da quantidade de veículos utilizados, o tempo de folga entre os agendamentos das entregas e a distância total percorrida.

1.1.1 Objetivos Específicos

Com base nos objetivo geral, este trabalho almeja alcançar os seguintes objetivos específicos:

- Identificar uma meta-heurística de melhoramento de solução única capaz de solucionar eficientemente o VRP com múltiplas janelas de tempo;
- Modelar e implementar uma meta-heurística de melhoramento de solução única competitiva para a otimização do VRP com múltiplas janelas de tempo;
- Definir e implementar o conjunto adequado de operadores a serem utilizados na execução da meta-heurística;
- Projetar e implementar uma solução exata eficiente para a otimização do cálculo de factibilidade das múltiplas janelas de tempo utilizando programação dinâmica;

1.2 METODOLOGIA DE PESQUISA

Esta seção apresenta a caracterização metodológica da pesquisa, assim como o detalhamento do procedimento metodológico utilizado no desenvolvimento deste trabalho.

1.2.1 Caracterização Metodológica

De acordo com a classificação metodológica encontrada na literatura, o tipo de pesquisa relacionada a este trabalho é de algo presumivelmente melhor (MARCONI; LAKATOS, 2003), pois ele visa propor uma forma mais eficiente para a resolução de um problema abordado pela literatura. Do ponto de vista do paradigma da pesquisa científica, este trabalho é classificado como tecnocatra (EDEN, 2007), pois utiliza a avaliação de experimentos empíricos para compreender as contribuições da solução, ou seja, as informações são analisadas *a posteriori*. Quanto ao procedimento técnico adotado, este trabalho se enquadra como uma Pesquisa-Ação, ou seja, a investigação e a prática são aplicadas de forma iterativa para que gerem melhorias incrementais nos próximos ciclos de iteração (TRIPP, 2005). Em relação à forma de avaliação da pesquisa, neste trabalho é utilizado o método indutivo, ou seja, o pesquisador busca por padrões em

casos particulares que expliquem ou se apliquem a todos os casos análogos ao analisado (MORIN; MOIGNE, 2000). Por fim, a pesquisa se enquadra no nível de maturidade 2 (WAZLAWICK, 2017), pois propõe um método diferente e busca conhecimento empírico *a posteriori* para validação da contribuição proposta.

1.2.2 Procedimento Metodológico

O procedimento metodológico adotado inicia como uma revisão bibliográfica sobre os conceitos básicos do problema de roteamento de veículo. Em seguida, se concentra na análise do Problema de Minimização do Tempo de Viagem (*Minimum Tour Duration Problem* – MTDP) e no problema de agendamento com múltiplas janelas de tempo, sendo que o primeiro será estudados, pois serve de base para o entendimento da solução do segundo. Na sequência, inicia-se a revisão dos métodos de solução, nesse caso, é feita uma pesquisa sob métodos exatos, heurísticas e meta-heurísticas de otimização, com foco nas meta-heurísticas baseadas na busca em vizinhança. Nessa etapa, busca-se detalhar as principais características das formas de solução existentes e avaliar quais têm maior potencial de aplicação na solução do problema em tela. Como última etapa de revisão, serão analisados os trabalhos relacionados encontrados na literatura que melhor se enquadram na solução do VRPMTW.

Na sequência, utilizando a técnica de Pesquisa-Ação, o método de solução será desenvolvido e sucessivamente refinado a fim de alcançar o resultado desejado. A concepção do método inicia com a definição da meta-heurística de base para a otimização combinatória, a partir desse ponto são incorporadas técnicas e abordagens para o tratamento de cada uma das características do problema abordado ao longo das iterações. Como principais características a serem tratadas pelo método proposto podem-se destacar a restrição de capacidade dos veículos, a diminuição do número de veículos utilizados, a diminuição da duração das rotas e distância percorrida, e o cálculo da melhor combinação de janelas de tempo de atendimento. A cada iteração testes são realizados utilizando instâncias de *benchmarks* referenciadas pelos trabalhos relacionados para avaliação e refinamento do método proposto.

Por fim, a avaliação dos resultados é realizada com a execução do método proposto utilizando as instâncias de *benchmark* propostas por Belhaiza, Hansen e Laporte (2014), onde serão avaliadas métricas tais como: custo total da solução e número total de veículos da solução. Esses resultados são comparados com os apresentados nos trabalhos mais recentes relacionados ao mesmo tema para avaliar sua qualidade. Esses testes são avaliados por métodos estatísticos para assegurar relevância estatística. Ainda, para avaliar o impacto no tempo de execução do algoritmo em decorrência da incorporação da técnica de programação dinâmica, são comparados os tempos do cálculo completo das janelas de tempo, do cálculo incremental proposto nos trabalhos relacionados e do cálculo incremental proposto neste trabalho.

1.3 ESTRUTURA DO TRABALHO

O restante do trabalho está estruturado da seguinte forma. O capítulo 2 apresenta a revisão dos conceitos mais relevantes ao tema abordado neste trabalho. Dentre os conceitos abordados, podem-se destacar o VRP e suas principais variantes, o problema de *Minimum Tour Duration Problem* e o de VRPMTW. Também serão revisados os principais métodos de solução do VRP. Ainda, é apresentada uma revisão abrangente sobre os principais trabalhos relacionados à solução de problemas de otimização com múltiplas janelas de tempo.

O capitulo 3 apresenta o método de solução proposto para o VRPMTW. Ele inicia apresentando uma visão geral da solução proposta e posteriormente detalha o fluxo de operação das atividades que compõem o método. Na sequência, são apresentados os principais algoritmos contidos na solução e, por fim, são detalhados aspectos de sua implementação.

O capítulo 4 apresenta a análise dos resultados alcançados pelo método proposto. Essa avaliação é feita por meio da realização de experimentos com *benchmarks* encontrados na literatura em comparação com trabalhos relacionados para demonstrar as contribuições do método proposto.

Por fim, o capítulo 5 apresenta as conclusões obtidas por meio do desenvolvimento do trabalho em tela e aponta os possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO

O desenvolvimento deste trabalho foca na solução eficiente do problema de roteamento de veículos associado ao problema de agendamento, portanto, utiliza a variação do VRP que trata de Múltiplas Janelas de Tempo. Neste capítulo serão apresentados conceitos fundamentais utilizados neste trabalho. Definições como a do Problema de Roteamento de Veículos, suas variações, e problemas relacionados pertinentes, incluindo aplicações práticas, detalhes e formulações matemáticas serão relatadas. Mais especificamente, a Seção 2.1 apresenta as definições gerais do VRP. A Seção 2.2 apresenta o Problema de Minimização do Tempo de Viagem (*Minimum Tour Duration Problem* – MTDP), utilizado para otimizar o sequenciamento de atendimentos para obter o menor tempo de viagem. A Seção 2.3 apresenta a variação do VRP que trata de Janela de Tempo, limitada a uma por atendimento, e, na sequência, a Seção 2.4 conceitua o tratamento de Múltiplas Janelas de Tempo. A Seção 2.5 apresenta os métodos utilizados para a solução do VRP, bem como seus operadores. Por fim, a Seção 2.6 apresenta a revisão dos trabalhos relacionados ao problema abordado.

2.1 O PROBLEMA DE ROTEAMENTO DE VEÍCULOS (VRP)

O Problema de Roteamento de Veículos (*Vehicle Routing Problem* – VRP) tem por objetivo otimizar a alocação de um conjunto finito de requisições de clientes em uma frota de veículos, a fim de minimizar os custos associados à realização dos percursos necessários para atendê-las. Portanto, a partir da definição do local de origem e de destino dos veículos (comumente chamado de depósito), o problema consiste em organizar uma sequência de atendimentos das requisições por cada veículo, de tal forma que cada requisição seja atendida uma única vez e por um único veiculo (PUREZA; MORABITO; REIMANN, 2012). Esse é um tema que se mantêm em evidência e evolução, haja vista que existe uma gama abrangente de extensões introduzidas ao problema clássico de VRP que ainda carecem de soluções robustas e consistentes.

A primeira versão do VRP, introduzida por Dantzig e Ramser (1959), foi aplicada no contexto de planejamento de despacho de cargas em caminhões. O objetivo era encontrar as possíveis rotas que pudessem atender a todos os clientes a serem visitados, no menor custo possível, sem violar nenhuma restrição (capacidade de carga do veículo), minimizando a distância percorrida e o número de veículos utilizados.

Na academia, o VRP é considerado um problema de relevante importância no contexto da pesquisa operacional (YU; YANG, 2011). Da mesma forma, no mundo real, ele está correlacionado a diversas aplicações práticas na cadeia logística. Por exemplo, empresas de serviço de logística diariamente enfrentam o problema de roteirizar seus veículos para servir seus clientes. Portanto, um roteamento de qualidade é um componente chave de suma importância para o sucesso e eficiência no gerenciamento de distribuição de

itens e operações logísticas (WEI et al., 2018).

O problema de roteamento de veículos é um problema ND-Difícil clássico de otimização combinatória (WEI et al., 2018) que generaliza o problema do caixeiro viajante, pois otimiza as rotas de um conjunto de veículos, diferentemente do problema do caixeiro viajante clássico, onde há apenas um ciclo hamiltoniano no grafo (MIRHASSANI; ABOLGHASEMI, 2011b).

Dentre a possíveis aplicações de soluções para o VRP, podem ser destacadas as seguintes áreas de logística: coleta e entrega de cargas, transporte de pessoas, serviços de tele-entrega, atendimento de manutenção programada, roteamento de robôs, entre outros (YU; YANG, 2011). Aumentar a eficiência desses processo de operação logística é essencial para tornar o serviço atrativo e competitivo (WEI et al., 2018; LIU et al., 2013).

O VRP pode ser definido como um grafo $G(V, E)$, onde o conjunto $V = \{0, ..., n\}$ de vértices define as demandas de clientes, sendo 0 a origem/destino (chamado de depósito) e os vértices de 1 até n as demandas a serem atendidas. Os vértices se conectam por meio de um conjunto E de arestas, sendo que cada uma das arestas c_{ij} representa o custo de deslocamento do vértice i ao vértice j. Também é definido um limiar Q que representa a capacidade máxima de carga para cada caminho a ser gerado. Para cada nó em V é definida uma quantidade de demanda específica q_i necessária para ser transportada, de tal forma que para cada caminho a soma das demandas q_i não podem ultrapassar a capacidade Q definida para os veículos.

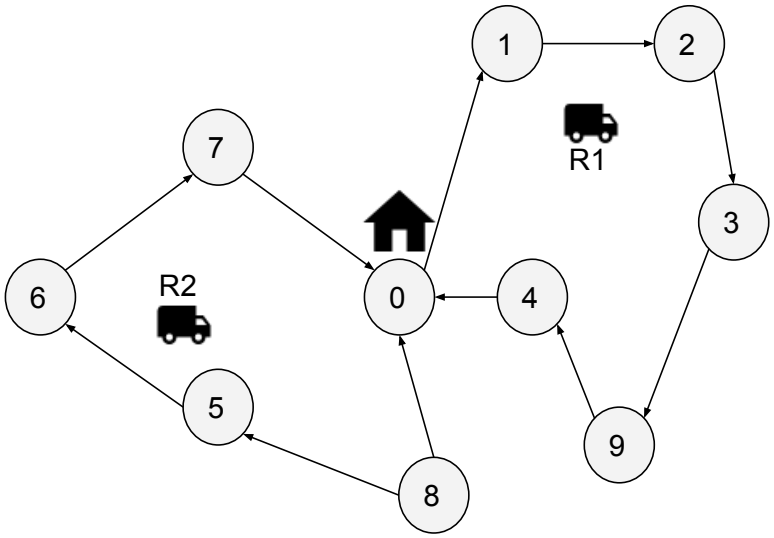
Portanto, o objetivo do VRP é determinar um conjunto M de rotas que minimizam a soma do custo total de deslocamento dado que todas as rotas devem iniciar e finalizar no depósito, cada cliente deve ser visitada exatamente uma vez e que para toda rota gerada a soma das demandas de atendimentos não pode ser superior à capacidade do veículo.

A Figura 1 a seguir ilustra o funcionamento básico do VRP. Os vértices V estão representados pelos círculos incluindo seu respectivo índice. É possível notar dois ciclos, representando duas rotas, partindo e retornando ao vértice 0, de forma que todos os vertíveis são conectados formando caminhos (uma aresta de entrada e saída). Além disso, para este exemplo foi definido que os veículos possuíam capacidade $Q = 5$ e cada atendimento possuía demanda $q_i = 1$. Dessa forma, para atender as 9 demandas são necessário ao menos duas rotas.

Dada a capilaridade da variedade de subproblemas e restrições do VRP adaptado para as necessidades específicas de cada empresa e seus clientes, diversas variações são encontradas na literatura. A seguir serão apresentadas as principais variações:

- VRPTW (*VRP with Time Window*): Cada cliente a ser visitado possui um intervalo de tempo definido para ser atendido. Portanto, para ser considerado um roteamento válido o veículo deve atender o cliente dentro desta janela de tempo (YU; YANG, 2011).

Figura 1 – Exemplo de VRP.



Fonte: Próprio autor

- VRPMTW (*VRP with Multiple Time Windows*): O VRPMTW é uma variação do VRPTW, em que um serviço pode possuir várias janelas de tempo de atendimento, sendo que o veículo deve atendê-lo dentro de uma dessas janelas (LIANG; ZHOU; ZHAO, 2009). Esta variação do VRP será abordada em mais detalhes na Seção 2.3.

- VRPSTW (*VRP with Soft Time Window*): O VRPMTW é outra variação do VRPTW, em que a janela de tempo não é uma restrição rígida, mas sim uma sugestão. Assim, é permitido que as restrições possam ser razoavelmente violada caso seja conveniente para a diminuição global do custo da solução (QURESHI; TANIGUCHI; YAMADA, 2011), seja em virtude da alteração da sequência de atendimento ou do custo de deslocamento (gerar um pequeno atraso para reduzir bastante a distância, por exemplo) (LIANG; ZHOU; ZHAO, 2009).

- CVRP (*Capacitated VRP*) ou HFVRP (*Heterogeneous fleet VRP*): Incorpora informações referentes à capacidade dos veículos utilizados no roteamento e demanda dos clientes. Esta capacidade pode ser uniforme para toda a frota (CVRP) ou específica para cada veículo (HFVRP). No segundo caso, em que a frota é heterogênea, é necessário identificar qual o melhor veículo para realizar um conjunto de atendimentos, sem manter uma considerável capacidade ociosa de ocupação (ADEWUMI; ADELEKE, 2016). Esta variação de capacidade dos veículos ocorre quando existe a limitação de quantidade de veículo ou quantidade de recursos que um veículo pode comporta, o que permite avaliar diferentes tipos de restrições e capacidades (tais como, número de passageiros, volume e peso, etc.) (FERDI; LAYEB, 2016).

- DVRP (*Dinamic VRP*) ou SVRP (*Stochastic VRP*): Incorpora a dinamicidade e estocasticidade de operações logísticas reais. Esta dinamicidade pode ser dividida em duas grandes vertentes: estocasticidade de requisições, em que clientes podem cancelar ou realizar novas requisições durante o dia (HANSHAR; OMBUKI-BERMAN, 2007; SCHMITT; BALDO; PARPINELLI, 2018), ou estocasticidade de custo, em que o custo de deslocamento entre vértices é modificado (ELHASSANIA; JAOUAD; AHMED, 2014). Como exemplo desse tipo de variante de VRP, pode-se destacar os serviços de tele-entrega, em que novas demandas surgem a todo momento e o próprio custo de deslocamento se altera dinamicamente dependendo da trafegabilidade das vias (DUMAS; DESROSIERS; SOUMIS, 1991).

- VRPPD (*VRP with Pickup and Delivery*): Bastante comum em transporte de pessoas e transporta de carga com logística reversa, nessa variação do VRP cada requisição é composta por dois serviços, um representando a coleta e outro representando a entrega (KHOUKHI et al., 2019). Um veículo deve realizar a coleta e posteriormente o mesmo veículo deve realizar a entrega sem necessariamente voltar ao depósito.

- MDVRP (*Multi-Depot VRP*): Ocorre quando a partida inicial ou destino final de um veículo pode ocorrer em diferentes locais, comum em empresas de distribuição com vários depósitos, dando mais liberdade a identificação de diferentes rotas e aumentando o espaço de busca (Nunes Bezerra et al., 2019).

Diversas outras variações do VRP podem ser encontradas. As seções 2.2, 2.3 e 2.4 detalham as principais variantes do VRP pertinentes ao trabalho em tela.

2.2 O PROBLEMA DE MINIMIZAÇÃO DO TEMPO DE VIAGEM (MTDP)

O Problema de Minimização do Tempo de Viagem (*Minimum Tour Duration Problem* – MTDP) é uma variação do problema do caixeiro viajante com janelas de tempo, também aplicável ao problema de roteamento de veículos com janelas de tempo, que visa minimizar o tempo total necessário para executar uma rota, não necessariamente a distância. Segundo Tilk e Irnich (2017), não existe uma nomeação consistente na literatura para este problema, mas neste trabalho ele será referido como MTDP.

Geralmente, o planejamento dos horários de atendimento infere que o veículo sempre realiza o atendimento das requisições o quanto antes possível, ou seja, os tempos de uma rota podem ser especificados simulando a sequência em que os pontos são visitados a partir do horário de saída do depósito (HURKAŁA, 2015). Essa abordagem pode parecer a opção mais simples sob o ponto de vista de análise de factibilidade de rota, porém ela não leva em consideração a minimização do tempo de execução da rota. Ao realizar o planejamento de uma rota, o tempo de permanência da equipe do caminhão também é

um custo existente, estando o veículo em movimento ou aguardando, tais como custo por hora do motorista ou tempo de utilização do veículo ocioso. Portanto, o MTDP tem por objetivo incorporar na solução do problema de otimização esse e outros tempos que estão direta ou indiretamente relacionados com os custos da duração da rota.

Na prática, o tempo total de viagem inclui o tempo produtivo, composto pelo tempo em que o veículo está em deslocamento ou atendimento, e o tempo improdutivo, composto pelo tempo em que o veículo se encontra aguardando para realizar o atendimento, sendo que ambos são restrições que precisam ser levadas em consideração na minimização do custo da rota. Os tempos improdutivos acontecem quando o veículo chega adiantado e precisa aguardar para realizar o atendimento de clientes em horários pré-determinados. Esse tipo de situação é comumente encontrada na solução de problemas de otimização com janelas de tempo. Tempos improdutivos também podem ocorrer por quesitos legais, como limitação de jornada de trabalho do motorista. Essa situação pode impactar na função objetivo do problema, pois os tempos improdutivos também precisam ser reduzidos, tendo em vista o custo existente com mão de obra.

Sendo assim, é necessário saber se o tempo improdutivo durante a viagem é considerado como tempo de trabalho, o que acaba incrementando o custo da rota. Caso esse tempo não acarrete em custo, este problema pode ser considerado um *Minimum Completion Time Problem* (MCTP), em que o horário de início de saída do depósito é fixo, sendo necessário otimizar apenas o horário de retorno (VIDAL et al., 2011). Caso o horário de partida do veículo não seja fixo, podendo ser atrasado, e a redução da jornada acarreta em redução do custo, este problema é definido como MTDP (PAULSEN; DIEDRICH; JANSEN, 2015). Embora o objetivo e a formulação matemática sejam parecidas entre o MCTP e o MTDP, identificar o tempo ótimo de partida é uma variável que precisa ser descoberta, adicionando mais complexidade à otimização (SAVELSBERGH, 1992).

Segundo Belhaiza, Hansen e Laporte (2014), uma vez que a ordem de atendimento dos clientes da rota foi definida pelo processo de otimização, geralmente é possível atrasar a partida da rota do depósito sem violar as janelas de tempo dos clientes, o que pode diminuir a duração total da rota. Para encontrar a rota que inicie o mais tarde e termine o mais cedo a função objetivo deve minimizar o tempo de trabalho tanto produtivo quanto improdutivo (PAULSEN; DIEDRICH; JANSEN, 2015). Savelsbergh (1992) sugere a utilização do conceito de *forward time slack* para indicar quanto tempo o início do atendimento pode ser postergado, reduzindo o tempo total da rota. Dessa forma, é possível alcançar um encolhimento nos tempos da rota para reduzir ao máximo o tempo ocioso sem afetar a factibilidade da rota ou outros componentes da função objetivo (TRICOIRE et al., 2010).

Para conseguir calcular o tempo ótimo de partida do depósito, é necessário calcular as seguintes variáveis para cada requisição (incluindo requisições e depósitos) para planejamento de atendimento:

- Chegada mais cedo (*Earliest Arrival*): Representa o menor tempo possível de chegada ao cliente, isto é, o tempo mais cedo em que um veículo consegue chegar a um cliente para realizar o atendimento do serviço na rota. Caso o local represente a saída do depósito, no início da rota, esse tempo é dado pelo início da jornada do veículo, caso contrário, é dado pela partida mais cedo do serviço anterior acrescido do tempo de deslocamento do nó anterior ao nó atual.

- Início de serviço mais cedo (*Earliest Service Start*): Representa o menor tempo possível de início do serviço, isto é, o tempo mais cedo em que um veículo efetivamente consegue começar a realizar o atendimento do serviço na rota. Caso o tempo de chegada mais cedo para o referido serviço seja posterior ao início da janela de tempo de atendimento, o atendimento pode iniciar imediatamente, fazendo com que o tempo de início de serviço mais cedo seja igual ao tempo de chegada mais cedo, caso contrário, se o veículo chegar para prestar o atendimento antes do início da janela de tempo, então ele deve aguardar até o início da janela de atendimento do cliente, caso em que o início de serviço mais cedo é igual ao início da janela de tempo do serviço. O tempo de espera é, portanto, definido como a diferença entre o tempo de início de serviço mais cedo e o tempo de chegada mais cedo.

- Partida mais cedo ou Fim de serviço mais cedo (*Earliest Departure/Earliest Service End*): Representa o menor tempo possível de fim de um serviço e consequente partida, isto é, o tempo mais cedo em que um veículo efetivamente poderá finalizar um atendimento ao cliente na rota e se encontra disponível para iniciar o deslocamento ao próximo serviço. Esse tempo é definido pela soma entre o tempo de início de serviço mais cedo acrescentado do tempo de serviço para a requisição em questão.

- Chegada mais tarde (*Latest Arrival*): Representa o maior tempo possível de chegada ao cliente, isto é, o tempo mais tarde em que um veículo pode chegar a um local para realizar o atendimento do serviço, caso contrário não será possível satisfazer as restrições de janela de tempo para pelo menos um serviço da rota, se tornando uma solução infactível. Esse tempo é definido pela subtração do tempo de serviço para a requisição em questão do tempo de início de serviço mais tarde.

- Início de serviço mais tarde (*Latest Service Start*): Representa o maior tempo possível de início do serviço, isto é, o tempo mais tarde em que um veículo efetivamente pode começar a realizar o atendimento do cliente na rota, caso contrário não será possível satisfazer as restrições de janela de tempo para pelo menos um serviço da rota, se tornando uma solução infactível. Caso o tempo de partida mais tarde menos o tempo de serviço para o referido serviço seja anterior ao final da janela de tempo de atendimento, o atendimento pode iniciar imediatamente, fazendo com que o tempo de início de serviço mais tarde seja igual ao tempo de partida mais

tarde decrescido do tempo de serviço, caso contrário, se o veículo estiver disponível para prestar o atendimento após o final da janela de tempo então ele deve adiantar o atendimento até o final da janela de tempo de atendimento do cliente, caso em que o início de serviço mais tarde é igual ao final da janela de tempo do serviço.

- Partida mais tarde ou Fim de serviço mais tarde (*Latest Departure/Latest Service End*): Representa o maior tempo possível de fim do serviço, isto é, o tempo mais tarde, em que um veículo efetivamente terá que finalizar um atendimento do serviço na rota e poderá se deslocar para o próximo serviço, caso contrário não será possível satisfazer as restrições de janela de tempo para pelo menos um serviço da rota, se tornando uma solução infactível. Caso o local represente a chegada ao depósito no final da rota, esse tempo é dado pelo final da jornada do veículo, caso contrário, é dado pela chegada mais tarde do serviço posterior decrescido do tempo de deslocamento do cliente posterior ao cliente atual.

- Tempo de folga (*Slack Time*): Representa a folga, isto é, a tolerância de tempo que um veículo pode chegar atrasado sem comprometer as janelas de tempo dos outros serviços. Esse tempo é definido pela diferença entre o tempo de partida mais tarde e o tempo de chegada mais cedo. Para o tempo mais tarde é referido como *Forward Time Slack* e para o tempo mais cedo é referido como *Backward Time Slack*.

A Figura 2 apresenta um exemplo do problema de agendamento para uma única rota e uma única janela de tempo por serviço. O eixo horizontal se refere a linha do tempo, crescente da esquerda para a direita, representando a realização do roteamento planejado. O eixo vertical se refere a sequência de atendimentos, se deslocando constantemente de cima para baixo conforme o avanço na realização dos atendimentos, com escala proporcional à quantidade de requisições (não necessariamente proporcional ao tempo de deslocamento entre requisições). Desta forma, o início da jornada do veículo é presentada no instante do canto superior esquerdo, enquanto final da jornada é representada pelo instante do canto inferior direito.

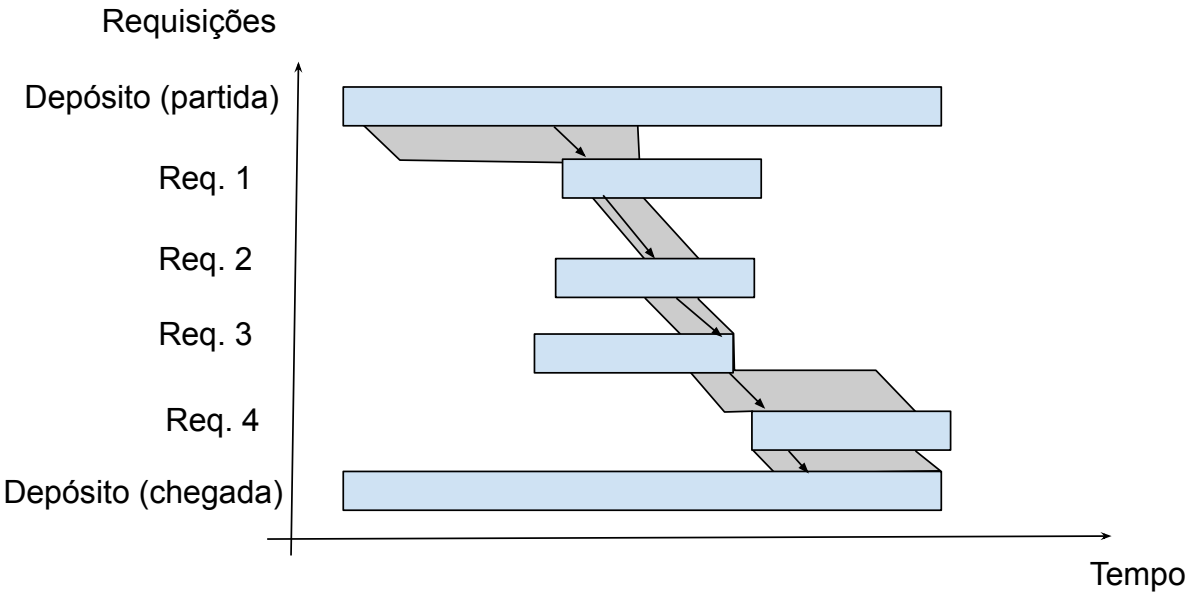
Para cada requisição no eixo vertical existe uma janela de tempo respectiva no eixo horizontal, representada pelos retângulos azuis. Cada cliente deve ter seu atendimento iniciado dentro de sua janela de tempo respectiva. Além disso, a jornada de trabalho do veículo é limitada pela janela de tempo do depósito (tanto a requisição de saída quanto a de chegada), de forma que o veículo deve iniciar e finalizar o seu trajeto dentro desse tempo. Uma representação da posição do veículo dentro da rota ao longo do tempo conforme o planejamento realizado, representado pela seta preta, .

Portanto, o diagrama da Figura 2 pode ser interpretado como uma adaptação do Diagrama de Gantt, que permite cruzar o deslocamento do veículo (progresso da execução da rota) ao longo do tempo, conforme os clientes são atendidos e o tempo progride, avançando sempre de cima para baixo e da esquerda para a direita. Contudo, a ilustração

da janela de tempo de atendimento simboliza o tempo tolerável para que o cliente seja atendido, não o momento ou duração da execução do atendimento, como em um Diagrama de Gantt habitual.

Para realizar os atendimentos no menor espaço de tempo possível sem violar suas janelas de tempo é necessário calcular o tempo de folga entre a saída mais tarde e a chegada mais para cada par de atendimentos e encontrar a sequência de atendimentos que minimize esse tempo de folga. No gráfico da Figura 2 os tempos de folga são representados pelas áreas em cinza entre cada par de atendimentos. Portanto, o objeto é fazer com que a seta, que representa o deslocamento do veículo, não esteja fora da área cinza.

Figura 2 – Exemplo de otimização de linha do tempo para o MTDP



Fonte: Próprio autor

Dada uma sequência definida de atendimentos, algoritmos eficientes podem ser utilizados para o cálculo destes valores e avaliação da solução. A avaliação de uma solução pelo MTDP é realizado em três etapas detalhadas a seguir, composta pelo cálculos dos tempos mais cedo e cálculos do tempos mais tarde (permitem determinar a factibilidade de uma rota), seguido pelo processo de compressão da região de factibilidade (permite determinar os tempos ótimos de atendimento de cada cliente para minimizar a duração de uma rota).

Os tempos mais cedo são calculados cumulativamente partindo do serviço inicial, seguido iterando pelos serviços subsequentes de uma rota. Caso o último atendimento seja alcançado dentro de sua janela de tempo, então a solução é considerada factível. Na Figura 2 estes tempos são representados pela linha preta à esquerda da área cinza.

Os tempos mais tarde são calculados cumulativamente partindo do serviço final, seguido por iterações sobre os serviços precedentes de uma rota, de forma analogicamente

inversa ao cálculo dos tempos mais cedo. Na Figura 2 estes tempos são representados pela linha preta à direita da área cinza.

A área pintada de cinza na Figura 2 representa a área de factibilidade da solução, isto é, para cada ponto dentro desta área é possível que o veículo complete a rota atendendo a todos os clientes na janela de tempo estipulado. Assim, na vida real, caso ocorra atrasos ou adiantamentos, é possível facilmente verificar os impactos ocorridos nos outros atendimentos, isto é, se mesmo com essa modificação do horário programado o veículo ainda se encontra dentro da região de factibilidade, então ele conseguirá concretizar a rota e o atendimento dos outros clientes. Portanto, a área de factibilidade é matematicamente representada pelo tempos de folga entre o *earliest arrival* e *latest departure*.

Entretanto, o objetivo principal do MTDP não é apenas encontrar e determinar a factibilidade de uma rota, mas sim determinar o melhor caminho possível de forma a minimizar a duração da jornada do veículo. Para isso, o processo de compressão da duração das viagens permite minimizar ao máximo os tempos de folga de uma rota, fazendo com que a duração total da viagem seja reduzida (Belhaiza; M'Hallah, 2016).

Após o processo de compressão de jornada da rota, o tempo de espera, e, consequentemente os tempos de folgas, serão minimizados, fazendo com que o tempo de duração total da jornada do veículo esteja minimizado. Esta área é representada pelas áreas e linhas pretas no meio da Figura 2. Embora essa região de otimalidade é formalmente definida por uma área, caso o tempo de folga seja reduzido a zero, área ótima pode ser eventualmente representada por uma linha (área de comprimento zero), de forma que qualquer atraso para fora desta área signifique que o tempo de execução da rota não será ótimo com relação ao planejamento, embora eventualmente ainda possa ser factível.

2.3 O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO (VRPTW)

Em aplicações reais do VRP, especialmente quando aplicada na prática a contextos urbanos, é comum que diversas restrições sejam introduzidas ao problema de forma a tornar o planejamento utilizável pelas empresa logísticas. Frequentemente, condições específicas precisam ser atribuídos a determinados trechos da rota ou a determinados clientes, de forma a satisfazer suas necessidades, mesmo que isso cause um aumento no custo da rota (Ben Ticha et al., 2017). Neste contexto, uma das variantes do VRP mais comumente encontradas na literatura é a adição de janelas de tempo de atendimento para os clientes, caracterizando o Problema de Roteamento de Veículos com Janela de Tempo de Atendimento (*Vehicle Routing Problem with Time Window* – VRPTW).

Esta restrição pode ser imposta no depósito (no início e final da rota), de forma a restringir o horário de partida e chegada do veículo, podendo ser utilizada para limitar a jornada de trabalho do motorista, por exemplo. Também é possível impor restrições de

janelas de tempo no atendimento dos serviços, possibilitando, por exemplo, definir um intervalo de tempo customizadas de entregas agendadas.

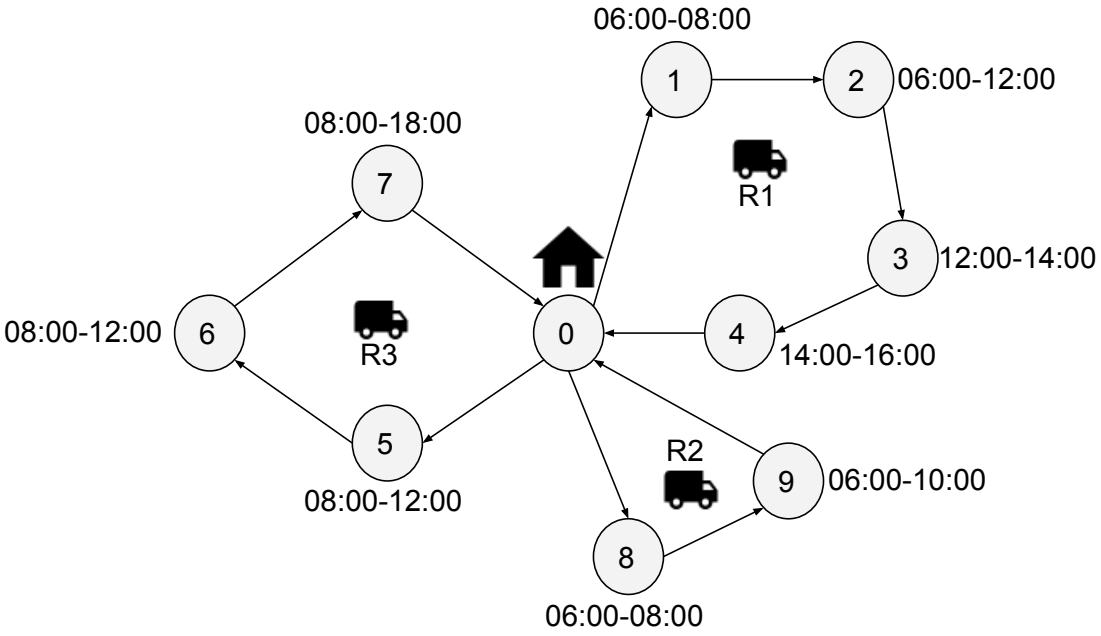
Como exemplo prático do problema envolvendo VRPTW, pode-se citar os serviços de compra e entrega de comida, em que pessoas realizam pedidos e esperam que seus produtos cheguem em um período específico de tempo, tendo em vista que este produto pode "expirar" caso fique muito tempo em rota, ou desagradar o cliente caso chegue muito antes ou depois do período esperado (PRATAMA; MAHMUDY, 2017). Também pode-se citar o caso de prestação de serviços *in loco*, comum em serviços de reparos domésticos, em que frequentemente a realização do atendimento não é instantânea, isto é, após a chegada do veículo existe um Tempo de Atendimento do cliente reservado, de forma que o veículo não pode se deslocar em direção ao próximo destino durante este período. Nesta restrição é exigida que apenas o início efetivo do atendimento esteja comportado pelas janelas de tempo, permitindo que o final do atendimento se encontre fora da janela de tempo especificada.

De forma a tornar possível o cálculo de tempo, é necessário que para cada arco do grafo, além do conhecimento do custo de deslocamento, se tenha também o conhecimento do tempo de deslocamento entre clientes, permitindo que seja calculado cumulativamente o tempo corrente em que uma requisição será atendida (Ben Ticha et al., 2017).

A Figura 3 apresenta um exemplo do VRPTW para um cenário fictício. Para este exemplo são utilizados 3 veículos, todos com capacidade de 5 unidades de carga, um depósito representado pelo nó 0 (com janela de tempo $[00:00, 24:00]$) e 9 requisições representadas pelos nós 1 ao 9 (com janelas de tempo variando de acordo com a figura e demanda de 1 unidade de carga e $00:15$ de tempo de serviço), assumindo que o tempo de deslocamento entre os nós é de $01:00$. Por questões de simplicidade de ilustração alguns parâmetros como demanda, tempo de serviço e tempo de viagem foram definidos iguais, porém é totalmente admissível que eles sejam distintos. A Tabela 1 detalha o planejamento das rotas apresentadas na Figura 3.

De acordo com Li e Lim (2002), o VRPTW é uma generalização do problema clássico do VRP com a complexidade adicional de restrições de janela de tempo. A formulação do problema é dado por um grafo $G = (V, E)$. A variável $V = \{v_0, v_1, ..., v_n\}$ representa o conjunto de vértices do grafo, simbolizando as localizações existentes referentes ao contexto do problema, sendo v_0 é considerado o depósito de onde o veículo inicia e termina sua jornada, e os outros v_i nós representam as requisições dos clientes. Pra cada $v_i \in V$ existe uma demanda q_i, um tempo de serviço s_i e uma janela de tempo $[e_i, l_i]$, sendo e_i o limite inferior para iniciar o atendimento da requisição e l_i o limite superior para iniciar o atendimento da requisição. A variável $E = \{(v_i, v_j) | v_i, v_j \in V\}$ representa o conjunto de arestas do grafo, representando os custos de deslocamento d_{ij} e tempo de deslocamento t_{ij} dos veículos entre um nó i e um nó j. Também é definida uma capacidade limite Q para a carga da frota de veículos.

Figura 3 – Exemplo de roteamento para o VRPTW.



Fonte: Próprio autor

Tabela 1 – Detalhes dos agendamentos dos atendimentos por rota.

Rota	Seq.	Req.	T. Chegada	T. Atendimento	T. Partida	Carga V.
R1	0	0	-	-	00:00	0
R1	1	1	01:00	06:00	06:15	1
R1	2	2	07:15	07:15	07:30	2
R1	3	3	08:30	12:00	12:15	3
R1	4	4	13:15	13:15	13:30	4
R1	5	0	14:30	-	-	-
R2	0	0	-	-	00:00	0
R2	1	8	01:00	06:00	06:15	1
R2	2	9	07:15	07:15	07:30	2
R2	3	0	08:30	-	-	-
R3	0	0	-	-	00:00	0
R3	1	5	01:00	08:00	08:15	1
R3	2	6	09:15	09:15	09:30	2
R3	3	7	10:30	10:30	10:45	3
R3	4	0	11:45	-	-	-

Fonte: Próprio autor

Dada uma rota como sequência de nós $v_i \in V$ a serem visitados, o custo total da viagem é dada pela soma dos custos de deslocamento (d_{ij}), e o tempo total de viagem de cada rota é dado pela soma do tempo de espera, tempo de serviço (s_i) e tempo de deslocamentos (t_{ij}). Para permitir o cálculo do tempo de espera, a seguinte sequência cronológica de eventos precisa ser calculada para cada etapa do caminho calculado:

- **Tempo de chegada**: Soma do tempo de partida do horário anterior mais o tempo de deslocamento.
- **Tempo do atendimento (início do atendimento)**: caso o tempo de chegada do veículo seja posterior ou igual ao limite inferior da janela de atendimento do cliente, o atendimento pode iniciar imediatamente, fazendo com que o tempo de início do atendimento seja igual ao tempo de chegada e que o tempo de espera seja zero. Caso contrário, se o tempo de chegada do veículo for anterior ao limite inferior da janela de atendimento do cliente, o atendimento não pode iniciar imediatamente, sendo necessário que o veículo aguarde até que o tempo de limite inferior da janela de tempo se concretize, de forma que o tempo de início de atendimento seja igual ao limite inferior da janela de tempo de atendimento, enquanto o tempo de espera seja igual a diferença entre o tempo de chegada e o tempo de atendimento.
- **Tempo de partida (final do atendimento)**: dado pela soma do tempo de atendimento mais o tempo de serviço do cliente.

Portanto, para criar uma solução factível ao VRPTW, além de satisfazer as restrições do VRP clássico, que involuem o limite de carga dos veículos, também é necessário que o tempo de atendimento de cada cliente, conforme cálculo explanado acima, esteja dentro dos limites da janela de tempo $[e_i, l_i]$ da requisição.

Como objetivo, o VRPTW pode ser tanto a redução do custo de deslocamento, redução de número de veículo necessários, redução do tempo de deslocamento, redução do tempo de espera, redução do tempo total de viagem (incluindo tempo de espera e de serviço), ou uma associação entre eles, contanto que as restrições detalhadas anteriormente sejam cumpridas.

2.4 O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM MÚLTIPLAS JANELAS DE TEMPO (VRPMTW)

O Problema de Roteamento de Veículos com Múltiplas Janelas de Tempo (*Vehicle Routing Problem with Multiple Time Windows* - VRPMTW) é uma generalização do VRPTW, e, portanto, do VRP, onde é inserida a restrição de que cada serviço possui uma ou mais janelas de tempo em que ele pode ser atendido (BELHAIZA et al., 2018). Todas as requisições devem ser atendidas de forma que apenas uma única janela de tempo seja utilizada para atender cada cliente (LI et al., 2020). No decorrer da pesquisa deste trabalho foram encontradas algumas nomenclaturas distintas para este problema, mas neste trabalhos este problema será referenciado como o problema VRPMTW.

Enquanto no VRP não existe restrições de tempo, isto é, qualquer requisição pode ser atendida a qualquer momento, e no VRPTW existe uma única restrição de tempo por requisição, em que ela deve ser atendida entre um tempo de início e um tempo de fim, no

VRPMTW a requisição precisa ser atendida no intervalo de uma das várias possíveis janelas de tempo existentes (Belhaiza; M'Hallah; Ben Brahim, 2017). Essa nova flexibilização aumenta ainda mais o tamanho do espaço de busca e, por consequência, a complexidade do problema (HOOGEBOOM et al., 2020).

O VRPMTW surge naturalmente quando os clientes podem ser atendidos em períodos de tempo diferentes. Larsen e Pacino (2019) argumentam que forçar a modelagem do problema de forma a atender um cliente em uma única janela de tempo pode acabar causando atrasos ou perturbações desnecessárias no restante da rota, o que acaba aumentando seu custo. As múltiplas janelas de tempo visam dar mais flexibilidade para a escolha dos horários de atendimento do serviço, ao mesmo tempo que podem ser mais adequadas para refletir o problema da vida real. Por exemplo, considerando um planejamento de rotas de longa distância, que possivelmente se entende por diversos dias, alguns clientes podem não estar disponíveis para serem atendidos todos os dias da semana (BELL et al., 1983; RANCOURT; CORDEAU; LAPORTE, 2013), de tal forma que um mesmo cliente possa ser atendido em vários períodos diferentes durante diferentes dias (PUREZA; MORABITO; REIMANN, 2012). Ao mesmo tempo, outros clientes podem não estar disponíveis para serem atendidos em alguns períodos específicos do dia, característica típica de empresas que possuem expediente definido para entregas de cargas em determinado horário, ou que estejam indisponíveis para atendimento durante o intervalo de almoço, por exemplo (LI et al., 2020). Ainda, visando proporcionar maior satisfação entre seus clientes, empresas podem permitir que seus clientes façam agendamentos de atendimentos em diferentes intervalos de tempos, proporcionando maior conforto e comodidade para os usuários (LARSEN; PACINO, 2019; BELHAIZA; HANSEN; LAPORTE, 2014)

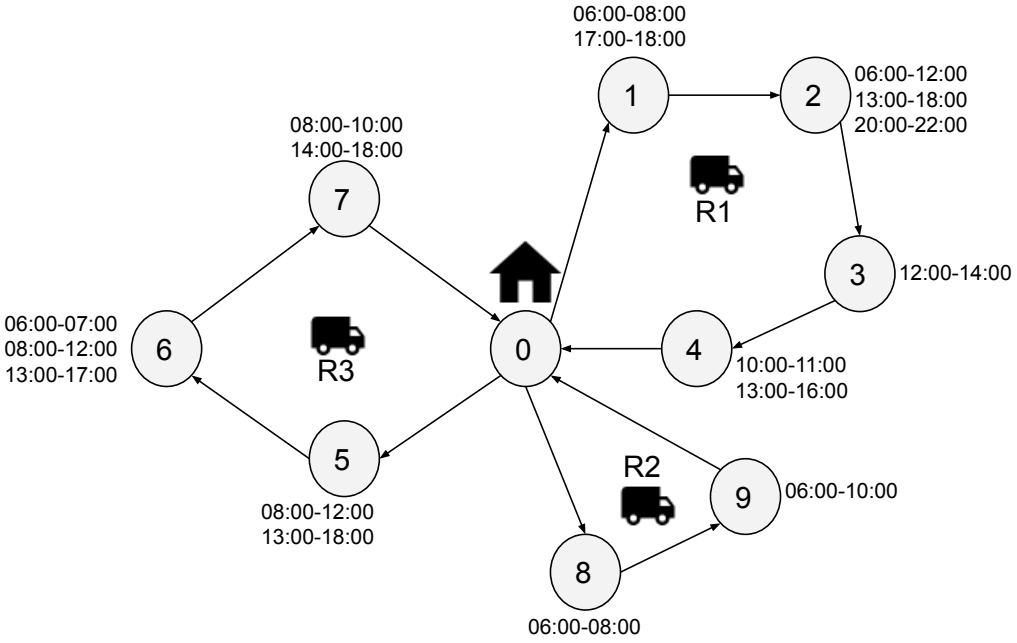
2.4.1 Formulação matemática

De forma sucinta, o VRPMTW é definido como um conjunto de clientes a serem atendidos, cada um com uma quantidade de demanda, um tempo de serviço e um conjunto de janelas de tempo e um conjunto de veículos com capacidade finita. A solução consiste de rotas, partindo do depósito e retornando ao depósito, que possam atender a todos os clientes uma única vez, cada um em uma única janela de tempo (BELHAIZA et al., 2018). Portanto, se o cliente i possui n janelas de tempo associadas, cada uma em um intervalo $tw_{ij} = [l_{ij}, e_{ij}]$, apenas uma das janelas de tempo tw_{ij} será utilizada, sendo que o tempo atual de atendimento t deve estar contido dentro do intervalo tw_{ij} selecionado (Bitao; Fei, 2010). O objetivo a ser otimizado, depende da aplicação, mas pode ser a utilização do número mínimo de veículos, a minimização do deslocamento, a diminuição do tempo de deslocamento, ou a diminuição da jornada do motorista (tempo entre saída e retorno ao depósito).

A Figura 4 apresenta um exemplo do VRPMTW para um cenário fictício. Para este exemplo são utilizados 3 veículos, todos com capacidade de 5 unidades de carga, um

depósito representado pelo nó 0 (com janela de tempo $[00:00, 24:00]$) e 9 requisições representadas pelos nós 1 ao 9 (com janelas de tempo variando de acordo com a figura) e demanda de 1 unidade de carga e $00:15$ de tempo de serviço), assumindo que o tempo de deslocamento entre os nós é de $01:00$. Por questões de simplicidade de ilustração, alguns parâmetros como demanda, tempo de serviço e tempo de viagem foram definidos iguais, porém é totalmente admissível que eles sejam distintos.

Figura 4 – Exemplo de roteamento para o VRPMTW



Fonte: Próprio autor

A Tabela 2 mostra o planejamento das rotas apresentadas na Figura 4, detalhando para cada atendimento do veículo um identificador da rota (Rota), a sequência do atendimento dentro da rota (Seq.), o identificador do nó do cliente (Req.), a janela de tempo selecionada (J. T. Utilizada), o tempo de chegada mais cedo do veículo o cliente (T. Chegada), o tempo de atendimento mais cedo do veículo ao cliente (T. Atend.), o tempo de partida mais cedo do veículo da requisição (T. Partida), além da carga acumulada presente no veículo (Carga V.).

Para formular matematicamente o VRPMTW, será utilizada a mesma terminologia de variáveis sugerida por (BELHAIZA; HANSEN; LAPORTE, 2014). Neste cenário é considerado a existência de múltiplos depósitos, frota finita de veículos com capacidade e limites heterogêneos.

O VRPMTW é definido como um grafo direcionado $G(V, E)$, em que V é o conjunto de vértices do grafo e E é o conjunto de arestas. O conjunto de vértices V é particionado em $V = N, D$, em que $N = 1, ..., n$ é o conjunto de requisições de clientes e D é o conjunto de depósitos existentes. Cada cliente $i \in N$ possui uma quantidade numérica de demanda q_i, um tempo de serviço não negativo s_i, e um conjunto $W_i = [l_i^p, u_i^p], p = 1, ..., p_i$ de p_i

Tabela 2 – Detalhes dos agendamentos dos atendimentos por rota com múltiplas janelas de tempo.

Rota	Seq.	Req.	J.T. Utilizada	T. Chegada	T. Atend.	T. Partida	Carga V.
R1	0	0	00:00 - 24:00	-	-	00:00	0
R1	1	1	00:00 - 24:00	01:00	06:00	06:15	1
R1	2	2	00:00 - 24:00	07:15	07:15	07:30	2
R1	3	3	00:00 - 24:00	08:30	12:00	12:15	3
R1	4	4	00:00 - 24:00	13:15	13:15	13:30	4
R1	5	0	00:00 - 24:00	14:30	-	-	-
R2	0	0	00:00 - 24:00	-	-	05:00	0
R2	1	8	00:00 - 24:00	06:00	06:00	06:15	1
R2	2	9	00:00 - 24:00	07:15	07:15	07:30	2
R2	3	0	00:00 - 24:00	08:30	-	-	-
R3	0	0	00:00 - 24:00	-	-	12:00	0
R3	1	5	00:00 - 24:00	13:00	13:15	13:15	1
R3	2	6	00:00 - 24:00	14:15	14:15	14:30	2
R3	3	7	00:00 - 24:00	15:30	15:30	15:45	3
R3	4	0	00:00 - 24:00	16:45	-	-	-

Fonte: Próprio autor

janelas de tempo. O tempo de viagem associado a aresta $(i, j) \in A$ é definido por t_{ij}. O conjunto de m veículos existentes é representado por R. Para cada veículo $k \in R$ são definidas as variáveis Q_k que representa a capacidade máxima do veículo k e a variável D_k representa a duração máxima da rota do k. A Tabela 3 detalha todas as variáveis do problema.

O VRPMTW é formulado por meio da otimização da função objetivo (*fitness*) denotada pela Equação 1a. Esta função de otimização está sujeita a todas as restrições do problema. A seguir estão detalhados estas equações de otimização e que definem e restringem o espaço de busca do problema.

Tabela 3 – Variáveis da formulação do VRPMTW

Variáveis	**Tipo**	**Descrição**
x_{ij}^k	Binário	Igual a 1 se e somente se a aresta (i, j) é percorrida pelo veículo k.
y_{ij}^k	Real	Demanda carregada na aresta pelo veículo k (i, j).
r^k	Binário	Igual a 1 se e somente se o veículo k é usado.
v_i^p	Binário	Igual a 1 se e somente se o cliente i é atendido na sua janela de tempo p.
w_i^k	Real	Tempo de espera do veículo k no cliente i.
z_i^k	Binário	Igual a 1 se e somente se o cliente i está na rota do veículo k.
q_d^k	Real	Demanda carregada pelo veículo k no depósito d.
d_k	Real	Duração da rota do veículo k.
a_i^k	Real	Tempo de chegada do veículo k no cliente i.
b_i^k	Real	Tempo de partida do veículo k no cliente i.
Parâmetros	**Tipo**	**Descrição**
t_{ij}	Real	Tempo de viagem entre as arestas i e j.
q_i	Real	Demanda associada com o vértice do cliente i.
l_i^p	Real	Tempo de início da janela de tempo p do cliente i.
u_i^p	Real	Tempo de final da janela de tempo p do cliente i.
Q_k	Real	Capacidade do veículo k.
D_k	Real	Duração máxima da rota do veículo k.
s_i	Real	Tempo de atendimento no cliente i.
e_d^k	Binário	Igual a 1 se e somente se a rota do veículo k inicia e finaliza no depósito d.
B	Binário	Igual a 1 se e somente se a duração total da rota deve ser minimizada.
F^k	Real	Custo para utilização do veículo k (em unidades de tempo).
M	Real	Constante grande arbitrária.

$$\min \quad \sum_{k\in R}\sum_{i\neq q} t_{ij} \times x_{ij}^k + B \times \sum_{k\in R}\sum_{i\in N} w_i^k + \sum_{k\in R} F^k r^k \tag{1a}$$

s.t.

$$\sum_{k\in T_i} z_i^k = 1,\ i \in V, \tag{1b}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = 0,\ i \in V, k \in R, \tag{1c}$$

$$2 \times x_{ij}^k \leq z_i^k + z_j^k,\ i,j \in V, k \in R, \tag{1d}$$

$$\sum_{k\in T_i\cap T_j}\sum_{j\in V} X_{ij}^k \leq 1,\ v \in V, \tag{1e}$$

$$\sum_{k\in T_i\cap T_j}\sum_{j\in V} X_{ji}^k \leq 1,\ v \in V, \tag{1f}$$

$$y_{ij}^k \leq Q_k x_{ij}^k,\ i,j \in V, k \in R, \tag{1g}$$

$$\sum_{j\in V} y_{dj}^k - \sum_{i\in V} y_{id}^k = q_d^k e_d^k,\ d \in D, k \in R, \tag{1h}$$

$$\sum_{j\in V} y_{ji}^k - \sum_{j\in V} y_{ij}^k \geq q_i z_i^k,\ i \in N, k \in R, \tag{1i}$$

$$b_d^k \geq l_d - M(1 - z_d^k),\ d \in D, k \in R, \tag{1j}$$

$$a_d^k \leq u_d + M(1 - z_d^k),\ d \in D, k \in R, \tag{1k}$$

$$a_d^k - b_d^k \leq D_k + M(1 - z_d^k),\ d \in D, k \in R, \tag{1l}$$

$$b_i^k \geq a_i^k + w_i^k + s_i - M(1 - z_i^k),\ i \in N, k \in R, \tag{1m}$$

$$a_j^k \geq b_i^k + c_{ij} - M(1 - x_{ij}^k),\ i,j \in V, k \in R, \tag{1n}$$

$$a_j^k \leq b_i^k + c_{ij} + M(1 - x_{ij}^k),\ i,j \in V, k \in R, \tag{1o}$$

$$a_i^k + w_i^k \geq l_i^p - M(1 - z_i^k) - M(1 - v_i^p),\ i \in N, p \in W_i, k \in R, \tag{1p}$$

$$a_i^k + w_i^k \leq u_i^p + M(1 - z_i^k) + M(1 - v_i^p),\ i \in N, p \in W_i, k \in R, \tag{1q}$$

$$\sum_{p=1}^{p_i} v_i^p = 1,\ i \in N \cap D, \tag{1r}$$

$$r^k \geq z_i^k,\ i \in V, k \in R, \tag{1s}$$

$$y_{ij}^k, w_i^k, q_d^k, d_k, a_i^k, b_i^k \geq 0, \tag{1t}$$

$$r^k, x_{ij}^k, v_i^p, z_i^k,\ Binario \tag{1u}$$

A função objetivo do problema a ser minimizada é representada pela Equação 1a. Esta equação é composta pela soma de três termos, sendo o primeiro responsável por minimizar o tempo de deslocamento em viagem, o segundo é responsável por minimizar o tempo de espera (se o valor parâmetro B for igual a 1), e o terceiro responsável por minimizar a frota de veículos. Utilizando esta equação, o custo da rota é calculado com base em tempo, de forma que todos os seus componentes devam ter seu custo em tempo,

ou convertido para um custo proporcional em tempo. Como esta função objetivo pretende minimizar estes diferentes fatores, o problema trata-se de um problema de otimização de múltiplos objetivos.

A seguir são detalhadas as descrições das restrições que permeiam a formulação mencionada:

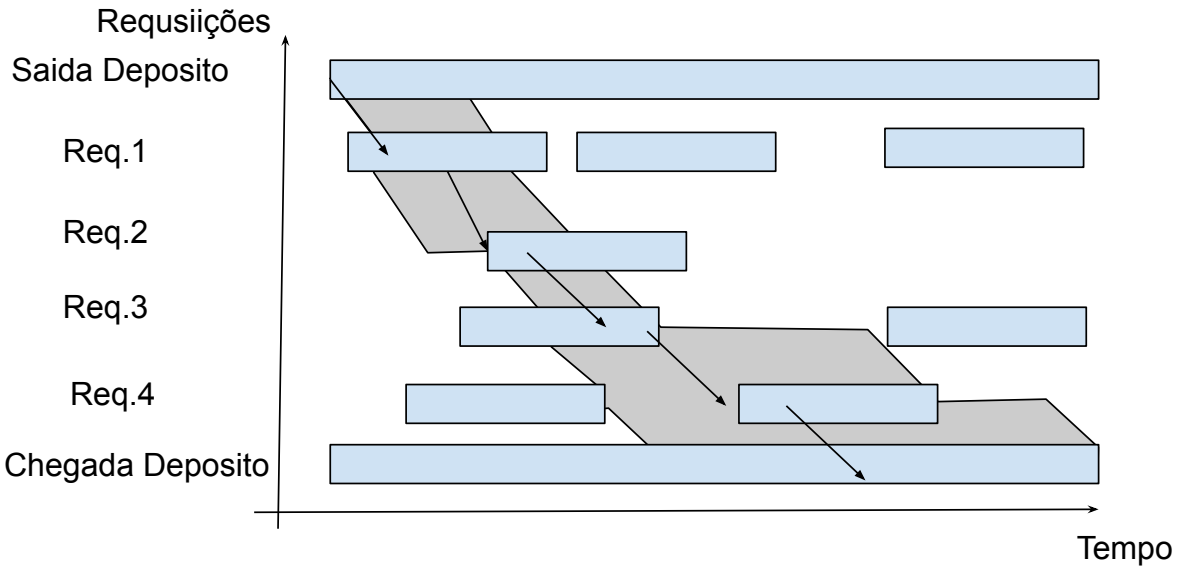
- Restrição 1b: garante que cada cliente se encontra na rota de exatamente um veículo;
- Restrição 1c: garante que cada rota do veículo k inicia e finaliza no depósito, e o número de arestas entrando no vértice i é igual ao número de arestas saindo do vértice i;
- Restrição 1d: garante que qualquer aresta (i, j) pode ser percorrida pelo veículo k somente se ambos z_i^k e z_j^k são iguais a 1;
- Restrição 1e e 1f: força que todo cliente i tenha que ser visitado por um veículo;
- Restrição 1g: garante que a carga na aresta (i, j) é menor que a capacidade Q_k do veículo k passando por esta aresta;
- Restrição 1h: garante que as demandas dos clientes visitados pelo veículo k são satisfeitas;
- Restrição 1i: garante que a demanda de cada cliente da rota do veículo k é satisfeita;
- Restrição 1j: garante que as janelas de tempos devem ser respeitadas;
- Restrição 1k: garante que o tempo de chegada do veículo k no depósito deve ser menor ou igual a u_d;
- Restrição 1l: garante que a duração total de cada rota não pode exceder a duração máxima D_k;
- Restrição 1m: garante que o tempo de partida do cliente i é superior ou igual ao tempo de chegada adicionado ao tempo de espera e ao tempo de serviço do cliente i somente se o cliente i for atendido pelo veiculo k;
- Restrição 1n e 1o: garante que o tempo de chegada do cliente j é igual ao tempo de partida do cliente i adicionado o custo da aresta (i, j) apenas se a aresta (i, j) for usada pelo veículo k;
- Restrição 1p e 1q: garante que o tempo de chegada acrescido do tempo de espera do veículo k no cliente i está entre os limites da janela de tempo $[l_i^p, u_i^p]$ somente se o cliente i estiver na rota do veículo k e a janela de tempo p for escolhida;

- Restrição 1r: garante que exatamente uma única janela de tempo é escolhida para cada cliente i;
- Restrição 1s: garante que o cliente i é atendido pelo veículo k apenas se o veículo for utilizado;
- Restrição 1t e 1u: define os intervalos de factibilidade para as variáveis de decisão;

2.4.2 Agendamento de Múltiplas Janelas de Atendimento

A Figura 5 apresenta um exemplo do problema de agendamento para múltiplas janelas de tempo de uma única rota. A representação gráfica da Figura 5 é semelhante ao da Figura 2, entretanto a Figura 5 apresenta mais de um retângulo azul por linha, indicando assim a existência de mais de uma opção de janela de tempo que pode ser utilizada para atender ao cliente.

Figura 5 – Exemplo de otimização de linha do tempo para o VRPMTW



Fonte: Próprio autor

Nesse cenário, dado que as janelas de tempo são fixas e as ordem das requisições é previamente definida pelo método de otimização, o problema em questão é determinar de forma ótima quais janelas de tempo serão utilizadas e qual o instante que elas serão utilizadas para cada requisição, visando minimizar o tempo de serviço do veículo (considerando o tempo de deslocamento e o tempo espera). Ou seja, diminuir o máximo possível os tempos de folga entre o *earliest arrival* e *latest departure*.

Na Figura 5 a área cinza representa a área de factibilidade, matematicamente representada pelos tempos de folga calculados entre o *earliest arrival* e o *latest departure*

de cada serviço contido na rota, que são interpolados em ordem crescente da sequência de atendimento dos serviços. Portanto, a área de factibilidade indica que caso o veículo mantenha seu deslocamento e atendimento dentro do tempo delimitado por ela, é possível que o veículo complete a rota atendendo a todos os serviços dentro do tempo esperado. Assim, na vida real, caso ocorram atrasos ou adiantamentos é possível facilmente verificar o impactos na factibilidade dos outros serviços, ou seja, se mesmo com essa modificação do horário programado o veículo ainda se encontra dentro da região factível, então ele conseguirá concretizar a rota e o atendimento dos outros serviços, mesmo que eventualmente fora do horário ou que não garanta mais a solução ótima.

Ainda com base na área de factibilidade apresentada na Figura 5, é possível constatar que algumas janelas de tempo tornam a rota infactível se utilizadas. Isso acontece porque elas estão fora da área de factibilidade (fora da zona cinza). Isso pode ser observado nas duas últimas janelas de tempo da Requisição 1, em que é necessário que a primeira janela de tempo seja utilizada para que uma solução factível seja alcançada. Caso parecido também com a Requisição 3, em que a última janela de tempo está fora da região de factibilidade. Também é possível constatar que algumas janelas de tempo se encontram parcialmente dentro da região factível. É o exemplo da primeira janela de tempo da Requisição 1, em que para manter a factibilidade da solução o veículo precisa partir antes do final da janela de tempo, caso contrário, isso acarretaria em um atraso em cadeia que poderia fazer com que a Requisição 3 não fosse atendida a tempo, mesmo que a Requisição 3 tenha outra janela de tempo disponível. Entretanto, essa outra janela de tempo da Requisição 3 está fora da região factível e caso seja utilizada implicaria que a Requisição 4 não pudesse ser atendida em tempo hábil, tornando a solução infactível. Outro exemplo de janela de tempo parcialmente infactível é o caso da primeira janela de tempo da Requisição 3, onde ela é factível apenas na segunda metade, tendo em vista que não é possível ter um planejamento que permita ao veículo chegar antes deste período.

2.5 MÉTODOS PARA SOLUÇÃO DO VRP

O problema de Roteamento de Veículos é um problema de otimização combinatória que pode ser retratado por uma formalização matemática, que o torna de grande relevância para a área de pesquisa operacional. Sendo assim, é natural que sejam desenvolvidas diversas abordagens distintas que permitam resolver o problema proposto. As primeiras abordagens utilizadas para resolver este problema eram soluções exatas, utilizando algoritmos para solução de problemas de Programação Linear Inteira (PLI), permitindo comprovadamente encontrar soluções ótimas.

Contudo, o VRP é considerado um problema NP-Difícil (DONG et al., 2018). Sendo assim, conforme aumenta o tamanho do problema, ocorre uma explosão combinatória de possibilidades, o que torna inviável realizar a busca da solução ótima utilizando métodos

exatos devido ao enorme tamanho do espaço de busca, o que faz com que o consumo de recursos computacionais e tempo de processamento cresçam exponencialmente, se tornando uma abordagem incompatível (SAVELSBERGH, 1990). Portanto, algoritmos baseados em métodos exatos são considerados ineficientes para resolução de instâncias de tamanho não trivial do problema (PAN, 2012).

De forma a permitir alcançar soluções para instâncias maiores do VRP, outros métodos precisam ser empregados. Nas últimas décadas vêm se popularizando a proposição de métodos de aproximação de solução ótima, tais como heurísticas e meta-heurísticas, que não garantem que serão capazes de encontrar a solução ótima, mas que poderão ser capazes de encontrar uma solução próxima da ótima, exigindo tempo computacional reduzido (Sakakibara; Tamaki; Nishikawa, 2007).

Dentre as possibilidades para solução do problema existem diferentes abordagens, tais como métodos exatos, heurísticas e meta-heurísticas de aproximação (OYOLA; ARNTZEN; WOODRUFF, 2017; OYOLA; ARNTZEN; WOODRUFF, 2018). A Figura 6 apresenta uma taxonomia de métodos utilizados para solucionar o VRP, conforme classificado por Danilova (2018). Nas próximas subseções são introduzidas brevemente as principais características dos métodos apresentados na Figura 6, bem como são destacados os métodos mais utilizados.

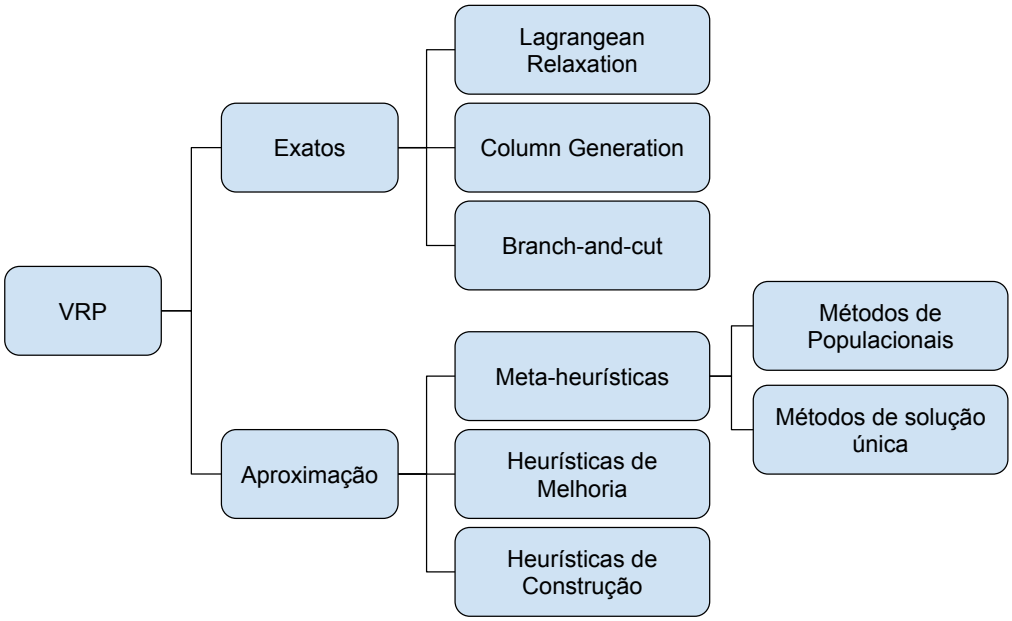
2.5.1 Métodos Exatos

Conforme mencionado anteriormente, métodos exatos permitem encontrar a solução ótima para o VRP, porém são aplicáveis apenas para instâncias pequenas do VRP devido a sua alta complexidade computacional.

Pelo fato de o VRP ser habitualmente modelado de forma matemática, não é incomum observar propostas de solução utilizando Programação Linear Inteira (PLI), visto que é possível integrar as equações que definem o VRP dentro de métodos de solução PLI já existentes. Entre os possíveis métodos de solução PLI aplicados ao VRP podem ser citados *Lagrangean Relaxation*, *Column Generation* e *Branch-and-bound*, entre outros.

Além dos métodos analíticos baseados na PLI, outro método exato aplicado à solução de problemas combinatórios é a Programação Dinâmica (*Dynamic Programming*) (KOK et al., 2009). Esse método consiste na formulação de estados para os diferentes resultados parciais de um problema. Assim, a Programação Dinâmica é frequentemente utilizada para problemas de otimização combinatória em que existe superposição de subproblemas (LINTZMAYER; MOTA, 2018). Com a memorização dos estados parciais que caracterizam um subproblema e seu resultado, é possível evitar o cálculo de um mesmo subproblema por diversas vezes, fazendo com que eles sejam calculados apenas uma vez e seu resultado seja reutilizado. Embora o método de Programação Dinâmica permita reduzir o tempo computacional de resolução do problema, ele ainda realiza uma busca exaustiva no espaço de busca, o que o torna impraticável para instâncias grandes do problema.

Figura 6 – Taxonomia de métodos de solução do VRP.



Fonte: Próprio autor

2.5.2 Heurísticas de Construção/Inserção

Em problemas de roteamento, heurísticas de construção, também conhecidas como heurísticas de inserção, são mecanismos empregados para realizar a criação de uma nova rota, ou inserção de novas requisições em rotas já existentes. O algoritmo visa realizar análises no entorno da solução de uma rota vazia ou existente, considerando a adição de uma nova requisição a ser atendida, procurando por soluções que atendam aos requisitos do problema e que incorporem o novo nó, sem prejudicar de forma significativa a solução existente.

Grande parte destas heurísticas funcionam de forma gulosa, buscando sucessivamente inserir os clientes, a cada passo, segundo o critério de inserção de menor curso.. Embora estas heurísticas possam criar uma nova rota com a nova requisição de forma rápida e eficiente, com baixo custo computacional, elas não garantem que a melhor solução possível seja encontrada. Isto é, elas são ferramentas interessantes no processo de busca para encontrar um ótimo local, porém, por si só, geralmente carecem de robustez para encontrar um ótimo global.

A seguir são elencadas e detalhadas algumas das principais heurísticas encontradas

na literatura, conforme expostas por Larsen e Pacino (2019), Aghezzaf e Fahim (2014), Ghilas, Demir e Van Woensel (2016):

- *Best Insert*: Essa heurística visa inserir o novo nó em uma rota existente, ou vazia, na melhor posição possível de forma gulosa. Assim, ela visa encontrar uma posição que entre todas as posições possíveis maximize a redução do custo da rota.
- *Second Best Insert*: Essa heurística visa inserir o novo nó em uma rota existente, ou vazia, na segunda melhor posição possível de forma gulosa. Portanto, ela visa encontrar uma posição que entre todas as posições possíveis maximize a redução do custo da rota, porém de forma não tão gulosa quanto o operador *Best Insert*, pois, ao invés de utilizar a melhor opção, a segunda melhor opção é utilizada, permitindo que a solução seja melhorada mas reduzindo a curva de convergência.
- *Random Insert*: Essa heurística visa inserir o novo nó em uma posição randômica de uma rota existente, ou vazia. Portanto, ela não visa construir uma solução necessariamente melhor, mas sim ser rápida e introduzir maior diversificação na nova solução gerada e a construindo de forma não enviesada. Esse tipo de inserção assume que existe um mecanismo implementado que permite reestabelecer a factibilidade da solução.
- *First Feasible*: Essa heurística visa inserir o novo nó em uma posição factível de uma rota existente, ou vazia. Dessa forma, ela não visa construir uma solução necessariamente melhor, mas sim ser rápida e introduzir o novo nó de forma a não quebrar a factibilidade da solução (o que acarretaria que o resto do algoritmo fosse tolerante a soluções infactíveis).
- *Best out of λ-feasible insertions*: Essa heurística visa inserir o novo nó em uma posição factível de rota existente, ou vazia. Ela não visa construir uma solução necessariamente melhor, tampouco degrada-la, mas sim ser um bom equilíbrio entre encontrar uma solução boa e rápida, introduzindo o novo nó de forma a não quebrar a factibilidade da solução (o que acarretaria que o resto do algoritmo fosse tolerante a soluções infactíveis). Um número máximo λ de soluções possíveis são pesquisadas e a melhor solução entre elas é selecionada.
- *Regret-k*: Essa heurística é semi-gulosa, pois utiliza critérios de arrependimento ao tomar uma decisão. Portanto, ela leva em consideração o impacto futuro da modificação da solução, onde é considerado o custo da melhor inserção e comparado com o impacto nas demais opções de inserção futura. Esse algoritmo considera as k melhores modificações a serem feitas e calcula o valor de arrependimento simulando uma futura modificação nessas k opções. A opção selecionada é a que

possuir o menor valor de arrependimento, isto é, a menor diferença entre os ganhos de inserções.

2.5.3 Heurísticas de Melhoria

Heurísticas de melhoria, também conhecidas como heurísticas de refinamento, ou busca local, são mecanismos empregados para realizar melhorias em uma solução já existente (possivelmente gerada por uma heurística construtiva). O algoritmo visa realizar buscas de forma rápida no entorno da solução existente, com baixo custo computacional, mas que possivelmente permita melhorar a solução de forma significativa.

Grande parte destas heurísticas funcionam de forma gulosa, buscando sucessivamente pela melhor solução possível no momento. Embora essa abordagem possa melhorar significativamente uma solução, as heurísticas não garantem que a melhor solução possível seja encontrada. Isto é, elas são ferramentas interessantes no processo de intensificação da busca para encontrar um ótimo local, porém geralmente carecem de robustez para encontrar um ótimo global (melhor solução possível).

A seguir são elencadas e detalhadas as principais heurísticas de melhoria encontradas da literatura, conforme expostas por Hurkała (2015), Ghilas, Demir e Van Woensel (2016), Lin e Yu (2015), Bitao e Fei (2010), Belhaiza, Hansen e Laporte (2014), Belhaiza (2018):

- *Best Adjacent Swap*: Essa heurística visa trocar a ordem de atendimento de dois clientes adjacentes um ao outro na mesma rota, selecionando como melhor resultado aquele que maximiza, entre todas as possíveis alterações, a redução do custo da rota.

- *Best Swap / Best Exchange*: Essa heurística visa trocar a posição de atendimento de dois clientes quaisquer (na mesma rota ou não), selecionando como melhor resultado aquele que maximiza, entre todas as possíveis alterações, a redução do custo da rota.

- *Random Swap / Random Exchange*: Essa heurística visa trocar a posição de atendimento de dois clientes quaisquer (na mesma rota ou não), selecionando os clientes de forma aleatória. Embora esse operador não gere necessariamente uma solução melhor do que a original, pode ser utilizado em meta-heurísticas para perturbar a solução atual, de forma a adicionar diversidade e escapar de um atrator local.

- *Best Move / Best Relocate*: Essa heurística visa mover a posição de atendimento de um clientes (na mesma rota ou não), isto é, o cliente é removido de uma rota e reinserido na melhor rota, selecionando como melhor resultado aquele que maximiza, entre todas as possíveis alterações, a redução do custo da rota.

- *Random Move / Random Relocate*: Essa heurística visa mover a posição de atendimento de um clientes (na mesma rota ou não), isto é, o cliente é removido de uma rota e reinserido em outra posição de forma aleatória. Embora este operador não gere necessariamente uma solução melhor do que a original, pode ser utilizado em meta-heurísticas para perturbar a solução atual, de forma a adicionar diversidade e escapar de um atrator local.

- *2-OPT / Best Range Inversion*: Essa heurística visa selecionar dois atendimentos i e j em uma rota e realizar a inversão da sequência entre esses pontos, ou seja, ela inverte uma sequência de atendimentos de um trecho da rota de forma que a seleção de i e j, entre todas as possíveis combinações, maximizem a redução do custo da rota.

- *Random Range Inversion*: Essa heurística visa selecionar dois atendimentos aléatórios i e j em uma rota e realizar a inversão da sequência entre eles, ou seja, ela inverte uma sequência de atendimentos de um trecho da rota de forma que a seleção de i e j. Embora esse operador não gere necessariamente uma solução melhor do que a original, pode ser utilizado em meta-heurísticas para perturbar a solução atual, de forma a adicionar diversidade e escapar de um atrator local.

- *Relocate Range*: Essa heurística visa selecionar dois atendimentos i e j em uma rota e realizar o translado da sequência de atendimentos entre eles para outra rota ou em outra posição na mesma rota. Portanto, ela remove e adiciona uma sequência de atendimentos de forma que a seleção de i e j de remoção, da rota e posição de inserção, entre todas as possíveis combinações, maximizem a redução do custo da rota.

- *I-exchange / Inverted Range Relocate*: Essa heurística visa selecionar dois atendimentos i e j em uma rota e realizar o translado desta sequência de forma invertida para outra rota ou em outra posição na mesma rota. Portanto, ela remove e adiciona de forma invertida uma sequência de atendimentos de forma que a seleção de i e j de remoção, da rota e posição de inserção, entre todas as possíveis combinações, maximizem a redução do custo da rota.

- *Cross-exchange / Range Exchange*: Essa heurística visa selecionar dois atendimentos i e j em uma rota, e outros dois atendimentos k e l em outra rota, e realizar a troca dessas sequência entre as rotas. Ou seja, ela remove a sequência de i a j da primeira rota e insere na posição k da segunda rota, ao mesmo tempo que remove a sequência de k a l da segunda rota e insere na posição i da primeira rota de forma que a seleção de i, j, k e j, entre todas as possíveis combinações, maximizem a redução do custo da rota.

2.5.4 Heurísticas de Remoção

Heurísticas de remoção são mecanismos empregados para destruir parte das rotas, ou seja, são dedicadas a realizar a remoção de algumas das requisições existentes. O algoritmo visa realizar análises no entorno da solução existente, considerando a remoção de requisições contidas nela, procurando por soluções que fiquem mais atraentes ao removerem alguns de seus nós.

O objetivo de uma heurística de remoção, diferentemente das outras heurísticas vistas anteriormente, não é incrementar ou obter uma solução melhor, mas sim desconstruí-la, para permitir que ela seja recriada de forma mais adequada na sequência. Esse comportamento é percebido nos algoritmos em que é realizada a desconstrução de parte da solução, removendo algumas requisições das rotas, de forma que na fase de inserção elas sejam reinseridas de uma forma mais adequada, visando reduzir o custo da rota.

A seguir são elencadas e detalhadas as principais heurísticas de remoção encontradas da literatura, conforme expostas por Ghilas, Demir e Van Woensel (2016), Wang et al. (2018), Aghezzaf e Fahim (2014):

- *Best Remove*: Essa heurística visa remover o nó mais custoso de uma rota existente. Portanto, ela visa encontrar a requisição que, entre todas as possíveis, se removida maximize a redução do custo da rota.

- *Random Remove*: Essa heurística visa remover um nó em uma posição randômica da rota existente. Ela não visa remover uma requisição que necessariamente prejudica a rota, mas sim tem por objetivo ser rápida e perturbar a solução existente, esperando que essa perturbação acabe por remover um máximo local.

- *Interval Remove*: Essa heurística visa remover um trecho de uma rota existente entre dois nós i e j. Portanto, ela visa eventualmente desconstruir uma seção da rota que possa estar desorganizada, para que seus nós possam ser melhor redistribuídos entre outras rotas existentes, ou que seja reconstruída de forma a maximizar a redução do custo da rota.

- *Route Reset*: Essa heurística visa remover um trecho de uma rota existente entre um nó i e o extremo mais próximo da rota (início ou final), de forma a deixar uma das pontas abertas para reestruturação. Portanto, ela visa eventualmente desconstruir uma seção de rota que possa estar desorganizada para que seus nós possam ser melhor redistribuídos entre outras rotas existentes, ou que seja reconstruída de forma a maximizar a redução do custo da rota.

- *Route Remove*: Essa heurística visa remover uma rota por completo da solução. Portanto, ela visa eventualmente desconstruir uma rota de forma que seus nós pos-

sam ser melhor redistribuídos entre outras rotas existentes, ou que seja reconstruída de forma a maximizar a redução do custo da rota.

- *Late Arrival*: Essa heurística visa remover o nó com maior tempo entre o horário de atendimento efetivo e o início da janela de atendimento da requisição. A analogia por trás dessa heurísticas é que se o veículo está chegando muito antes do tempo de atendimento, então essa não é uma alocação eficiente do veículo, pois ele ficara ocioso por muito tempo. Entretanto, se o veículo chegar muito tarde para atender a requisição, esta também não é uma alocação eficiente, pois pode estar "saturando" a rota, pois evita que qualquer outra requisição seja inserida anteriormente, tendo em vista que isso atrasará ainda mais o atendimento até criar uma solução infactível.

- Relacional (SHAW, 1998): Essa heurística visa remover requisições que são similares em determinados aspectos, tais como distância, tempo, demanda, por exemplo. Ao selecionar uma requisição aleatórios, todos as outras requisições que possuem características similares acima de um limite mínimo serão removidas, independente da rota. Essa heurística realiza uma espécie de clusterização que permite que requisições parecidas possam ser melhor alocadas em conjunto. Para cada um dos aspectos pode existir um peso específico, de forma que é possível alterar esse peso para tornar a métrica relevante para apenas um aspecto, como proximidade, por exemplo, podendo nesse caso específico ser utilizado como um outro operador.

2.5.5 Meta-heurísticas

De forma geral, heurísticas são uma boa abordagem para conseguir uma boa resposta ao problema de roteamento de veículo com tempo computacional reduzido. Contudo, esses métodos geralmente carecem de robustez, visto que raramente consegue gerar a solução ótima (Bitao; Fei, 2010). As meta-heurísticas são métodos de busca estocástica sofisticados que aplicam técnicas para conduzir a exploração em busca da melhor solução possível (Sakakibara; Tamaki; Nishikawa, 2007).

As meta-heurísticas são compostas por uma combinação de métodos heurísticos que visam trabalhar de forma conjunta e complementar para explorar o espaço de busca. Portanto, elas são compostas por métodos heurísticos que são utilizados para guiar a busca realizada por outros métodos heurísticos (DORIGO; STüTZLE, 2006). A utilização de meta-heurísticas na solução de problemas permite que se obtenha uma solução de qualidade superior à encontrada apenas por métodos heurísticos sozinhos, mesmo que não garantam que a solução encontrada seja ótima, porém ainda em um tempo aceitável.

É possível classificar as meta-heurísticas de acordo com diversos critérios, tais como: número de soluções sendo exploradas (solução única/busca local ou populacional), modo como a solução é gerada ou aprimorada (meta-heurísticas construtivas ou evolutivas), entre

outros. A seguir são elencadas as principais meta-heurísticas encontradas na literatura para a solução de problemas de VRP:

- Algoritmo Genético (*Genetic Algorithm* - GA): Segundo Khoukhi et al. (2019), algoritmos genéticos são um tipo de procedimento de busca aleatória adaptativa inspirada na evolução de organismos biológicos. A partir de uma população (conjunto) de indivíduos (soluções) representados por cromossomos, diversas gerações (iterações) são criadas com operações como seleção, *crossover* e mutação, visando continuamente recombinar soluções existentes e guiar seu aprimoramento, melhorando o *fitness* (qualidade) dos indivíduos. Lu e Yu (2012) apresentam diversos operadores para realizar essas operações visando obter o melhor resultado para diferentes cenários. Por se tratar de um método que trabalha simultaneamente com múltiplas soluções, esse método possui bons resultados ao lidar com problemas com quantidade considerável de restrições e muitos máximos locais, já que é possível explorar simultaneamente vários segmentos do espaço de busca.

- Otimização por Colônia de Formigas (*Ant Colony Optimization* - ACO): De acordo com Rizzoli et al. (2007), os métodos baseados em atuação de formiga são um conjunto de meta-heurísticas que simulam o comportamento de exploração observado em colônias de formigas. O princípio de funcionamento desse método se baseia em utilizar trilhas de ferormônios para atrair e transmitir informações entre diferentes formigas, de forma que as trilhas (soluções) com maior quantidade de ferormônio (melhores resultados) se tornem as mais atraentes. Esse ferormônio influencia nas sucessivas buscas e faz com que ao final, de forma geral, o caminho com maior quantidade de ferormônio seja o melhor encontrado, sendo, portanto, considerado a melhor solução para o problema. Para Tripathi, Glenn Kuriger e Wan (2009), por se tratar de um algoritmo de busca construtivo, esse método apresenta boa performance para aplicações dinâmicas, tendo em vista que pode facilmente incorporar modificações, além do que o ciclo de *feedback* positivo permite encontrar bons resultados de forma rápida. Contudo, quando o problema apresenta muitos ótimos locais, devido a natureza construtiva do método, nem sempre é possível obter qualidade satisfatória no resultado.

- Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO): Conforme afirmado por Okulewicz e Mandziuk (2014), esse método bio-inspirado de inteligência de enxame (*Swarm Intelligence*) é inspirado na ação de revoada de pássaros, abstraído a um conjunto de partículas independentes (soluções) com suas próprias propriedades (posição, velocidades, inércia, atratores) que realizam a comunicação e troca de informações (influenciam) outras partículas. Desta forma, múltiplos conjuntos de partículas percorrem um espaço de busca, sendo guiados com

base nos atratores existentes (qualidade das soluções encontradas) e pela troca de informações com outras partículas (atratores regionais e globais), de modo que ao final do processo se espera a convergência das partículas em torno de uma solução comum (TAO et al., 2008). Por ser uma abordagem populacional, esse algoritmo consegue lidar bem com problemas multimodais com várias restrições, sem que haja uma penalidade grande de tempo computacional.

- Busca em Vizinhança (*Neighborhood Search* - NS): Consiste em uma meta-heurística de otimização em que ocorre a tentativa de melhoramento de uma solução por meio de sucessivas modificações na solução existente. Sempre que uma nova solução melhor é encontrada, esta é utilizada como base para a próxima rodada de aprimoramento contínuo da solução (POP; FUKSZ; MARC, 2014). Para encontrar o ótimo global, esse algoritmo tenta explorar vizinhanças distantes, ao mesmo tempo que realiza buscas locais para encontrar soluções locais ótimas. Diversas variações do NS podem ser encontradas na literatura, tais como o ALNS (*Adaptive Large Neighborhood Search*), em que o algoritmo continuamente se adapta para melhorar as chances de encontrar uma boa solução por meio da variação dos operadores utilizados para explorar a vizinhança.

- GRASP (*Greedy Randomized Adaptive Search Procedure*): Essa meta-heurística visa iterativamente equilibrar os aspectos gulosos, randômicos e de busca local (SOUFFRIAU; VANSTEENWEGEN; BERGHE, 2013). Portanto, ela funciona como um algoritmo de busca em que suas decisões de construção são tomadas não de forma determinística, sempre em direção ao ótimo, mas atuando de forma semi-gulosa, onde as decisões são feitas com base em vieses provenientes das buscas locais. Esse tipo de meta-heurística é bem mais robusta e computacionalmente custosa quando comparada com uma única heurística, porém pode gerar um resultado mais otimizado, pois não é totalmente enviesada a sempre sugerir uma solução ótima local imediata.

- Recozimento Simulado (*Simulated Anealing* - SA): O seu funcionamento ocorre por meio de uma técnica de simulação de resfriamento de temperatura (FERDI; LAYEB, 2016). Essa temperatura é utilizada para determinar o quão abrangente a procura será realizada no espaço de busca. Durante a etapa inicial a temperatura é alta e a busca é ampla. Com o passar das iterações a temperatura diminui e a busca se torna mais concentrada, convergindo em um ponto percebido com o ótimo global (AFIFI; DANG; MOUKRIM, 2013). Esse algoritmo é uma das mais simples meta-heurísticas de otimização utilizadas para tratar o VRP. Embora seja possível utilizá-lo para solucionar o VRP, muitas vezes ele não é utilizado como a rotina principal de otimização, mas sim como uma rotina auxiliar que otimiza diversos parâmetros dos algoritmos principais (XU; LI; WANG, 2011).

- Busca Larga Adaptativa em Vizinhança / Busca Adaptativa em Vizinhança de Larga Escala (*Adaptive Large Neighborhood Search* - ALNS): Uma das principais características negativas do algoritmo *Simulated Anealing* é a sua limitação na modificação da solução, o que acaba fazendo com que ela tenha maior propensão de ficar presa em ótimos locais. Para contornar esse problema, Shaw (1998) propôs o algoritmo de Busca em Vizinhança de Grande Porte (*Large Neighborhood Search* - LNS), baseado no conceito de relaxamento e re-otimização. Portanto, durante o processo de execução do algoritmo existem diversos ciclos de diversificação (grandes modificações nas solução para mudança de vizinhança) e intensificação (procura por modificações melhores que melhorem a solução corrente) dentro de uma mesma vizinhança, por meio de operadores de destruição e de reconstrução da solução. Mais recentemente, Pisinger e Ropke (2007) propuseram uma extensão ao algoritmo LNS, chamado de Busca Larga Adaptativa em Vizinhança, também chamado de Busca Adaptativa em Vizinhança de Larga Escala (*Adaptive Large Neighborhood Search* - ALNS), que permite utilizar diversas heurísticas como operadores dentro do algoritmo. Portanto, a própria meta-heurística se torna capaz de identificar qual operador deve ser utilizado com base em seu critério de sucesso, o que permite que problemas que requerem diferentes estratégias de otimização utilizem operadores adequados, melhorando o resultado do algoritmo em diferentes cenários (LUTZ, 2014).

2.5.6 Considerações sobre os métodos de solução do VRPMTW

No decorrer da revisão da literatura, diversos métodos de abordagem do VRP e suas variantes foram encontrados. A abordagem mais ingênua para solução do VRP são os métodos exatos de busca exaustiva. Embora esses métodos garantam encontrar a solução ótima, em virtude da complexidade do problema e tamanho do espaço de busca, a sua aplicabilidade fica restrita a problemas de tamanho reduzido, tendo em vista que o tempo de execução necessário para resolver instâncias de tamanho razoável é impraticável.

Portanto, sabendo que métodos exatos geralmente são inviáveis na prática, abordagens alternativas são necessárias. Nesse caso, as heurísticas e meta-heurísticas de aproximação se apresentam como uma boa alternativa de solução para esse tipo de problema. Esse tipo de técnica é enquadrada no âmbito do contexto de *soft computing*, que se caracterizam com o uso de mecanismos alternativos que permitem resolver problemas computacionais complexos, onde os cálculos são feitos de forma aproximada, mas que permitem obter uma solução aceitável em um tempo tolerável.

Diversos métodos heurísticos existem tanto para criação de soluções, quanto para melhoramento de soluções existentes. Contudo, heurísticas puras tendem a ser muito tendenciosas e carecem de robustez, frequentemente fazendo com que as melhorias da solução fiquem presas em um atrator local e impedindo a melhora da solução. Para resolver

esse problema, meta-heurísticas são vistas como uma alternativa, pois apresentam procedimentos mais robustos para este tipo de problema. As meta-heurísticas são procedimentos sofisticados que utilizam um conjunto de heurísticas para guiar a evolução da solução, visando obter o melhor resultado global possível.

De acordo com a revisão realizada, foi possível constatar que, de forma geral, meta-heurísticas populacionais nem sempre se apresentam como a melhor alternativa para o VRP, pois consomem mais tempo e recursos computacionais necessários para gerenciar várias soluções simultaneamente, sem apresentar ganhos significativos quando comparado a outras meta-heurísticas de solução única. Além disso, estudos mais recentes mostram que as meta-heurísticas adaptativas de melhoria de solução única têm apresentado melhor desempenho quando comparadas com meta-heurísticas populacionais, pois conseguem selecionar os melhores operadores de busca, se adaptando automaticamente para cada cenário, para obter um melhor resultado final (Belhaiza, 2018).

Por estes motivos, este trabalho propõe um método de solução para o VRPMTW baseado na meta-heurística ALNS, tendo em vista que este método permite alcançar soluções com boa qualidade, além de bom desempenho computacional frente a outras meta-heurísticas.

2.6 REVISÃO DE TRABALHOS RELACIONADOS

Nesta seção são apresentados os trabalhos relacionados ao estudo do problema de roteamento de veículos envolvendo múltiplas janelas de tempo (VRPMTW). Embora o estudo do VRP seja uma área muito ativa, poucos trabalhos encontrados na literatura propõem soluções para VRP com múltiplas janelas de tempo. Por esse motivo, esta revisão abrange não só trabalhos no âmbito do VRPMTW, mas também considera trabalhos que tratam o problema de múltiplas janelas de tempo em outros tipos de problemas de otimização combinatória. O intuito é avaliar se é possível adaptar tais soluções para a aplicação no problema em tela. A seguir são apresentados os trabalhos relacionados mais relevantes ao problema de pesquisa abordado neste trabalho.

2.6.1 Trabalhos relacionados

O trabalho de Bell et al. (1983) traz uma das primeiras menções ao problema de roteamento envolvendo múltiplas janelas de tempo. Os autores descrevem um problema real de logística para distribuição recorrente de gases industriais, em que uma das características elencadas era que nem todos os clientes estavam disponíveis para entrega em todos os dias da semana, nem todos as horas do dia. Dessa forma, existiam vários períodos específicos durante a semana em que os clientes podiam ser visitados. Esse trabalho também incluiu diversas outras características associadas a sua aplicação prática, tais como, controle de inventário, entregas parciais, predição de demanda, entre outras. Um ponto negativo

desse trabalho é que ele não tratou da formulação do problema com múltiplas janelas de tempo. Os autores propuseram a resolução do problema por meio do método de Relaxação Lagrangeana, um algoritmo exato de otimização com restrições. Ao aplicar a solução na prática, foi possível perceber uma redução de cerca de 6% a 10% de custos da operação logística.

O trabalho de Pesant et al. (1999) aborda o problema de múltiplas janelas de tempo aplicado ao Problema do Caixeiro Viajante (TSPTW). Em seu trabalho, os autores descrevem de forma matemática as restrições do problema. Para resolver o problema de satisfação de restrições e minimização do custo, foi utilizado um algoritmo *iterative-deepening depth-first search*, que é uma abordagem baseado no método de *branch-and-cut*. Como não haviam instâncias disponíveis na literatura para o problema até aquele momento, os autores realizaram adaptações em algumas instâncias do TSPTW de Solomon (1987). Eles fizeram a fragmentação das janelas de tempo originalmente existentes, de forma que o tempo de atendimento fosse o mesmo, mas múltiplas janelas de tempo poderiam ser oferecidas respeitando as restrições das janelas de tempo iniciais. Como resultado, foi possível obter o resultado ótimo conhecido para 55 das 63 instâncias de teste analisadas

Os trabalhos de Tricoire et al. (2010) e Tricoire et al. (2013) relatam a incorporação de múltiplas janelas de tempo ao *Team Orienteering Problem* (TOP). O TOP é um problema parecido ao VRP, porém, nele cada requisição possui uma pontuação diferenciada, sendo que o objetivo do problema é atender o máximo de clientes possíveis, maximizando a pontuação de atendimento, sem necessariamente atender a todos os serviços. Os autores aplicaram a solução a um problema real de vendedores que visava maximizar o lucro de vendas ao tentar visitar o máximo de potenciais clientes, cada um podendo ter um potencial de venda presumido distinto. Ele foi o primeiro trabalho a utilizar uma meta-heurística de solução única para a resolução do problema, sendo utilizado o algoritmo *Variable Neighborhood Search* (VNS). Por consequência, od autores relatam a implementação de diversos operadores construtivos e de vizinhança utilizados para executar a meta-heurística. Além disso, as múltiplas janelas de tempo são tratadas como um subproblema, sendo resolvidas com uso da técnica de cálculo do *Minimum Tour Duration*. Para validar os resultados, os autores utilizaram dados reais próprios e também dados de *benchmarks* adaptados de Rancourt, Cordeau e Laporte (2013) e Solomon (1987). Como resultado, os autores alegam que obtiveram desempenho superior a alternativas comerciais para instâncias pequenas, mas enfrentaram problemas com instâncias grandes devido à quantidade de cálculos e verificações das janelas de tempo.

Uma das principais contribuições do trabalho de Favaretto, Moretti e Pellegrini (2007) foi a realização da caracterização e formulação matemática do VRPMTW da forma com ele é utilizado até os dias de hoje. Além disso, os autores desenvolveram um algoritmo chamado MACS-VRPMTW, baseado em otimização por colônia de formigas, que possui dois sub-algoritmos de otimização: o ACS-TIME, que é utilizado para melhorar

a qualidade de rotas existentes, e o ACS-VEI, que é utilizado para reduzir o número de veículos utilizados. Para cada requisição, cada possível janela de tempo possui uma quantidade de feromônio para simbolizar a probabilidade dessa janela ser a selecionada para a rota final. Como os autores justificam que não haviam conjuntos de testes disponíveis para o problema, foram realizadas adaptações a *benchmarks* já existentes para problemas de roteamento de veículos que não possuíam restrições de janelas de tempo, mas que a solução ótima já era conhecida. Assim, ao incluir novas múltiplas janelas de tempo, contanto que exista uma janela de tempo que atenda a cada requisição no horário do *benchmark* original, a solução ótima não é alterada e o novo cenário pode ser utilizado para buscar pela solução ótima já conhecida.

O trabalho de Souffriau, Vansteenwegen e Berghe (2013) incorporou o problema de múltiplas janelas de tempo ao problema de *Orienteering Problem*. Neste problema, um turista deseja visitar o maior número de pontos turísticos de uma cidade, sendo que cada ponto turístico está aberto em diferentes períodos de tempo. Para resolver esse problema, os autores utilizam uma implementação do algoritmo híbrido de *Greedy randomized adaptive search procedure* (GRASP) e *Iterated Local Search* (ILS), visando balancear entre diversificação e intensificação, ao qual eles batizaram como *Greedy Randomized Iterated Local Search* (GRILS). Na implementação desse algoritmo é possível perceber que foi necessário implementar adaptações em diversos operadores da meta-heurística para torná-los aptos ao contexto de otimização de múltiplas janelas de tempo.

Belhaiza, Hansen e Laporte (2014) partiram da formalização do VRPMTW definida por Favaretto, Moretti e Pellegrini (2007) e realizaram a implementação de uma meta-heurística híbrida que utiliza uma combinação das meta-heurísticas *Variable Neighborhood Search* (VNS) e *Tabu Search* (TS). Para implementar o método proposto alguns operadores de diversificação e de melhoramento foram adaptados para resolver o problema de múltiplas janelas de tempo. Também foi usado o algoritmo proposto por Tricoire et al. (2010) para calcular eficientemente o *Minimum Forward Time Slack* e o *Minimum Backward Time Slack* e verificar a factibilidade das soluções. Para validação dos resultados, os *benchmarks* de Favaretto, Moretti e Pellegrini (2007) foram utilizados, além do desenvolvimento de novas instâncias.

Em Belhaiza e M'Hallah (2016) os autores incorporaram o método proposto por Belhaiza, Hansen e Laporte (2014), incorporando conceitos de *Game Theory*, em que diferentes rotas "competem" entre si para tentar melhorar seu desempenho até alcançar o *Nash Equilibrium*, onde as soluções se harmonizam e estabilizam. Em Belhaiza, M'Hallah e Ben Brahim (2017) o algoritmo VNS foi incrementado com alguns operadores e conceitos de algoritmos genéticos visando melhorar seus resultados.

Em Belhaiza (2018) também é utilizado VNS e *Tabu Search* para resolver o VRPMTW, mas com uma função objetivo multicritério utilizada para balancear a carga entre diferentes veículos, visando obter como resultado o conjunto Pareto de soluções.

Mais recentemente, Belhaiza et al. (2018) implementaram diversos outros operadores na meta-heurística se baseando nos conceitos apresentados em Belhaiza, Hansen e Laporte (2014). Embora sejam artigos do mesmo grupo de pesquisa e não tenham conceitos tão impactantes individualmente, no conjunto, eles apresentam um bom leque de heurísticas, meta-heurísticas e operadores a serem utilizadas para o VRPMTW, além de disponibilizar conjuntos de teste.

Em Hurkała (2015), o autor elenca o problema de múltiplas janelas de tempo como um subproblema dos problemas de TSP, VRP e OP. No contexto do TSP, o autor propôs modificações ao algoritmo proposto por Tricoire et al. (2010) para calcular de forma mais eficiente o *Minimum Forward Time Slack* e o *Minimum Backward Time Slack*, e verificar a factibilidade das soluções. As alterações permitiram adicionar características estocásticas, tais como, atualizar a solução existente quando o tempo de deslocamentos entre os nós é modificado (por motivos de congestionamento, por exemplo), ou o tempo de serviço de um nó é alterado. O autor propõe como solução um algoritmo que contempla várias etapas, incluindo VNS, SA, *List-Based Threshold acceptance*. Por incorporar características estocásticas e dinâmicas ao problema, para validar os resultados o autor criou seus próprios conjuntos de teste de forma randomizada.

Paulsen, Diedrich e Jansen (2015) propuseram uma solução para o problema de TSP com múltiplas janelas de tempo (TSPMTW) utilizando programação dinâmica. Os autores incorporaram os conceitos do *Minimum Tour Duration Problem* (MTDP) para resolver o problema de avaliação e alocação ótima das janelas de tempo. Eles projetaram como os conceitos de programação dinâmica poderia ser utilizados para abordar de forma eficiente o subproblema de múltiplas janelas de tempo dentro do TSP. Segundo os autores, essa técnica também pode ser utilizada para resolver outros problemas de otimização de roteamento que envolvem janelas de tempo, tais como o VRPMTW.

Larsen e Pacino (2019) propuseram a utilização de uma meta-heurística adaptativa, o *Adaptive Large Neighborhood Search* (ALNS), se baseando nos conhecimentos teóricos produzidos nos trabalhos de Belhaiza, Hansen e Laporte (2014) e Favaretto, Moretti e Pellegrini (2007), para abordar o VRPMTW. Os autores reconhecem que uma das grandes limitações de abordagens ao VRPMTW é justamente a questão de resolver dois problemas de otimização combinatória simultaneamente: o problema de agendamento das janelas de tempo e o problema de roteamento de veículos em si. Para tentar amenizar o impacto da combinação desses dois problemas, eles propuseram a utilização de Δ-*Evaluation* para checar a factibilidade de alterações dentro dos operadores da meta-heurística. Dessa forma, as consultas de factibilidade realizadas tendem a ser mais rápida, pois avaliam apenas as alterações feitas na rota, ao invés de avaliar a rota completamente. Com isso, a intenção dos autores era conseguir aumentar o desempenho do algoritmo por meio do aumento da eficiência computacional dos operadores, que tiveram sua complexidade assintótica reduzida.

Hoogeboom et al. (2020) abordam o VRPMTW utilizando operadores que suportam a avaliações incrementais (Δ-*Evaluation*) das soluções já existentes. Eles adaptaram o cálculo proposto por Tricoire et al. (2010) para lidar com múltiplas janelas de tempo para realizar avaliações incrementais de alterações nas soluções, tornando a avaliação mais rápida, conseguindo aumentar em 4 vezes a velocidade da execulção do algoritmo. Tais implementações são utilizadas como operadores para uma meta heurística de *Adaptive Variable Neighborhood Search* (AVNS). Para avaliar os resultados, foram utilizadas comparações com os resultados de Tricoire et al. (2013) e também instâncias de *benchmark* de Solomon adaptadas conforme o método proposto por Favaretto, Moretti e Pellegrini (2007).

2.6.2 Considerações sobre os trabalhos relacionados

A partir dos trabalhos encontrados durante a revisão bibliográfica, foi possível perceber que a utilização de Múltiplas Janelas de Tempo não é uma restrição exclusiva às aplicações de VRP. No decorrer da pesquisa foi possível encontrar diversas aplicações dessa restrição em outros problemas, tais como, TSP, OP e TOP. Embora esses problemas não sejam equivalentes, são problemas onde a maioria das adaptações tratam as múltiplas janelas de tempo como um subproblema. Portanto, guardando-se as devidas particularidades de cada aplicação, a mesma abordagem para tratar as múltiplas janelas de tempo como um subproblema pode ser utilizada independente do problema de otimização subjacente.

A Tabela 4 apresenta uma compilação dos principais aspectos relacionados a VRPMTW encontrados nos trabalhos revisados. A tabela elenca os trabalho sendo analisado (primeira coluna), bem como suas respectivas características de tipo problema sendo abordado (segunda coluna), variações do problema que são tratadas (terceira coluna), e método utilizado para solução (quarta coluna), bem como a referência das instâncias *benchmarks* utilizadas para avaliar o método. Analisando a tabela é possível perceber que as soluções para o VRPMTW mostram uma tendência na proposta de modelos que aplicam meta-heurísticas adaptativas de melhoramento de solução única, sendo utilizada a programação dinâmica, por meio do cálculo do Δ-*Evaluation*, na avaliação da factibilidade das janelas de tempo.

Também é possível notar uma expressiva carência de operadores para as meta-heurísticas de melhoramento adaptados ao uso do Δ-*Evaluation*. Ou seja, os trabalhos que utilizam esse tipo de solução adaptam um número mínimo possível de operadores necessários para a avaliação da sua proposta de solução, gerando assim lacunas de otimização que acabam não sendo atendidas. Isso pode ser explicado porque a adaptação de cada operador deve ser feita de forma personalizada para manter as particularidades de seu funcionamento, o que torna esse processo complexo e demorado. Ainda, foi constatado que os trabalhos que utilizam o método Δ-*Evaluation* o aplicam apenas na avaliação local das modificações realizadas pelos operadores da meta-heurística. Entretanto, esse método poderia ser melhor

utilizado se fosse também aplicado no cálculo completo da solução efetivamente modificada.

Portanto, por meio da revisão da literatura foi possível perceber que as soluções mais eficientes para o VRPMTW utilizam meta-heurísticas adaptativas de solução única, como o ALNS, o VNS e o AVNS, devido ao bom desempenho e qualidade nos resultados que essas meta-heurísticas proveem. A reduzida quantidade de operadores adaptados para o uso do Δ-*Evaluation* para o VRPMTW abre espaço para a proposição da adaptação de outros operadores que possam cobrir as lacunas de otimização existentes. Além disso, a utilização da abordagem Δ-*Evaluation* apenas na avaliação da factibilidade das modificações feitas na solução traz a oportunidade de melhorias de desempenho também ao aplicá-la na avaliação completa da solução. Portanto, essas duas melhorias mencionadas acima aplicadas a um algoritmo ALNS são os principais pontos de contribuição destacados pela solução proposto neste trabalho.

Tabela 4 – Trabalhos Relacionados

Trabalho	Problema	Variações	Método	Benchmark
(HURKAIA, 2015)	TSP	MTW	VNS/SA	Próprio – gerado aleatoriamente
(BELHAIZA; HANSEN; LAPORTE, 2014)	VRP	MTW	VNS+TS	Próprio – adaptação de instâncias de (SOLOMON, 1987), (FAVARETTO; MORETTI; PELLEGRINI, 2007)
(PESANT et al., 1999)	TSP	MTW	Branch-and-bound	Próprio – adaptação de instâncias RC2 de (SOLOMON, 1987)
(BELHAIZA et al., 2018)	VRP	MTW	VNS	(BELHAIZA; HANSEN; LAPORTE, 2014)
(PAULSEN; DIEDRICH; JANSEN, 2015)	TSP	MTW	DP	(PESANT et al., 1999)
(Belhaiza, 2018)	VRP	MTW	VNS+TS	Próprio – cenário real
(Belhaiza; M'Hallah, 2016)	VRP	MTW	VNS+TS	(BELHAIZA; HANSEN; LAPORTE, 2014)
(Belhaiza; M'Hallah; Ben Brahim, 2017)	VRP	MTW	VNS+GA	(BELHAIZA; HANSEN; LAPORTE, 2014)
(FAVARETTO; MORETTI; PELLEGRINI, 2007)	VRP	MTW	ACO	Próprio – adaptação de instâncias de (FISHER, 1994)
(HOOGEBOOM et al., 2020)	VRP	MTW	AVNS	(BELHAIZA; HANSEN; LAPORTE, 2014)
(TRICOIRE et al., 2010)	TOP	MTW	VNS	Próprio – cenário real, (SOLOMON, 1987), (RANCOURT; CORDEAU; LAPORTE, 2013)
(TRICOIRE et al., 2013)	TOP	MTW	VNS	(MONTEMANNI; GAMBARDELLA, 2009)
(SOUFFRIAU; VANSTEENWEGEN; BERGHE, 2013)	OP	MTW	GRASP+ILS	(SOLOMON, 1987), (RANCOURT; CORDEAU; LAPORTE, 2013)
(LARSEN; PACINO, 2019)	VRP	MTW	ALNS	(BELHAIZA; HANSEN; LAPORTE, 2014)
Trabalho em Tela	VRP	MTW	ALNS	(BELHAIZA; HANSEN; LAPORTE, 2014)

Fonte: Próprio autor

3 SOLUÇÃO PROPOSTA

Este trabalho tem por objetivo propor um método eficiente, ágil e robusto para solucionar o Problema de Roteamento de Veículos com Múltiplas Janelas de Tempo (VRPMTW), permitindo minimizar a quantidade de veículos, o tempo de operação e o custo total das viagens. Este capítulo detalha o método de solução proposto, começando pela apresentação de sua modelagem conceitual, passando pela representação do fluxo do processo de solução e detalhamento dos seus principais algoritmos, e finalizando com a apresentação de alguns aspectos técnicos de implementação.

Conforme descrito na Seção 2.4, o VRPMTW é na verdade a associação de dois problemas de otimização combinatória: 1º) o problema de roteamento de veículos e 2º) o problema de agendamento com múltiplas janelas de tempo. Geralmente, o segundo problema é tratado como um problema subordinado ao primeiro. Portanto, neste trabalho, o problema de agendamento e minimização de tempo de serviço é incorporado a um método para solução de VRP, de forma que ele se torne um subproblema de otimização embutido no processo de otimização do VRP.

Para a solução do VRPMTW, este trabalho assume que cada veículo é definido por uma localização do depósito, jornada de trabalho, distância máxima da rota, capacidade de carga e custo de utilização. Ainda, cada serviço é definido por sua localização, demanda, tempo de atendimento e janelas de tempo associadas. O método de otimização deve juntar todas essas informações e criar o melhor conjunto de rotas possível. Portanto, a solução para VRPMTW apresentada neste trabalho tenta encontrar a melhor combinação possível de rotas que consiga atender a todos os serviços, minimizando a quantidade e o tempo total de operação dos veículos.

Devido à complexidade dos VRP, meta-heurísticas de melhoramento de solução única se apresentam como uma abordagem de solução adequada. Conforme abordado na Seção 2.5, elas têm capacidade de obter bons resultados utilizando menos recursos computacionais que as abordagens populacionais. Entretanto, o VRPMTW tem uma complexidade a mais, que é a análise da melhor combinação das múltiplas janelas de tempo de cada serviço atendido. Esse processo precisa ser realizado a cada melhoramento da solução, o que significa executá-lo demasiadas vezes dentro da execução da meta-heurística, o que o torna um processo computacionalmente custoso. Portanto, para se obter um método de otimização robusto para o problema VRPMTW, é necessário utilizar uma abordagem eficiente de análise das múltiplas janelas de tempo.

Considerando esses requisitos, este trabalho propõe um método para a solução do VRPMTW a partir da meta-heurística *Adaptive Large Neighborhood Search* (ALNS), com a aplicação de técnicas de programação dinâmica para realizar a avaliação incremental (Δ-*Evaluation*) da factibilidade dos agendamentos e minimização do tempo de viagem. A meta-heurística ALNS foi escolhida devido à sua capacidade de geração de bons resultados

e eficiência computacional. Essa meta-heurística realiza sucessivas modificações na solução corrente por meio da aplicação de operadores selecionados de forma adaptativa. Esses operadores modificam a solução visando guiá-la para o mínimo global apontado pela função objetivo. A utilização de programação dinâmica para realizar a avaliação incremental (Δ-*Evaluation*) foi selecionada devido a sua eficiência na redução do esforço necessário para computar os tempos dos serviços a partir das alterações feitas pela meta-heurística à solução.

3.1 MÉTODO PROPOSTO

A formulação matemática do problema abordado neste trabalho é semelhante à apresentada na Seção 2.4. Entretanto, neste trabalho são feitas algumas simplificações, tais como a utilização de frota homogênea. Portanto, todo veículo possui o mesmo valor para Q_k, o que significa a mesma capacidade. Ainda, todo veículo possui D_k com valor infinito, fazendo com que o tamanho máximo da rota não seja limitado. Além disso, o parâmetro de diminuição da frota está sempre ativado, ou seja, $B = 1$.

A função objetivo utilizada na avaliação de uma solução tem por objetivo minimizar a quantidade de veículos e o custo total de suas rotas, sendo que a minimização do número de veículos tem prioridade em relação a minimização do custo das rotas. Entretanto, estes dois objetivos podem ser conflitantes, haja vista que em alguns casos uma solução com maior número de veículos tem um custo total das rotas menor do que uma solução com menos veículos (SCHMITT; BALDO; PARPINELLI, 2018). Devido a esse fato e sabendo que a minimização do número de veículos é hierarquicamente mais prioritária do que o custo total das rotas, o modelo proposto neste trabalho é composto por dois algoritmos ALNS que minimizam simultaneamente duas soluções distintas, sendo um utilizado para minimizar a quantidade de veículos e o outro para minimizar o custo total das rotas. Assim, para cada iteração do ciclo de otimização do método proposto são realizadas otimizações nas duas soluções, que são modificadas e avaliadas de acordo com seus respectivos objetivos com aceitação de forma hierárquica, sempre priorizando a redução do número de veículos.

A minimização do número de veículos é feita a partir de uma solução infactível que contém um veículo a menos que a melhor solução factível encontrada. No momento em que essa solução se torna factível, ela é repassada para a minimização do custo total. Como as penalidades sobre uma solução infactível são proporcionais as violações contida nela, o ALNS responsável pela minimização do número de veículo tenta reduzir o número de violações e gerar uma solução factível com um número menor de veículos, que é posteriormente otimizada pelo ALNS minimizador de custo. Por outro lado, a solução utilizada pelo ALNS de minimização de custo é sempre uma solução factível. Portanto, na minimização de seu custo todas as modificações passam por verificações de aceitação que devem necessariamente descartar a solução quando qualquer infactibilidade por detectada.

Com repeito a avaliação incremental das janelas de tempo, quando da criação da solução inicial do problema todos os tempos envolvidos com as janelas de atendimento são calculados e têm sua factibilidade testada. A partir desse ponto, a cada modificação da solução realizada pelos ALNSs apenas serão recalculados os tempos e testadas as factibilidades dos serviços que foram afetados pela modificação. Portanto, a utilização do Δ-*Evaluation* evita que cálculos sobre tempos de serviços não afetados pelas modificações sejam realizados. Os cálculos do Δ-*Evaluation* são realizados tanto pelo ALNS que minimiza veículos quanto pelo ALNS que minimiza o custo total. Entretanto, apenas o ALNS que minimiza veículos aceita soluções com tempos de serviços infactíveis.

A Figura 7 apresenta o fluxo do processo de otimização do método proposto, detalhando o uso do ALNS para a solução do VRPMTW com avaliação incremental (Δ-*Evaluation*) das Múltiplas Janelas de Tempo utilizando Programação Dinâmica.

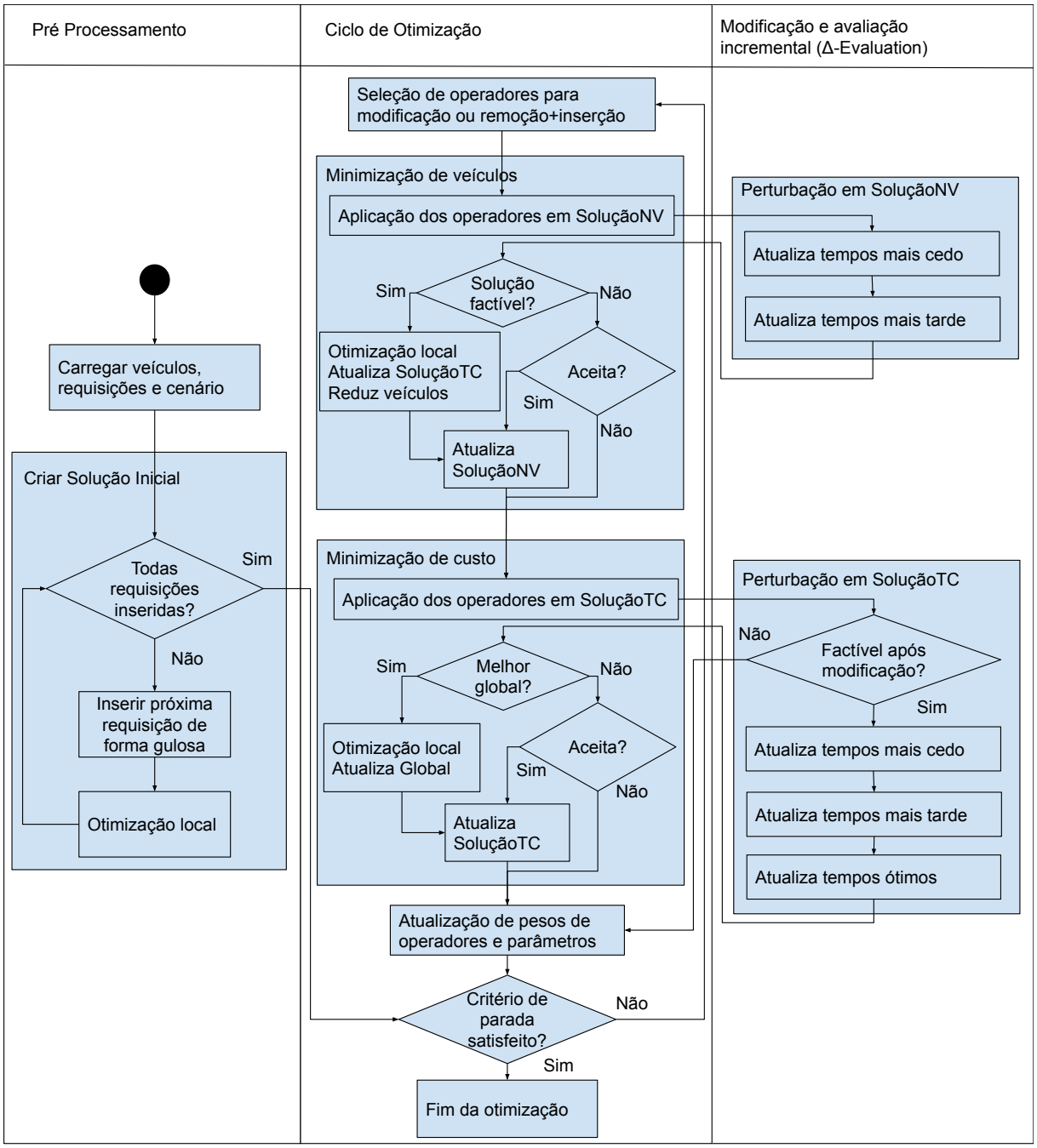
A primeira etapa do processo representado pela Figura 7 começa pelo carregamento e interpretação das informações do problema. Isso inclui todos os serviços e suas restrições, informações de veículos, tais como jornada e restrições de capacidade, informações do ambiente, tais como distância, custo e tempo de deslocamento entre serviços, e entre serviços e depósito. É importante destacar que o método de otimização proposto não inclui um mecanismo de cálculo de distância entre os nós dos serviços. Portanto, é necessário um pré-processamento externo para realizar os cálculos de distância e tempo de deslocamento entre os serviços. Essa decisão foi tomada pois não cabe ao método de otimização lidar com condições específicas de cada cenário de aplicação, tais como localização geográfica, cálculo de distâncias em malha viária ou euclidiana, condições de tráfego e etc.

Em seguida, ainda no pré-processamento, uma vez que as informações foram carregadas é realizada a geração da solução inicial por meio de um operador de construção heurística. Essa etapa é necessária devido ao fato de que o ALNS é uma meta-heurística de melhoramento, ou seja, seu objeto é melhorar uma solução existente, portanto, ele não é capaz de criar uma solução do zero.

O processo de construção da solução inicial opera intercaladamente entre um procedimento de inserção, e procedimento de melhoramento local, até que todos os serviços estejam alocados em rotas. Para o método proposto, é necessário que a solução inicial seja factível, ou seja, todas as restrições do problema sejam satisfeitas. Portanto, é fundamental que a heurística de criação da solução inicial consiga cumprir essa tarefa, caso contrário, mesmo que uma solução factível exista, a meta-heurística de melhoramento não poderá ser executada. É desejável que a solução inicial seja robusta, pois irá servir de base para ser aprimorada pelo pela meta-heurística ALNS embutida no método proposto.

De posse da solução inicial factível, antes de iniciar o ciclo de otimização, são primeiramente inicializadas as variáveis necessárias, tais como temperatura, pesos iniciais dos operadores, critério de paradas, entre outros. A solução inicial previamente calculada é utilizada como ponto de partida para o ALNS minimizador de custo, enquanto que o

Figura 7 – Método proposto para resolução do VRPMTW com ALNS



Fonte: Imagem gerada Próprio autor

ponto de partida para o ALNS minimizador de veículos é uma solução infactível calculada a partir da remoção de um veículo aleatório da solução inicial. Finalizada essa etapa então se inicia o ciclo de otimização propriamente dito, com as seguintes atividades:

- **Critério de parada**: o método proposto apresenta 3 critérios de encerramento: i) finalização por tempo, ii) finalização por número de iterações e iii) finalização por estagnação. Cada um deles pode ser ativado ou desativado de forma independente e quando qualquer um deles estiver ativo e for satisfeito o algoritmo é encerrado. O

critério de finalização por tempo de execução é contabilizado pela quantidade em milissegundos de execução decorridos a partir do início da otimização, ao atingir o tempo de execução determinado o ciclo de otimização é encerrado. O critério de finalização por número de iterações é contabilizado pela quantidade de iterações realizadas pela meta-heurística. Ao atingir um determinado número de iterações a execução da otimização é finalizada. O critério de finalização por estagnação é contabilizado pelo número de iterações sem que seja observado melhoramento na melhor solução global, tanto pelo número de veículo quanto pelo custo total da função objetivo. Entretanto, sempre que uma solução melhor global é encontrada o contador de iterações restantes é restaurado.

- **Seleção de Operadores**: Existem três tipos distintos de operadores propostos no modelo: operadores de modificação (apenas alteram uma solução, sem retirar qualquer serviço da solução), operadores de remoção (realizam a remoção de serviços da solução) e operadores de inserção (realizam a inserção de eventuais serviços deixados de fora da solução). Primeiramente, um operador é selecionado entre o conjunto de operadores de modificação e operadores de remoção por meio de um algoritmo de seleção probabilístico baseado na pontuação acumulada de cada operador. Caso um operador de remoção seja selecionado, então é realizada mais uma seleção probabilística para a seleção de um operador de inserção. No caso deles serem de remoção e inserção, primeiramente é aplicado um operador de remoção que remove um número n de serviços para desconstruir parte da(s) rota(s) contidas na solução e, em seguida, um operador de inserção é aplicado para reinserir os serviços em outra(s) rota(s). Portanto, os operadores selecionados podem ser um simples operador de modificação de solução, ou pode ser um par de operadores de remoção e inserção associados.

- **Minimização do Número de Veículos**: Nesta etapa os operadores selecionados anteriormente são aplicados e o cálculo dos tempos da nova solução é feito simultaneamente por meio das operações de avaliação incremental Δ-*Evaluation*. Portanto, toda a vez que a solução é modificada os tempos associados a ela também são recalculados para todos os serviços que foram afetados. Ao término da aplicação dos operadores uma nova solução é criada e existem três possíveis cenários distintos a serem avaliados: i) Caso essa solução seja factível, isto é, a nova solução gerada respeita todas as restrições do problema, ela passa por um processo de melhoramento, por meio da aplicação de operadores de otimização local, e substitui tando a melhor solução global quanto a solução utilizada no minimizador de custo. Por fim, é feita a remoção de um veículo aleatório e ela passa a ser a nova solução do minimizador de veículos. ii) Caso a nova solução, mesmo infactível, seja melhor do que a solução atual, pois conseguiu diminuir o número e/ou a intensidade das

restrições infringidas, ela é aceita como nova solução atual e substitui a solução anterior do minimizador de veículos. iii) Caso a nova solução seja infactível e pior do que a solução atual, então a função de aceitação da meta-heurística decide se a nova solução irá substituir a solução atual com base no quanto ela é prior e na temperatura atual do ALNS.

- **Minimização do Custo Total**: Assim como na etapa anterior, nesta etapa os operadores selecionados anteriormente são aplicados e o cálculo dos tempos da nova solução é feito simultaneamente por meio das operações de avaliação incremental Δ-*Evaluation*. Portanto, toda a vez que a solução é modificada os tempos associados a ela também são recalculados para todos os serviços que foram afetados e a sua factibilidade e avaliada. Ao término da aplicação dos operadores uma nova solução é criada e existem três possíveis cenários distintos a serem avaliados: i) Caso ela seja um novo ótimo global, isto é, a nova solução gerada é uma solução melhor do que a solução global atual, essa nova solução passa por um processo de aprimoramento pela aplicação de operadores de otimização local até que a solução se estabilize. Ao final, ela é aceita como nova solução e substitui tanto a solução atual quanto a solução global. ii) Caso a nova solução seja melhor do que a solução atual, mas não melhor que a global, ela é aceita como nova solução atual e substitui a solução anterior. iii) Caso a nova solução seja pior do que a solução atual, então a função de aceitação da meta-heurística decide se a nova solução irá substituir a solução atual com base no quanto ela é prior e na temperatura atual do ALNS. Dessa forma, a busca inicia de forma abrangente no início e conforme a função de temperatura tem seu valor diminuído o algoritmo acaba mudando seu comportamento e intensificando as buscas em vizinhanças mais próximas.

- **Atualização dos pesos dos operadores e outros parâmetros**: Ao final de cada passagem pelo ciclo de otimização alguns parâmetros do algoritmo são ajustados. Por exemplo, a temperatura do sistema é reaquecida à temperatura inicial caso um novo melhor global com número menor de veículos seja encontrado, caso contrário, a temperatura do sistema é diminuída de acordo com a função de resfriamento definida. As variáveis utilizadas na verificação do critério de parada são atualizadas, sendo elas a solução corrente, o número de iterações e o tempo de processamento. Por fim, os pesos dos operadores do ALNS são atualizados com base na pontuação atribuída caso a solução atual seja aceita.

No que tange ao tratamento das múltiplas janelas de tempo, sabendo que elas são fixas e que a ordem de atendimento dos serviços é definida pelo método de otimização, o problema em questão é determinar de forma ótima quais janelas de tempo serão utilizadas e qual o instante que elas serão utilizadas para cada serviço, visando minimizar o tempo total de operação do veículo (considerando os tempos de deslocamento e os tempos de

espera). Portanto, o intuito dessa análise é diminuir o máximo possível os tempos de folga entre o *earliest arrival* e *latest departure*.

Todos os operadores de criação e de melhoramento utilizados no ALNS tiveram que ser adaptados para realizar a avaliação incremental das janelas de atendimento utilizando o Δ-*Evaluation*. Portanto, com essa adaptação todos os operadores se tornaram capazes de interpretar as informações das múltiplas janelas de tempo e se tornaram capazes de atualizar essas informações de forma a mantê-las condizentes e consistentes com as alterações realizadas na solução.

Para permitir que os operadores do ALNS possam realizar as checagens nas modificações nas soluções de forma incremental utilizando o Δ-*Evaluation*, um conjunto de operações básicas foi definido. Assim, os operadores podem ser adaptados e implementados para tratar do Δ-*Evaluation* tomando como base essas operações básicas. Essas operações são elencadas e detalhadas a seguir:

- **Remoção Unitária**: Remove um serviço de uma posição específica da rota. O serviço deixa de fazer parte da solução corrente.
- **Inserção Unitária**: Insere um serviço em uma posição específica da rota. O serviço passa a fazer parte da solução corrente.
- **Troca (*Swap*/*Exchange*)**: Realiza a troca de posições entre dois serviços dentro da rota.
- **Movimento (*Move*)**: Realiza a movimentação de posição de um serviço específico dentro da rota.
- **Remoção de sequência**: Remove uma sequência de tamanho específico de serviços a partir de uma posição específica da rota. Os serviços removidos deixam de fazer parte da solução corrente.
- **Inserção de sequência**: Insere uma sequência de serviços em uma posição específica da rota. Os serviços passam a fazer parte da solução corrente.

As duas operações fundamentais são a de remoção unitária e a de inserção unitária. Todas as outras operações elencadas acima podem ser derivadas a partir dessas duas, porém, por questões de desempenho, elas foram especializadas para permitir alcançar o resultado de forma mais rápida e eficiente. Para cada tipo de operação existem duas possíveis ações a serem feitas: checagem de factibilidade e modificação da solução.

A checagem de factibilidade é uma operação simples, pois não realiza a modificação da solução. Ela apenas verifica se é possível aplicar a operação desejada sem violar a factibilidade da solução. Devido a possível grande quantidade de restrições e implicações advindas das múltiplas janelas de tempo, é relativamente frequente realizar operações que

tornam a rota infactível. Portanto, é vantajoso realizar essa verificação antes de efetivar a alteração da solução.

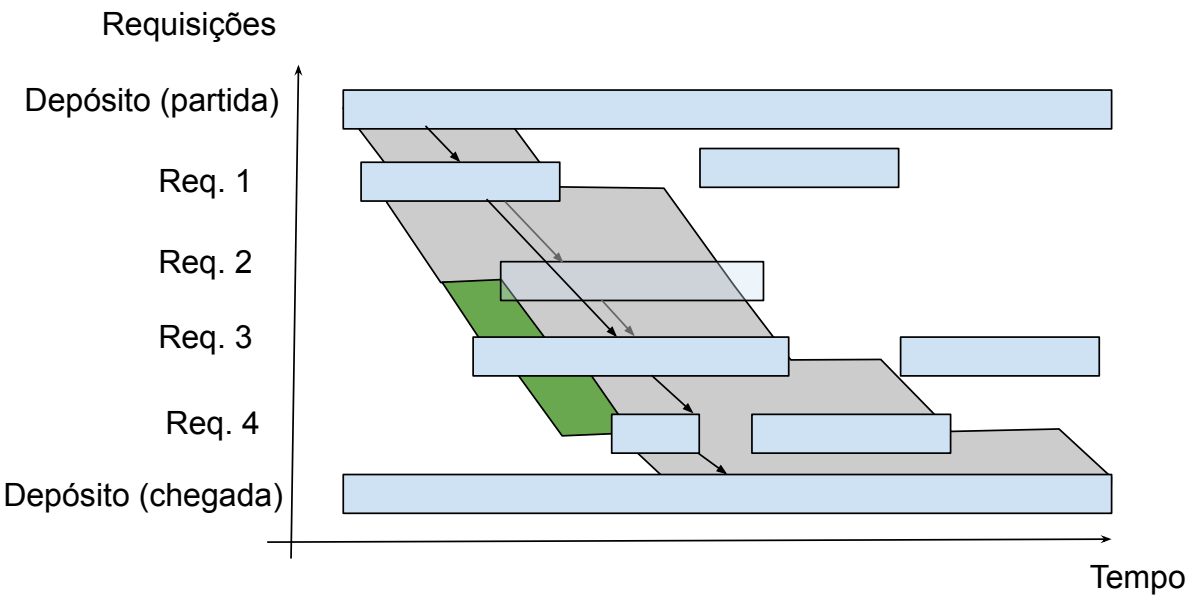
A ação de modificação da solução realiza efetivamente uma alteração na solução, criando assim uma nova solução. Esse tipo de operação é mais custosa do que a checagem de factibilidade, tendo em vista que ela recalcula os tempos dos atendimentos afetados pelas modificações. Entretanto, ela permite avaliar a qualidade da solução de forma bem mais rápida e eficiente do que com o recálculo completo da solução. Por exemplo, após uma ação de inserção de um atendimento os tempos de chegada mais cedo dos atendimentos subsequentes afetados precisam ser recalculados, assim como tempos de partida mais tarde dos atendimentos posteriores. Esses recálculos de tempo são feitos apenas para os horários dos atendimentos afetados. Portanto, para os atendimentos em que for identificado que os horários recálculo não diferem dos originais, não é necessário calcular o restante dos agendamentos, tendo em vista que o tempo estabilizou e mais nenhum recálculo modificaria os tempos já calculados. Esse processo de recálculo parcial da solução implica em um ganho significativo de desempenho se comparado com a necessidade de recalcular todos os tempos da solução a cada modificação, pois geralmente essas modificações na rota afetam apenas localmente o estado de uma solução.

A Figura 8 apresenta um cenário exemplo do modelo proposto de atualização incremental de uma solução. Nele é possível averiguar o impacto que a Requisição 2 exerce na organização das janelas de tempo da rota como um todo. Dois cenários sobrepostos são mostrados na imagem: a área cinza representa a região de factibilidade da rota com a Requisição 2 inclusa, enquanto a união da área cinza com a área verde mostra a região de factibilidade sem a Requisição 2. Ou seja, a região impactada pelas operações de adição (caso não esteja presente) ou remoção (caso esteja presente) é representada pela área verde do gráfico, sendo que a presença da Requisição 2 torna a região verde infactível.

Considerando uma operação de inserção, em que se deseja verificar a factibilidade da rota ao inserir a Requisição 2, essa checagem acaba sendo facilitada com as informações de tempo da rota já pré-calculadas, tendo em vista que é necessário simplesmente verificar se alguma das janelas de tempo da requisição (subtraindo os tempos de deslocamento entre requisições vizinhas) que se deseja inserir intersecta a região factível da solução. Caso não exista uma interseção, então não será possível criar uma rota factível. Caso contrário, se existir uma interseção entre as áreas de factibilidade e janela de tempo, uma solução factível poderá ser gerada.

No cenário representado na Figura 8 é possível perceber o potencial de ganho de desempenho da avaliação incremental das janelas de tempo, pois ao invés de recalcular todas as janelas de tempo da rota, o que acaba sendo custoso para rotas grandes, é necessário atualizar apenas os tempos de algumas requisições. No cenário da Figura 8, o início da janela de tempo da Requisição 2 diminui a área factível da rota, fazendo com que a chegada mais cedo das requisições 3 e 4 subsequentes (representada pela área verde no

Figura 8 – Modelo proposto cálculo incremental das múltiplas janelas de tempo



Fonte: Próprio autor

gráfico) sejam recalculados. Ao mesmo tempo, o final da janela de tempo da Requisição 2 se encontra fora da área factível da rota. Portanto, a Requisição 2 não influencia na área factível das requisições anteriores a ela. Dessa forma, não será necessário recalcular os tempos de saída mais tarde das requisições atendidas antes da Requisição 2. Entretanto, caso ela reduzisse a área factível, então os tempos de saída mais tarde das requisições anteriores à Requisição 2 precisariam ser recalculados.

3.2 MODELAGEM DO ALGORITMO

A seção anterior descreveu de forma geral o funcionamento do método proposto. Nesta seção são detalhados alguns aspectos e características técnicas do modelo proposto.

3.2.1 *Adaptive Large Neighborhood Search* (ALNS)

O método de solução proposto, baseada na meta-heurística ALNS, é representado no Algoritmo 1 e seus aspectos de funcionamento são detalhados a seguir.

O primeiro passo para a execução do ALNS é a criação de uma solução inicial, feita pela chamada do procedimento heurístico de criação de solução inicial (linha 1). Esse procedimento construtivo cria uma solução S, inicialmente vazia, e de forma semi-gulosa insere cada um dos serviços disponíveis por meio da heurística de inserção *Regret*, que realiza a inserção dos serviços considerado o custo da melhor inserção e comparado com o impacto nas demais opções de inserção futura. Cada inserção é seguida por um procedimento de intensificação com a aplicação dos operadores de busca local *best-relocate*,

Algorithm 1 Algoritmo ALNS

Require: I: Instância do problema

Crie uma solução inicial S para a instância I.
$Stc \leftarrow Stc_{best} \leftarrow S$
$Snv \leftarrow reduzVeiculo(S)$
Inicialize os pesos dos operadores q_i
while Critério de parada não satisfeito **do** ▷ Loop de orimização
▷ Seleção de operadores
    $q_a \leftarrow$ Selecione operador de modificação ou remoção com base nos pesos
    $q_b \leftarrow$ Seleciona operador de inserção com base nos pesos
▷ Minimiza veículos
    $Snv' \leftarrow$ cópia da solução Snv
    if q_a é operador de modificação **then**
        Modifique a solução Snv' com o operador q_a
    else
        Remove requisições de Snv' utilizando o operador q_a
        Insira requisições removidas por q_a em Snv' utilizando o operador q_b
    end if
    if $feasible(Snv')$ **then**
        Aplicar busca local em Snv'
        $S_{best} \leftarrow Stc \leftarrow Snv'$
        $Snv \leftarrow reduzVeiculo(Snv')$
    else if $cost(Snv') < cost(Snv)$ **then**
        $Snv \leftarrow Snv'$
    else if $accept(Snv')$ **then**
        $Snv \leftarrow Snv'$
    end if
▷ Minimiza custo
    $Stc' \leftarrow$ cópia da solução Stc
    if q_a é operador de modificação **then**
        Modifique a solução Stc' com o operador q_a
    else
        Remove requisições de Stc' utilizando o operador q_a
        Insira requisições removidas q_a por em Stc' utilizando o operador q_b
    end if
    if $cost(Stc') < cost(Stc_{best})$ **then**
        Aplicar busca local em Stc'
        $Stc_{best} \leftarrow Stc \leftarrow Stc'$
        Incremente as pontuações de q_a e q_b com θ_1
    else if $cost(Stc') < cost(Stc)$ **then**
        $Stc \leftarrow Stc'$
        Incremente as pontuações de q_a e q_b com θ_2
    else if $accept(Stc')$ **then**
        $Stc \leftarrow Stc'$
        Incremente as pontuações de q_a e q_b com θ_3
    end if
▷ Atualização de parâmetros
    Atualize os pesos q_i dos operadores.
    Realiza controle de temperatura dos sistema
end while
return Stc_{best} ▷ Retorna a melhor solução encontrada

best-exchange e *2-OPT*, cujo o objetivo é melhorar iterativamente a solução existente até que ela se estabilize. O detalhamento do funcionamento desses operadores pode ser encontrado na Seção 3.2.4.

Com uma solução inicial do problema construída, as variáveis necessárias para a execução do ALNS também são iniciadas, tais como a temperatura do sistema, a configuração do critério de parada, os pesos dos operadores distribuídos uniformemente, além da melhor soluções global factível (Stc_{best}, linha 2), da solução utilizada para o minimizador de custo (Stc, linha 2) e da solução utilizada pelo minimizador de veículos (Snv, linha 3), da qual é realizada a remoção de um de seus veículos.

Em seguida, o laço de repetição principal do algoritmo é inciado na linha 5. A cada iteração é realizada a checagem do critério de parada, conforme detalhado na Seção 3.1, em que é verificado o tempo de execução, quantidade de iterações e estagnação da solução. Quando qualquer uma dessas condições for satisfeita, o laço de repetição é interrompido.

Ao início de cada laço de iteração é feita a seleção dos operadores, de forma proporcional a sua respectiva pontuação, realizada entre as linhas 6 e 8. O operador q_a é selecionado dentro do conjunto de operadores de modificação ou remoção. Caso o operador q_a seja um operador de remoção, outro operador q_b de inserção é selecionado e ambos os operadores serão utilizados sequencialmente para modificar as soluções (destruindo e reconstruindo elas), caso contrário, apenas o operador de modificação q_a é suficiente para realizar a perturbação.

Em seguida, entre as linhas 10 e 25 é realizada a minimização do número de veículos. Primeiramente, é criada uma cópia Snv' da solução Snv (linha 10), que é perturbada pelos operadores q_a e q_b (linhas 11 a 16) selecionados anteriormente. Caso a nova solução gerada Snv' seja factível (linha 17), ela é aceita como a melhor por possuir um veículo a menos. Portanto, é feito um processo de intensificação nessa nova solução por meio da aplicação de busca local (linha 18). Na sequência, ela substituirá a melhor solução global Stc_{best} e a solução de minimização de custo Stc (linha 19). Ainda, uma nova solução Snv é calculada com a redução de mais um veículo da solução Snv' (linha 20). Caso a nova solução Snv' não seja factível, mas possua um custo inferior a Snv (linha 21), por reduzir a quantidade e intensidade das violações das restrições, a solução Snv é substituída pela nova solução Snv' (linha 22). Caso nenhum dos critérios anteriores seja satisfeito, ou seja, Snv' é infactível e pior do que Snv, então uma função de aceitação probabilística determina se a nova solução deve substituir Snv com base na diferença de avaliação entre elas e na temperatura do ALNS (linhas 23 e 24).

Em seguida, a minimização do custo total da solução é realizada entre as linhas 27 e 44. Primeiramente, é criada uma cópia Stc' da solução Stc (linha 27), que é perturbada pelos operadores q_a e q_b (linhas 28 a 33) selecionados anteriormente. Caso a nova solução gerada Stc' seja melhor que a melhor solução conhecida Stc_{best} (linha 34), é feito um processo de intensificação nessa nova solução por meio da aplicação do procedimento de

busca local (linha 35). Posteriormente, é feita a substituição da melhor solução global Stc_{best} e da solução de minimização de custo Stc pela melhor solução Stc' (linha 36), e a pontuação dos operadores utilizados é incrementada com θ_1 (linha 37). Caso contrário, se a solução Stc' possuir um custo inferior a Stc (linha 38), a solução Stc é substituída pela nova solução Stc' (linha 39) e a pontuação dos operadores utilizados é incrementada com θ_2 (linha 40). Caso nenhum dos critérios anteriores seja satisfeito, ou seja, Stc' é pior do que Stc_{best} e Stc, porém ela é factível, então uma função de aceitação probabilística determina se a nova solução deve substituir Stc com base na diferença de avaliação entre elas e também na temperatura do ALNS (linhas 41 e 42), e a pontuação dos operadores utilizados é incrementada com θ_3 (linha 43).

Após a avaliação da nova solução, as variáveis do ALNS são atualizadas, o que inclui a atualização periódica dos pesos dos operadores com base em suas pontuações, o controle de temperatura do ALNS, a coleta de estatísticas, entre outros (linha 45 a 47).

Ao final do laço de otimização, o melhor resultado encontrado para Stc_{best} é retornado como a melhor solução (linha 49).

3.2.2 Avaliação incremental

O método proposto permite a utilização de diversos operados durante a sua execução. Essa diversidade de operadores agrega flexibilidade ao método, permitindo que características distintas do problema possam ser melhor tratadas, tendo em vista que diferentes operadores podem lidar melhor com diferentes cenários e situações encontradas.

Para otimizar o cálculo do Δ-*Evaluation* pelas operadores do ALNS, um conjunto de operações básicas que incorporam esse cálculo foram implementadas, funcionando como base para as adaptações às implementações dos operadores da meta-heurística para torná-los aptos a lidarem com o cálculo das múltiplas janelas de tempo.

Com a utilização da abordagem Δ-*Evaluation*, os tempos de atendimento dos serviços são incorporados à otimização da solução. Portanto, para cada operação de alteração de uma solução é necessário lidar com os impactos dessas modificações nos tempos de atendimento. Isto é, uma vez que uma modificação é realizada na rota, é necessário recalcular os valores das janelas de tempo dos serviços afetados para manter concisas as informações da solução. Para isso, foram implementados procedimentos representados pelos Algoritmos 2, 3 e 4, que realizam esse processo de validação/invalidação das solução e ajustes nos valores associados às janelas de tempo de forma incremental.

O Algoritmo 2 e o Algoritmo 3 são utilizados para calcular a área factível da solução incorporando conceitos de programação dinâmica para recalcular apenas as modificações necessárias. Como pode-se perceber, os dois são semelhantes. Entretanto, enquanto o Algoritmo 2 calcula os tempos mais cedo associados às janelas de tempo dos serviços, o Algoritmo 3 calcula os tempos mais tarde associados às janelas de tempo dos serviços. Esses algoritmos são compostos por um laço de repetição iniciando a partir do serviço i

da rota S até o extremo da rota (início da rota para o caso dos tempos mais cedo e final da rota para o caso dos tempos mais tarde).

Embora a abrangência do recálculo dos tempos esteja limitado aos extremos de início e de fim da rota, espera-se que na prática o número de iterações necessárias seja significativamente menor. Isso porque os valores associados às janelas de tempo são calculados sob demanda pela programação dinâmica. Portanto, uma vez que dentro do recálculo iterativo da janelas os valores calculados após a modificação da solução sejam iguais aos valores anteriores à modificação da solução, não é necessário continuar recalculando os valores das janelas dos outros serviços da sequência. Ou seja, as modificações necessárias para atualizar as informações de tempo da solução, isto é, o Δ em $\Delta - Evaluation$, são calculados dinamicamente no momento em que os valores de tempo são atualizados, encerrando quando o algoritmo identifica que a solução se estabilizou.

Portanto, o Algoritmo 2 e o Algoritmo 3 apresentam a seguinte lógica de execução. Eles iniciam, na linha 3, com a criação de uma cópia das informações originais das janelas de tempo de S_{i+1} e S_{i-1}, respectivamente. Na sequência, o laço entre as linhas 4 e 9 realiza a invalidação dos valores associados às janelas de tempo do serviço atual. Em seguida, entre as linhas 10 e 17 é realizada a combinação das janelas de tempo do serviço vizinho, serviço anterior no caso do tempo mais cedo (Algoritmo 2) e serviço posterior para o caso do tempo mais tarde (Algoritmo 3), com as janelas de tempo do serviço atual, verificando as possibilidades de tempo a serem utilizados. Na linha 18 é verificado se os valores associados às janelas de tempo foram modificados. Caso isso tenha ocorrido, o algoritmo volta para a linha 2 para calcular as janelas de tempo do próximo serviço ou do serviço anterior nos Algoritmos 2 e 3, respectivamente. Caso não hajam modificações, o algoritmo encerra retornando o índice i da solução do último serviço que teve a janela de tempo recalculada (linha 19).

A programação dinâmica é usada de forma que uma vez que os tempos para as janelas de tempo de um serviço sejam calculados, eles permanecem disponíveis para os sucessivos cálculos dos tempos dos serviços vizinhos, evitando o recálculo, inclusive no caso de modificação posterior da rota.

Além do cálculo dos tempos que compreendem a área factível da solução (Algoritmo 2 e Algoritmo 3), também é necessário recalcular o valor de tempo ótimo para atendimento de cada serviço. Isto é feito pelo pseudocódigo apresentado no Algoritmo 4. A estrutura desse algoritmo é similar à estrutura dos Algoritmos 2 e 3, porém ele calcula informações diferentes. O algoritmo inicia a atualização dos valores de tempo ótimo a partir da primeira alteração realizada (considerando a ordem dos serviços) (linha 5), seguindo em direção ao final da rota enquanto o tempo ótimo de atendimento dos serviços for modificado. Assim como nos algoritmos visto anteriormente, o procedimento inicia invalidando os valores antigos e subsequentemente calculando iterativamente (linhas 7 a 11) até que o valor do $bestTime$ estabilize (linha 12).

Algorithm 2 Atualização de Tempos Mais Cedo

Require: S: solução existente, i: índice a iniciar o recálculo

function updateEarliestTime(S,i)
  while $i+1 < |S|$ **do**
    $twb \leftarrow TW(S_{i+1})$
    for each $j \in TW(S_{i+1})$ **do**
      $earliestArrival(j) \leftarrow \infty$
      $earliestServiceStartTime(j) \leftarrow \infty$
      $earliestServiceEndTime(j) \leftarrow \infty$
      $earliestDeparture(j) \leftarrow \infty$
    end for
    for each $i \in TW(S_i)$ **do**
      for each $j \in TW(S_{i+1})$ **do**
        $earliestArrival(j) \leftarrow Max(earliestArrival(j), earliestDeparture(i) + cost(S_i, S_{i+1}))$
        $earliestServiceStartTime(j) \leftarrow Max(earliestArrival(j), lower(j))$
        $earliestServiceEndTime(j) \leftarrow earliestServiceStartTime(j) + serviceTime(S_{i+1})$
        $earliestDeparture(j) \leftarrow earliestServiceEndTime(j)$
      end for
    end for
    if $twb = TW(S_{i+1})$ **then**
      return i
    end if
    $i \leftarrow i+1$
  end while
end function

Algorithm 3 Atualização de Tempos Mais Tarde

Require: S: solução existente, i: índice a iniciar o recálculo

function updateLatestTime(S,i)
  while $i > 0$ **do**
    $twb \leftarrow$ cópia de $TW(S_{i-1})$
    for each $j \in TW(S_{i-1})$ **do**
      $latestArrival(j) \leftarrow -\infty$
      $latestServiceStartTime(j) \leftarrow -\infty$
      $latestServiceEndTime(j) \leftarrow -\infty$
      $latestDeparture(j) \leftarrow -\infty$
    end for
    for each $i \in TW(S_i)$ **do**
      for each $j \in TW(S_{i-1})$ **do**
        $latestDeparture(j) \leftarrow Max(latestDeparture(j), latestDeparture(i) - cost(S_{i-1}, S_i))$
        $latestServiceEndTime(j) \leftarrow Min(latestDeparture(j), upper(j) + serviceTime(S_{i-1}))$
        $latestServiceStartTime(j) \leftarrow latestServiceEndTime(j) - serviceTime(S_{i-1})$
        $latestArrival(j) \leftarrow latestServiceStartTime(j)$
      end for
    end for
    if $twb = TW(S_{i-1})$ **then**
      return i
    end if
    $i \leftarrow i-1$
  end while
end function

Algorithm 4 Atualização de Melhor Tempo

Require: S: solução existente, i: índice a iniciar o recálculo

```
1: function updateBestTime(S,i)
2:     while i < |S| do
3:         twb ← cópia de TW(S_{i+1})
4:         for each j ∈ TW(S_{i+1}) do
5:             bestTime(j) ← latestDeparture(j)
6:         end for
7:         for each i ∈ TW(S_i) do
8:             for each j ∈ TW(S_{i+1}) do
9:                 bestTime(j) ← Min(bestTime(j), Max(earliestArrival(j), bestTime(i) + cost(S_i, S_{i+1})))
10:            end for
11:        end for
12:        if twb = TW(S_{i+1}) then
13:            return i
14:        end if
15:        i ← i + 1
16:    end while
17: end function
```

Uma vez que todas as informações das janelas de tempo se encontrem calculadas, torna-se trivial calcular o custo da rota, haja vista que ele é representado pela diferença de tempo de atendimento ótimo entre o nó final S_n e o nó inicial S_0, que já foram previamente calculados pelo Algoritmo 4. Portanto, o Algoritmo 5 apresenta o cálculo do custo da rota.

Algorithm 5 Operação de custo de rota com $\Delta - Evaluation$

```
1: function cost(S)
```

Require: S: solução existente

```
2:     return bestTime(S_n) − bestTime(S_0)
3: end function
```

3.2.3 Operações adaptadas para utilização do $\Delta - Evaluation$

Com base nos Algoritmos 2, 3 e 4, que realizam o $\Delta - Evaluation$, é possível implementar os operadores da meta-heurística utilizados para a modificação da solução. Neste trabalho foram implementados os operadores de remoção e inserção unitária e em sequência, e troca e movimento de posição de atendimentos, conforme apresentado na Seção 3.1. A seguir, a implementação dessas operações é detalhada.

3.2.3.1 Remoção

A operação de remoção remove um serviço de uma posição específica da rota, conforme apresentado no Algoritmo 6.

O algoritmo inicia preparando um vetor com a nova sequência de serviços, que exclui o serviço contido no índice i (linha 2). Em seguida, executa o Algoritmo 2 para atualizar os tempos mais cedo da posição removida i até o final da rota (linha 3). Na

sequência, executa o Algoritmo 3 para atualizar os tempos mais tarde da posição removida i até o início da rota (linha 4), armazenando o primeiro nó que foi alterado na rota na variável k. Ainda, executa o Algoritmo 4 para atualizar o tempo ótimo de atendimento (linha 5) a partir do serviço k, sobre as informações da nova rota já pré-computadas. Por fim, a solução gerada é retornada como resultado final (linha 6).

Algorithm 6 Operação de remoção de serviço com $\Delta - Evaluation$

Require: S: solução existente, i: posição a ser removida
1: **function** REMOVE(S,i)
2: $S' \leftarrow \{S_0, ..., S_{i-1}, S_{i+1}, ..., S_n\}$
3: $updateEarlestTimes(S', i)$
4: $k \leftarrow updateLatestTimes(S', i-1)$
5: $updateBestTime(S', k)$
6: **return** S'
7: **end function**

3.2.3.2 Inserção

A operação de inserção insere um serviço em uma posição específica da rota, conforme apresentado no Algoritmo 7.

O algoritmo inicia preparando um vetor com a nova sequência de serviços, que inclui os serviços presentes em S e também o serviço r inserido na posição i (linha 2). Em seguida, executa o Algoritmo 2 para atualizar os tempos mais cedo da posição inserida i até o final da rota (linha 3). Após isso, uma verificação de factibilidade da rota é realizada, para isso é verificado se o tempo de chegada do último nó foi alcançado, de forma que se o último nó não foi alcançado então a solução é infactível e uma solução inválida é retornada (linha 4 a 6). Na sequência, executa o Algoritmo 3 para atualizar os tempos mais tarde da posição inserida i até o início da rota (linha 7), armazenando o primeiro nó que foi alterado na rota na variável k. Ainda, executa o Algoritmo 4 para atualizar o tempo ótimo de atendimento (linha 8) a partir do serviço k. Por fim, a solução gerada é retornada como resultado final (linha 9).

3.2.3.3 Troca

A operação de troca realiza a inversão entre dois serviços em posições distintas de uma rota, conforme apresentado no Algoritmo 8. Diferentemente dos métodos de inserção e remoção, o método de troca realiza múltiplas modificações na rota em uma mesma chamada. Portanto, eventualmente são necessárias várias chamadas aos procedimentos de atualização de tempos caso uma única chamada não abranja os dois serviços alterados. Inicialmente, por questões de padronização, é realizada uma validação na entrada de informações para que as posições de troca i e j sejam sempre $i <= j$, realizando a troca desses de seus valores caso contrário.

Algorithm 7 Operação de inserção de serviço com $\Delta - Evaluation$

Require: S: solução existente, i: posição a ser inserida, r: requisições a ser inserida

```
1:  function INSERT(S,i,r)
2:      S' ← {S_0, ..., S_{i-1}, r, S_i, ..., S_n}
3:      updateEarlestTimes(S', i)
4:      if earliestArrival(S'_n) = ∞ then
5:          return nil
6:      end if
7:      k ← updateLatestTimes(S', i)
8:      updateBestTime(S', k)
9:      return S'
10: end function
```

O algoritmo inicia preparando um vetor com a nova sequência de serviços, que inclui os serviços presentes em S, porém com as posições dos serviços S_i e S_j trocadas (linha 3). Após a criação da nova sequência de serviços, é executado o Algoritmo 2 para atualizar os tempos mais cedo a partir da posição i e, novamente, a partir da posição j até o final da rota (linhas 4 e 5). Embora a chamada do procedimento $updateEarlestTimes$ possa atualizar os tempos até o final da rota, isto não é uma garantia, tendo em vista que as vezes a troca do serviço na posição i teve um impacto local que não chegou a se propagar até j. Assim, é necessário chamar o procedimento $updateEarlestTimes$ também para j para garantir que essa alteração também será propriamente tratada (linha 5). Caso a reavaliação dos tempos mais cedo a partir de i se propaguem até chegar a j, quando o procedimento for chamado para j o seu custo computacional será constante, tendo em vista que ele identificará que a solução já se encontra estável e não precisa mais ser recalculada. Por esse motivo, o procedimento é sempre chamado a partir de i e posteriormente a partir de j, caso contrário, as alterações eventualmente podem ser recalculadas desnecessariamente duas vezes para o serviço S_j. Após isso, uma verificação de factibilidade da rota é realizada, para isso é verificado se o tempo de chegada no último serviço foi alcançado. Caso o último nó não tenha sido alcançado, então a solução é infactível e uma solução inválida é retornada (linha 6 a 8).

Em seguida, executa o Algoritmo 2 para atualizar os tempos mais tarde da posição inserida i e j até o início da rota (linha 9 e 10). Portanto, de forma análoga à chamada dos procedimentos para cálculo dos tempos mais cedo, porém, de forma inversa, tendo em vista que os tempos mais tardes são calculados em direção ao início da rota, armazenando o primeiro nó alterado da rota por cada serviço i e j nas variáveis k_a e k_b, respectivamente. Ainda, o Algoritmo 4 utilizado para atualizar o tempo ótimo de atendimento é aplicado (linha 11 e 12) a partir dos serviços k_a e k_b, provenientes dos processos de atualização de tempos mais tarde, utilizando todos os tempos dos serviços da nova rota já pré-computadas. Por fim, a solução gerada é retornada como resultado final (linha 13).

Algorithm 8 Operação de troca de serviços com $\Delta - Evaluation$

Require: S: solução existente, i, j: posições a serem trocadas

1: **function** SWAP(S,i,j)
2: $i, j = min(i, j), max(i, j)$
3: $S' \leftarrow \{S_0, ..., S_{i-1}, S_j, S_{i+1}, ..., S_{j-1}, S_i, S_{j+1}, S_n\}$
4: $updateEarlestTimes(S', i)$
5: $updateEarlestTimes(S', j)$
6: **if** $earliestArrival(S'_n) = \infty$ **then**
7: **return** nil
8: **end if**
9: $k_a \leftarrow updateLatestTimes(S', j)$
10: $k_b \leftarrow updateLatestTimes(S', i)$
11: $updateBestTime(S', k_b)$
12: $updateBestTime(S', k_a)$
13: **return** S'
14: **end function**

3.2.3.4 Movimentação

Na operação de movimentação é realizado o movimento da posição de um serviço em uma mesma rota, sendo removido da posição i e reinserido na posição j, conforme apresentado no Algoritmo 9. O Algoritmo 9 é similar ao Algoritmo 8, sendo sua principal diferença concentrada na definição do novo vetor de sequência de serviços.

O algoritmo inicia verificando se o serviço na posição i será movido para frente ou para trás de sua posição atual, o que determina a forma como o vetor S' será criado (linhas 2 a 7). Além disso, essa verificação realiza a inversão dos valores de i e j caso seja necessário para que $i <= j$ (linha 6). Em seguida, entre as linhas 8 e 16, são realizadas a atualização dos tempos mais cedo (linha 8 e 9), a checagem de factibilidade (linha 10 a 12), a atualização dos tempos mais tarde (linha 13 e 14) e a atualização do melhor tempo (linha 15 e 16). O comportamento do algoritmo é idêntico ao trecho entre as linhas 4 a 12 do Algoritmo 8, de forma que seu detalhamento pode ser consultado na Seção 3.2.3.3. Por fim, a solução gerada é retornada como resultado final (linha 17).

3.2.4 Operadores

No intuito de demonstrar a aplicação das operações adaptadas na Seção 3.2.3 para utilização do $\Delta - Evaluation$, os operadores *Best-Insert* e 2-opt implementados no método de otimização baseado em ALNS proposto são detalhados nesta seção. Entretanto, cabe ressaltar que diversos outros operadores foram implementados neste trabalho, mas não serão aqui detalhados.

O operador *Best-Insert* é um operador relativamente simples de inserção gulosa, representado pelo Algoritmo 10, em que dada uma requisição r, o algoritmo verifica exaustivamente todas as possíveis opções de inserção, escolhendo ao final a posição com

Algorithm 9 Operação de movimentação de serviço com $\Delta - Evaluation$

Require: S: solução existente, i: posição a ser removida, j: posição a ser inserida

```
1:  function MOVE(S,i,j)
2:      if i <= j then
3:          S' ← {S_0, ..., S_{i-1}, S_{i+1}, ..., S_j, S_i, S_{j+1}, .., S_n}
4:      else
5:          S' ← {S_0, ..., S_{i-1}, S_j, S_i, ..., S_{j-1}, S_{j+1}, .., S_n}
6:          i, j = j, i
7:      end if
8:      updateEarlestTimes(S', i)
9:      updateEarlestTimes(S', j)
10:     if earliestArrival(S'_n) = ∞ then
11:         return nil
12:     end if
13:     k_a ← updateLatestTimes(S', j)
14:     k_b ← updateLatestTimes(S', i)
15:     updateBestTime(S', k_b)
16:     updateBestTime(S', k_a)
17:     return S'
18: end function
```

menor custo para realizar a inserção.

Esse operador inicia criando uma cópia da solução atual como a melhor solução encontrada (linha 2). Em seguida, um laço de repetição é iniciado (linhas 3 a 8), iterando sobre todas as possíveis posições de inserção na solução S. Para cada iteração, é realizada a operação de inserção utilizando $\Delta - Evaluation$ (linha 4), conforme relatado na Seção 3.2.3, seguida pela comparação da nova solução com a melhor solução encontrada até o momento (linha 5). Devido ao $\Delta - Evaluation$, essa comparação de avaliação possui complexidade de tempo constante. Caso a nova solução seja avaliada melhor que a melhor solução conhecida, então essa nova solução é tida com melhor solução encontrada até o momento (linha 6). Ao final do processo a melhor solução encontrada é retornada como resultado (linha 9).

Algorithm 10 Operador de inserção *Best-Insert*

Require: S: solução existente, r: requisição a ser inserida

```
1:  function BESTINSERT(S,r)
2:      bestSolution ← S
3:      for each i ← 0 to |S| do
4:          newSolution ← insert(S, i, r)
5:          if cost(newSolution) < cost(bestSolution) then
6:              bestSolution ← newSolution
7:          end if
8:      end for
9:      return bestSolution
10: end function
```

O operador 2-opt é um operador de otimização local complexo, representado pelo Algoritmo 11. O algoritmo tem por objetivo realizar a inversão de trechos da rota para verificar se essas modificações diminuem o custo da rota. Para isso, ele verifica exaustivamente para todas as possíveis combinações de pares de serviços em uma rota, qual inversão da sequência de nós entre esses serviços possui a capacidade de melhor reduzir o custo da rota, realizando essa inversão no trecho que minimize o custo da rota.

Esse operador inicia criando uma cópia da solução atual como a melhor solução encontrada (linha 2). Em seguida, um laço de repetição é iniciado (linhas 3 a 11), iterando sobre todas as possíveis posições de início de inversão i na solução S. Para cada iteração é realizada a cópia da solução inicial (linha 4) e iniciado outro laço de repetição aninhado (linhas 5 a 10), partindo da posição posterior a i até o final da rota, iterando sobre todas as possíveis posições de final da inversão j da solução S. Cada uma dessas iterações incrementalmente modifica a cópia da solução criada na linha 4, por meio de uma única operação de movimento (linha 6), conforme relatado na Seção 3.2.3, de forma que ela represente a inversão dos nós entre o índice i e j da rota. Em seguida, é feita a comparação da nova solução com a melhor solução encontrada até o momento (linha 7). Devido ao $\Delta - Evaluation$, essa comparação de avaliação possui complexidade de tempo constante. Caso a nova solução seja avaliada melhor que a melhor solução conhecida, então essa nova solução é tida com melhor solução encontrada até o momento (linha 8). Ao final do processo a melhor solução encontrada é retornada como resultado (linha 12).

Algorithm 11 Operador de otimização 2-opt

```
Require: S: solução existente
 1: function BESTINSERT(S)
 2:     bestSolution ← S
 3:     for each i ← 0 to |S| do
 4:         newSolution ← S
 5:         for each j ← i+1 to |S| do
 6:             newSolution ← move(newSolution, i, j)
 7:             if cost(newSolution) < cost(bestSolution) then
 8:                 bestSolution ← newSolution
 9:             end if
10:         end for
11:     end for
12:     return bestSolution
13: end function
```

3.3 IMPLEMENTAÇÃO E TECNOLOGIA

A implementação do método proposto foi desenvolvida em etapas ao longo do progresso da pesquisa. Utilizando como base a metodologia Pesquisa-Ação, o desenvolvimento se deu de forma incremental onde, a cada etapa, era realizado o levantamento básico de

requisitos, o planejamento das alterações, a implementação ou refatoração dos algoritmos, a elaboração e execução dos testes, e a análise dos resultados e proposta de alterações. Portanto, ao final de cada ciclo os resultados dos testes eram avaliados para servirem de apoio à decisão de quais atividades desenvolver no próximos ciclo.

Sob o ponto de vista do desenvolvimento, o software desenvolvido foi codificado utilizando a linguagem de programação Java (Versão 11). Essa linguagem de programação foi escolhida principalmente devido a sua robusteza, ferramentas de auxílio de desenvolvimento disponíveis e boa performance computacional.

Para garantir a qualidade do protótipo produzido, diversos testes foram realizados durante o seu desenvolvimento utilizando a biblioteca de teste JUnit, permitindo o avanço rápido do desenvolvimento devido à garantia de funcionamento e conformidade das partes implementadas. Esses testes foram implementados de forma a avaliar principalmente os operadores e procedimentos utilizados durante a execução do método de otimização. Para os operadores e outros procedimentos determinísticos foram realizadas avaliação por meio de testes exatos, enquanto que para operadores e outros procedimentos estocásticos foram realizadas avaliações estatísticas utilizando o método qui-quadrado (χ^2) para comprovar se os métodos aleatórios estavam se comportando dentro dos parâmetros esperados.

Além dos testes de validação, também foram realizados testes de desempenho. A ferramenta JMH *Java Microbenchmark Harness* permitiu não apenas medir o desempenho dos algoritmos implementados, mas também comparar de forma adequada diferentes estratégias para a implementação deles. Também foi utilizado o Software VisualVM, com suas ferramentas *monitor*, *sampler* e *profiler*, para analisar a utilização de recursos computacionais durante a execução do programa e identificar gargalos, códigos não otimizados, entre outros problemas de desempenho a serem aprimorados. Para complementar estes testes, também foram implementadas instrumentações para avaliar o desempenho da metaheurística durante sua execução, que incluem registros em arquivos de texto tabulares (formato CSV) e ferramentas web interativas para monitoramento do progresso e estado da otimização.

4 ANÁLISE DOS RESULTADOS

Este capítulo apresenta os experimentos realizados e resultados alcançados pelo algoritmo que implementa o método proposto. A Seção 4.1 detalha o protocolo de experimentação utilizado nos testes. A Seção 4.2 apresenta as bases de dados utilizadas nos experimentos e as características das instâncias de *benchmark* de cada uma delas. A Seção 4.3 detalha os valores dos parâmetros utilizados na realização dos testes. A Seção 4.4 analisa os resultados obtidos pela otimização realizada pelo algoritmo proposto, enquanto a Seção 4.5 compara seu tempo de execução. A Seção 4.6 detalha aspectos intrínsecos da execução do algoritmo, destacando suas vantagens e limitações. Por fim, a Seção 4.7 apresenta as considerações sobre os resultados alcançados.

4.1 PROTOCOLO DE EXPERIMENTAÇÃO

O protocolo de experimentação utilizado neste trabalho tem como objetivo analisar e comparar o método proposto com os trabalhos pertencentes ao estado da arte na solução do VRPMTW encontrados na literatura. A análise visa investigar o comportamento do método proposto, tanto quantitativamente, no que diz respeito ao seu desempenho perante outras abordagens presentes na literatura, quanto qualitativamente, referente ao seu comportamento durante a execução.

Portanto, ao todo são realizadas três analises, duas quantitativas e uma qualitativa.

- A primeira análise quantitativa visa avaliar a minimização da função objetivo das soluções geradas pelo algoritmo proposto e compará-las com os resultados alcançados por outros trabalhos do estado da arte encontrados na literatura.
- A segunda análise quantitativa visa avaliar o tempo de processamento do algoritmo proposto para mensurar os ganhos de tempo obtidos pela implementação do Δ-*Evaluation* no cálculo da factibilidade das janelas de atendimento e dos tempos das rotas.
- Por fim, a análise qualitativa tem o intuito de avaliar o comportamento interno do algoritmo no decorrer de sua execução, analisando aspectos relacionados a curva de convergência e temperatura do ALNS, entre outros. Ainda, nessa análise qualitativa são destacadas as principais contribuições do algoritmo, assim como suas limitações.

Para comparar a minimização do custo das soluções produzidas pelo método proposto foram selecionados os trabalhos realizados por Belhaiza, Hansen e Laporte (2014), Belhaiza, M'Hallah e Ben Brahim (2017) e Hoogeboom et al. (2020). Todos eles utilizaram a mesma base de instâncias de *benchmark* que também será utilizada a avaliação do trabalho em tela. Portanto, será possível comparar objetivamente os resultados alcançados por este trabalho com os resultados já existente na literatura. Após a execução de todos

os experimentos e coleta dos resultados, para cada instância do *benchmark* é calculado o Desvio Relativo Percentual (*Relative Percent Difference* – RPD) entre o custo obtido pelo algoritmo proposto (F^{Br}) e os trabalhos da literatura ($F^{LitBest}$) aos quais ele é comparado, sendo o valor RPD dado por $(F^{Br} - F^{LitBest})/F^{LitBest}$. Sobre esses resultados é aplicado o teste de Wilcoxon (WILCOXON, 1945) para refutar a hipótese nula e garantir a significância estatística dos resultados obtidos.

Para analisar os benefícios da utilização do cálculo incremental (Δ-*Evaluation*) na análise da factibilidade das janelas de atendimento e cálculo dos tempos das rotas da solução, foram realizado três experimentos utilizando como base o método proposto. No primeiro, foram aferidos os tempos de execução das instâncias de *benchmark* utilizando o cálculo completo da factibilidade das janelas de atendimento e tempos das rotas. No segundo, foram aferidos os tempos de execução das mesmas instâncias utilizando o cálculo incremental na análise da factibilidade das soluções e o cálculo completo dos tempos das rotas, conforme proposto no trabalho de Hoogeboom et al. (2020). Por fim, no terceiro, foram aferidos os tempos de execução das mesmas instâncias utilizando o cálculo incremental na análise da factibilidade das soluções e no cálculo dos tempos das rotas, abordagem proposta no trabalho em tela. O objetivo dessa análise é avaliar se há redução no tempo de computação utilizando o cálculo incremental também na definição dos tempos das rotas contidas na solução. Os três experimentos foram realizados sobre o método proposto neste trabalho para evitar qualquer influência externa ao cálculo incremental Δ-*Evaluation* na criação e otimização das soluções. Ainda, dessa forma é possível eliminar qualquer outra influência de características relacionadas à diferença de linguagens de programação e até mesmo de hardware na análise das contribuições da abordagem Δ-*Evaluation* em todo o ciclo de otimização. Após a execução dos três experimentos é feita uma comparação dos tempos obtidos neles.

Além das avaliações quantitativas, também são feitas avaliações qualitativas. Para realizar essas avaliações são selecionadas instâncias do *benchmark* que apresentem características relevantes à execução do método proposto. Por exemplo, as instâncias onde o método proposto obteve as maiores diferenças, tanto positiva quanto negativamente, em relação aos trabalhos utilizados na comparação. Dessa forma, pretende-se avaliar as vantagens observadas no método proposto, assim como apontar suas limitação em relação ao estado da arte da literatura. Portanto, a partir da análise qualitativa será observado o comportamento do método proposto em aspectos como a temperatura do ALNS, a curva de convergência, a evolução da solução, entre outros fatores.

Com base na revisão da literatura, para analisar o método proposto foram selecionadas as instâncias de *benchmark* propostas por Belhaiza, Hansen e Laporte (2014). Esse *benchmark* é comumente utilizado como referência para o VRPMTW por diversos trabalhos devido à sua abrangência tanto na distribuição espacial dos pontos, quanto no aspectos relacionados às múltiplas janelas de tempo. As instâncias desse *benchmark* têm como

objetivo principal a minimização do número total de veículos, seguida pela minimização do tempo total da viagem dos veículos (somando tempo do veículo em deslocamento e em espera). Mais detalhes sobre as instâncias de *benchmark* usadas são apresentados na Seção 4.2.

A maioria dos pesos e parâmetros utilizados na meta-heurística são fixos e definidos antes do início da execução do algoritmo com base em testes empíricos e valores sugeridos pela literatura. Contudo, por se tratar de uma meta-heurística adaptativa, o algoritmo tem capacidade de automaticamente ajustar alguns de seus parâmetros, tais como: o número de iterações (influência sobre o critério de parada), peso para seleção de operadores, temperatura, penalidade por violação de restrições, entre outros. Maiores detalhes sobre os parâmetros utilizados na execução do algoritmo que implementa o método proposto são abordados na Seção 4.3.

Por utilizar uma abordagem estocástica, para se mitigar qualquer viés nas otimizações geradas pelo método proposto, nos experimentos, para cada instância de *benchmark* foram realizadas 30 execuções do algoritmo. Durante cada iteração, e ao final de sua execução, são coletadas diversas informações relevantes à avaliação do seu desempenho, para permitir uma posterior análise detalhada de seu comportamento e de determinadas variáveis durante o processo de otimização.

Todos os experimentos com a execução das instâncias de *benchmark* pelo algoritmo foram realizados no mesmo computador, dotado de um processador *Intel(R) Core(TM) i7-3770 @ 3.40 GHz*, com 4 núcleos físicos e 8 núcleos virtuais de processamento, e 24.0 GB de memória RAM. Com relação ao software, os experimentos foram realizados utilizando como *runtime environment* Java o OpenJDK 11.0.11, executado no sistema operacional Ubuntu 18.04.5 LTS. Embora a implementação do algoritmo na linguagem de programação Java suporte sua execução em diferentes arquiteturas de hardware e software, sem necessidade de adaptações, entretanto, a execução dos experimentos foi realizada em um único equipamento para permitir a comparação de tempos de execução do algoritmo de forma mais confiável. Ainda, dado que as instâncias do *benchmark* não são significativamente grandes, pois têm em torno de 100 serviços, a sua execução no ambiente proposto não se tornou um impeditivo.

Ainda, é importante destacar que nem todos os recursos computacionais disponíveis foram utilizados na execução do algoritmo, haja vista que, considerando os cenários avaliados, por exemplo, a quantidade de memória RAM disponível no sistema é muito maior do que o necessário. Também, embora o processador utilizado possua 4 núcleos físicos, não foram dedicados esforços para paralelizar a computação de uma solução. Todavia, para obter todas as soluções mais rapidamente, foi implementado o paralelismo da avaliação de múltiplas execuções de uma mesma instâncias, sendo para isso utilizada a biblioteca padrão *Stream* da linguagem Java.

Ao final da execução das três análises realizadas, um abrangente resumo das

considerações assimiladas a partir dos experimentos é apresentado na Seção 4.7.

4.2 BASE DE DADOS

Durante os experimentos realizados neste trabalho são utilizadas as instâncias de *benchmark* para VRPMTW proposta por Belhaiza, Hansen e Laporte (2014). Esse *benchmark* é composto por conjuntos de instâncias adaptadas de Solomon (1987), com a inclusão de múltiplas janelas de atendimento, haja vista que elas originalmente não possuíam essas características, e também por novas instâncias construídas por Belhaiza, Hansen e Laporte (2014).

A partir de algumas instâncias originais de Solomon (1987), Belhaiza, Hansen e Laporte (2014) aplicaram um procedimento para a adaptação e criação de múltiplas janelas de tempo. Portanto, para cada requisição foram definidos o número mínimo e máximo de janelas de tempo a serem criadas, o intervalo mínimo e máximo entre as janelas de tempo, e o tempo de duração de cada uma das janelas (BELHAIZA; HANSEN; LAPORTE, 2014). Além dessas adaptações a instâncias existentes, Belhaiza, Hansen e Laporte (2014) também criaram novas instâncias, seguindo o mesmo padrão de parametrização das instâncias anteriores.

A escolha desse conjunto de instâncias de *benchmark* se mostrou relevante não apenas por ser bastante utilizado por trabalhos que tratam de VRPMTW, mas também porque elas possuírem características, tais como: dispersão de posicionamento entre os serviços, quantidade de janelas de tempo, amplitude das janelas de tempo, distanciamento entre janelas de tempo e quantidade de veículos necessários.

Uma das características mais relevantes das instâncias do *benchmark* é referente ao modo de geração das múltiplas janelas de tempo. Ao todo, o conjunto de dados é composto por 72 instâncias que podem ser classificadas em instâncias do tipo P (24 instâncias separadas em 6 classes, com resultado ótimo conhecido, dada a criação a partir de instâncias sem janelas de tempo com resultado ótimo conhecido) e instâncias do tipo M (48 instâncias separadas em 6 classes, sem resultado ótimo conhecido). Enquanto as instâncias do tipo M não possuem solução ótima comprovada, as instâncias do tipo P possuem suposto valor ótimo conhecido. Esses valores foram obtidos por meio de um processo de relaxação das janelas de tempo em que o caminho ótimo é encontrado ignorando a noção de tempo e, posteriormente, as janelas de tempo são adicionadas de forma que pelo menos uma das janelas de tempo contenha o tempo de atendimento ótimo do serviço. Efetivamente, isso faz com que o caminho mais curto seja também o caminho mais rápido, haja vista que ele deve conter tempo de espera nulo. Como os trabalhos utilizados para comparação utilizaram apenas as instâncias do tipo M, com número maior de janelas de tempo, portanto, neste trabalho serão abordadas características pertinentes apenas às instâncias essa tipo classe.

De acordo com as características do *benchmark*, os autores dividiram as suas instâncias em 6 classes diferentes, totalizando 48 instâncias, conforme apresentado na Tabela 5 e detalhado a seguir.

Tabela 5 – Características das instâncias por grupos.

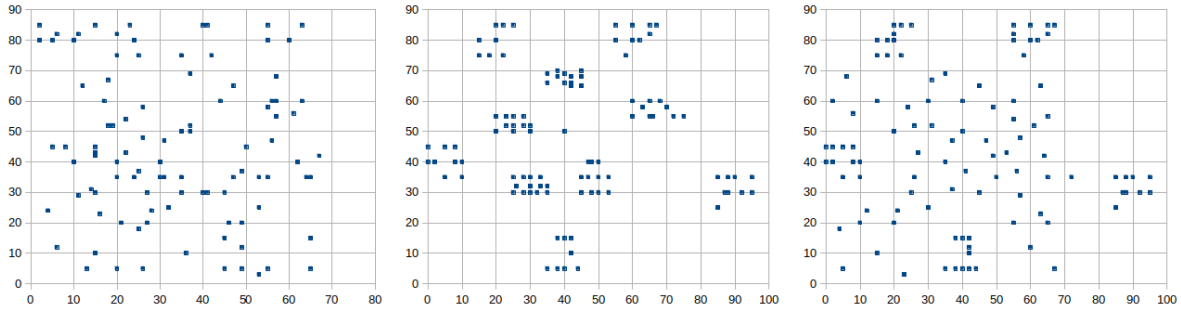
Classe	Instâncias	Serviços	Cluster	Random	Num. Janelas	Horizonte T.
RM1	8	100	Não	Sim	Maior	Menor
RM2	8	100	Não	Sim	Maior	Maior
CM1	8	100	Sim	Não	Maior	Menor
CM2	8	100	Sim	Não	Maior	Maior
RCM1	8	100	Sim	Sim	Maior	Menor
RCM2	8	100	Sim	Sim	Maior	Maior

Todas as instâncias propostas por Belhaiza, Hansen e Laporte (2014) possuem 100 serviços, sendo as do tipo 2 geradas por eles, enquanto as do tipo 1 são baseadas nas instâncias de Solomon (1987). O tempo (custo) de deslocamento entre dois serviços é dado pela distância euclidiana entre as coordenadas X e Y dos nós dos serviços. Outras características herdada das instâncias de Solomon (1987) diz respeito a capacidade dos veículos e ao tempo de serviço, ambas constantes para todos os serviços de uma mesma instância.

Uma das características mais visíveis que distingue as classes de instâncias está relacionada à distribuição espacial dos serviços, ou seja, como eles estão localizados dentro do plano cartesiano. As instâncias propostas por Solomon (1987) são rotuladas com tipo R, C, ou RC. Conforme pode ser visto na Figura 9, em que são plotadas as distribuições das instâncias $RM1$, $CM1$ e $RCM1$, nas instâncias do tipo R os serviços estão espacialmente distribuídos de forma uniformemente randômica no espaço. Nas instâncias do tipo C os serviços estão espacialmente agrupados em *clusters*, isto é, os nós estão espacialmente distribuídas dentro de diferentes bolsões concentradores de serviços, formando pequenos grupos. Nas instâncias do tipo RC os serviços estão espacialmente distribuídos tanto de forma randômica quando de forma agrupada, ou seja, esse tipo de instância é uma mescla entre os tipos R e C, onde parte dos serviços possui características do tipo R e outra parte possui características do tipo C. Para as diferentes instâncias da mesma classe, as posições dos serviços permanecem os mesmos, modificando apenas os outros parâmetros.

Além das coordenadas de posição dos serviços, provenientes das instâncias de Solomon (1987), Belhaiza, Hansen e Laporte (2014) tiveram que incluir múltiplas janelas de tempo nas instâncias. Para isso, foram parametrizados o número mínimo e máximo de janelas de tempo a serem criadas, a distância mínima e máxima entre janelas de tempo do mesmo serviço, e o tamanho mínimo e máximo das janelas de tempo. Os valores utilizados são selecionados randomicamente entre esses limites. Portanto, eles tomaram o cuidado para que valores efetivados se encontrassem aproximadamente distribuídos de

Figura 9 – Distribuição espacial de instâncias do tipo R, C, ou RC, respectivamente



Fonte: Próprio autor

forma uniforme entre os limites escolhidos para um dado parâmetro.

Visando abranger o maior número de cenários possíveis, os parâmetros de criação das instâncias foram variadas, incluindo o número de janelas de tempo por serviço e o horizonte de tempo dos cenários. O tamanho do horizonte de tempo do cenário foi variado de forma que um horizonte de tempo maior implica em janelas de tempo maiores e mais espaçadas, enquanto que um horizonte de tempo menor implica em janelas de tempo mais curtas e com distanciamento menor. Com esta parametrização, as instâncias do tipo P foram criadas contendo um número de janelas menor, variando entre 1 a 6, na média entre 1.5 e 3.8, enquanto as instâncias do tipo M variaram entre 1 a 10, na média entre 2.7 a 5.4. Além disso, as instâncias do tipo 1 foram criadas contendo um horizonte de tempo reduzido, variando entre 230 e 1236 unidades de tempo, com valor médio de 569, enquanto as instâncias do tipo 2 variam entre 960 e 3390 unidades de tempo, com valor médio de 1783. Segundo Solomon (1987), este fator faz com que as instâncias do tipo 1 necessitem de mais veículos em sua solução mínima, ao contrário das instâncias do tipo 2, que têm horizonte de tempo e tamanho das janelas maiores, o que diminui a quantidade necessária de veículos, pois neste cenário existe maior liberdade para combinar os serviços e assim realizar rotas maiores.

Todas as instâncias de *benchmark* do problema utilizadas neste trabalho estão disponíveis para acesso aberto na internet por meio da página pessoal do autor das instâncias (BELHAIZA, 2014). As instâncias são fornecidas pelos autores em um arquivo no formato *.rar* que contém diversos arquivos de texto no formato *.txt* organizadas em pastas. Cada um dos arquivos de texto representa uma instância de *benchmark* e se encontra dentro de uma pasta com o nome do identificador da classe da instâncias.

O conteúdo dos arquivos de texto de instâncias é interpretado no formato tabular. Cada arquivo inicia com duas linhas de cabeçalho, que contêm informações como os parâmetros utilizados para criar a instância, quantidade de serviços, quantidade de depósitos, quantidade de veículos, capacidade dos veículos, entre outros. É importante destacar que todas as instâncias disponibilizadas e utilizadas nos experimentos contêm frota homogênea de veículos. A partir da terceira linha são definidos os nós do problema

(depósito e serviços), contendo o identificador do nó, a coordenada espacial X, a coordenada espacial Y, a quantidade de tempo de serviço, a quantidade de demanda do serviço, se é um depósito ou um requisição de serviço, o número de janelas de tempo, seguida pelos pares de início e fim de cada janela de tempo. Essas instâncias contêm sempre um único depósito na primeira linha, com identificador igual a 0, uma única janela de tempo representando a jornada máxima do veículo, e os tempos de serviço e de demanda nulos. O custo de deslocamento é igual ao tempo de deslocamento, que é dado pela distância euclideana bidimensional entre os pontos formados pela coordenada X e Y dos dois nós.

4.3 DEFINIÇÃO DE PARÂMETROS DO ALGORITMO

A definição dos valores dos parâmetros do algoritmo ALNS foi feita com valores sugeridos por outros trabalhos encontrados na literatura e aprimorados para o algoritmo desenvolvido por meio de testes empíricos.

O processo de refinamento dos parâmetros foi realizado a partir da execuções do algoritmo em um conjunto diversificado de amostras de instâncias (selecionados a partir do conjunto de instâncias *benchmark* utilizadas), sendo que após cada execução os resultados foram numericamente avaliados, permitindo comparar o efeito das modificações nos valores dos parâmetros, identificando efeito neutro, positivo, ou degradante no resultado final. De acordo com a modificação no resultado, os valores dos parâmetros foram sucessivamente ajustados até que o melhor resultado fosse encontrado. Uma vez que estes parâmetros foram definidos, todas as avaliações realizadas neste trabalho utilizaram o mesmo valor.

A Tabela 6 detalha os parâmetros gerais utilizados para controlar o algoritmo, tais como controle da temperatura e função de aceitação. Embora também tenham sido implementados outros critérios de parada, tais como por tempo de execução e por estagnação, devido à forma de avaliação utilizada pela literatura, apenas o critério de parada por número de iterações é utilizado na execução dos testes.

A Tabela 7 detalha os parâmetros referentes às métricas de avaliações de solução, tais como custos extras, penalidades, entre outros. É importante notar que esses parâmetros são utilizados como penalidade apenas quando há possibilidade de aceitação de soluções infactíveis. Portanto, nos momentos em que não são toleradas soluções infactíveis eles não são utilizados.

Além dos parâmetros com valores fixos apresentados na Tabela 6 e na Tabela 7, por se tratar de um algoritmo adaptativo, ele é capaz de ajustar automaticamente alguns de seus parâmetros durante a execução. Na implementação realizada neste trabalho, o algoritmo realiza ajustes automáticos nos pesos dos operadores utilizados, isto é, manipula automaticamente a probabilidade de utilização de cada um dos operadores de acordo com o seu desempenho na otimização que está sendo realizada, assim como seu reaquecimento. A Tabela 8 apresenta os parâmetros referentes ao critério de seleção e ajuste de pesos dos

Tabela 6 – Definição dos valores dos parâmetros – Parâmetros gerais da meta-heurística e operadores

Parâmetro	Valor	Descrição
Critério de parada por iterações	100000	Número total de iterações a serem executadas.
Função de Temperatura (T)	$\left(1 - \frac{itCur}{itMax}\right)^2$	Função de decaimento que define a temperatura do sistema, representada pelo quadrado do inverso das iterações restantes, em que $itCur$ indica a quantidade de iterações ocorridas e $itMax$ indica a quantidade máxima de iterações que podem ser executadas.
Função de Aceitação ($accept$)	$T \times \frac{cost(S)}{cost(S')}$	Função de aceitação que determina a probabilidade de aceitação de uma nova solução. Ela se baseia na temperatura atual do sistema (T) multiplicada pela razão entre o custo da solução corrente (S) e da nova solução calculada (S'). Portanto, quanto maior a temperatura e quanto menos pior for a nova solução comparada à solução corrente, maior será a probabilidade de aceitação.
q_{min}	2	Limite mínimo de número de serviços que podem ser removidas ao utilizar um operação de remoção.
q_{max}	$\min\{2, 0.3n\}$	Limite máximo de número de serviços que podem ser removidas ao utilizar um operação de remoção, considerando n o número de serviços em uma solução.

Tabela 7 – Definição dos valores dos parâmetros – Parâmetros de avaliação da solução

Parâmetro	Valor	Descrição
Custo booleano por quebra de restrições	50	Penalidade extra por cada restrição quebrada (independente de intensidade).
Custo proporcional por quebra de restrição	100	Penalidade extra equivalente ao tamanho da violação da restrição.
Custo mínimo por quebra de restrição	5	Penalidade mínima para cada restrição violada (booleana ou proporcional).
Intensificador da quebra da restrição	2	Valor da potência intensificadora para cada restrição violada.

operadores.

Além desses parâmetros internos, relacionados à execução do algoritmo, outros parâmetros são dependentes do problema, tais como, os serviços (suas demandas, tempo de serviços, custo e tempo de deslocamento), os veículos disponíveis (quantidade, capacidade, custo), e as customizações do objetivo (necessidade de diminuir frota e/ou jornada de trabalho), entre outros, conforme mencionado na Seção 4.2.

Tabela 8 – Definição dos valores dos parâmetros – Seleção de operadores

Parâmetro	Valor	Descrição
θ_1	20	Pontuação dada aos operadores quando uma melhor solução global é encontrada.
θ_2	10	Pontuação dada aos operadores quando uma melhor solução local é encontrada.
θ_3	2	Pontuação dada aos operadores quando uma pior solução local é encontrada.
ρ	0.1	Fator de reação, peso histórico no ajuste dos pesos dos operadores.
k	100	Tamanho do segmento (número de iterações) necessário para atualização dos pesos dos operadores.

4.4 ANÁLISE DO CUSTO

Esta seção apresenta a análise quantitativa realizada a partir dos resultados obtidos da execução do algoritmo para as instâncias de teste apresentadas na Seção 4.2. Os resultados foram comparados com os mais recentes trabalhos relacionados encontrados na literatura. Para quantificar as diferenças assimiladas na comparação, foi calculada a métrica RPD e a relevância estatística dos resultados foi aferida pelo método de Wilcoxon, conforme descrito na Seção 4.1.

A métrica RPD quantifica percentualmente a diferença entre o valor das soluções encontrada pelo método proposto e a apresentada pelo trabalho relacionado. Essa diferença é calculada sobre o custo da solução encontrada por ambos os trabalhos. Portanto, um RPD igual a 0% significa que ambos os trabalhos chegaram a soluções com o mesmo valor, um valor de RPD positivo significa que solução encontrada pelo método proposto teve custo maior que a obtida pelo trabalho relacionado (portanto, resultado pior), enquanto que um RPD negativo significa que a solução encontrada pelo método proposto teve custo menor que o da literatura (portanto, resultado melhor).

Devida a grande quantidade de estatísticas e ao elevado número de instâncias avaliadas, esta seção apresenta de forma sumarizada os resultados obtidos na análise quantitativas das soluções encontradas pelo método proposto.

A Tabela 9 apresenta a comparação dos resultados alcançados pelo método proposto no trabalho em tela em relação aos trabalhados de Belhaiza, Hansen e Laporte (2014), Belhaiza, M'Hallah e Ben Brahim (2017) e Hoogeboom et al. (2020). Essa tabela é composta por 5 colunas principais, onde cada uma representa um dos trabalhos comparados. Cada linha da tabela apresenta os melhores resultados dos algoritmos para uma determinada instância analisada. Para cada trabalho analisado são apresentadas duas colunas, uma contendo o número de veículos da melhor solução encontrada (NV) e a outra contendo

o custo da melhor solução encontrada (OBJ). Para as colunas referentes ao resultados do trabalho em tela, foi utilizado o o melhor valor observado dentre todas as execuções da mesma instância. Ainda, os valores em negrito representam o valor da melhor solução encontrada entre todos os trabalhos para uma determinada instância. A última coluna da tabela apresenta a métrica de RPD dos resultados alcançados pelo trabalho em tela e o melhor resultado encontrado nos trabalhos comparados. Por fim, a última linha da tabela apresenta o número de melhores soluções encontradas por cada um dos trabalhos avaliados.

Na Tabela 9, é possível observar que para as 48 instâncias avaliadas, o trabalho de Belhaiza, Hansen e Laporte (2014), que propôs as instâncias de *benchmark*, apresentou melhores resultados para apenas 1 das 48 instâncias, sendo esta uma instância de distribuição randômica com horizonte de tempo reduzido. Já o trabalho de Belhaiza, M'Hallah e Ben Brahim (2017) encontrou melhores resultados para 10 das 48 instâncias avaliadas, sendo assim o segundo melhor trabalho em número de melhores resultados obtidos nesta avaliação. Ele encontrou melhores resultados principalmente nas instâncias randômicas de horizonte de tempo reduzido e agrupadas de horizonte de tempo longo.

Hoogeboom et al. (2020) propuseram dois algoritmos, um chamado A-AVNS e o outro chamado E-AVNS. O A-AVNS não conseguiu nenhum resultado de otimização melhor que os outros, sendo, portanto, o que apresentou o pior desempenho. Já a implementação E-AVNS foi capaz de encontrar melhores resultados para 6 das 48 instâncias, se tornando assim o terceiro melhor trabalho desta avaliação. Ele apresentou melhor desempenho principalmente em instâncias agrupadas que possuem horizonte de tempo reduzido.

Por fim, o trabalho em tela apresentou melhores resultados em 31 das 48 instâncias analisadas. Portanto, ele pode ser considerado o método que apresentou a maior quantidade de melhores resultados nesta avaliação. Ele apresentou melhor desempenho em instâncias randômicas de horizonte de tempo longo, agrupadas com horizonte de tempo longo e randômico agrupadas com horizonte de tempo curto e longo, sendo que neste último cenário ele foi melhor em 14 das 16 instâncias. Ainda, é importante destacar que o método proposto no trabalho em tela conseguiu atingir o número mínimo de veículos alcançados pelos trabalhos relacionados, obtendo um número menor de veículos para a instância RCM207.

A última coluna da tabela representa o valor do RPD entre a solução proposta e a melhor solução da literatura. Este RPD é dado pela razão entre a diferença entre os resultado obtido (considerando o método proposto e o melhor resultado da literatura) e pelo melhor resultado alcançado pela literatura. Portanto, o valor do RPD representa uma porcentagem de desvio que o método proposto obteve comparado-o à literatura. Como o número de veículos foi equivalente nos dois cenários, será analisado o RPD apenas para o valor da função objetivo. Com base na média geral apresentada na tabela, foi possível observar uma redução média de 0,77% do valor da função objetivo ao utilizar o novo

Tabela 9 – Comparativos de resultados nas instâncias de Belhaiza, Hansen e Laporte (2014)

	BHL14		BMB17		A-AVNS		E-AVNS		Trabalho em tela		
Instance	NV	OBJ	NV	OBJ	NV	OBJ	NV	OBJ	NV	OBJ	RPD
RM101	10	3041,9	10	3027,1	10	3064,5	10	3026,1	**10**	**2981,09**	-1,49%
RM102	9	2765,1	9	2751,2	9	2788,6	9	2774,8	**9**	**2709,74**	-1,51%
RM103	9	2708,5	9	2703,0	9	2727,0	9	2700,6	**9**	**2697,97**	-0,10%
RM104	9	2718,0	**9**	**2701,2**	9	2733,2	9	2707,1	9	2702,15	+0,04%
RM105	9	2688,8	**9**	**2687,2**	9	2713,2	9	2690,5	9	2691,72	+0,17%
RM106	**9**	**2692,9**	9	2708,4	9	2727,6	9	2714,8	9	2713,73	+0,77%
RM107	9	2701,4	**9**	**2692,8**	9	2716,1	9	2700,4	9	2696,85	+0,15%
RM108	9	2729,1	**9**	**2722,6**	9	2735,9	9	2738,1	9	2735,72	+0,48%
RM201	3	3808,2	3	3805,4	2	2925,9	2	2888,9	**2**	**2745,47**	-4,96%
RM202	2	2739,0	2	2706,8	2	2754,0	2	2721,9	**2**	**2672,69**	-1,26%
RM203	2	2710,3	2	2696,9	2	2700,2	2	2693,2	**2**	**2665,05**	-1,05%
RM204	2	2691,9	2	2674,5	2	2692,7	2	2671,7	**2**	**2659,12**	-0,47%
RM205	2	2689,9	2	2668,1	2	2681,7	2	2668,4	**2**	**2652,28**	-0,59%
RM206	2	2703,4	2	2684,9	2	2689,2	2	2672,6	**2**	**2665,54**	-0,26%
RM207	2	2701,7	2	2664,3	2	2667,9	2	2662,4	**2**	**2657,41**	-0,19%
RM208	2	2682,8	2	2664,3	2	2678,4	**2**	**2663,6**	2	2664,54	+0,04%
CM101	10	3320	10	3319,1	10	3496,5	10	3345,4	**10**	**3267,82**	-1,54%
CM102	12	3492,1	**11**	**3410,7**	11	3458,7	11	3482,3	11	3543,79	+3,37%
CM103	12	3641,1	12	3632,4	11	3666,7	**11**	**3592,2**	11	3621,32	+0,81%
CM104	14	4087,8	14	4098	13	3978,4	**13**	**3927,8**	13	3982,84	+1,40%
CM105	11	3083,4	**10**	**3027**	10	3097,5	10	3066,3	10	3128,94	+3,37%
CM106	10	3073,9	10	3059	10	3191,3	10	3066,4	**10**	**3007,83**	-1,67%
CM107	11	3324,2	11	3318	10	3138,6	**10**	**3108,4**	10	3126,29	+0,58%
CM108	10	2990,4	10	2986	10	3010,3	10	2985,9	**10**	**2968,47**	-0,58%
CM201	5	4520,1	5	4498,8	5	4576,6	**5**	**4468,4**	5	4468,41	0,00%
CM202	6	5027,3	6	5025,1	6	5079,6	6	5020,2	**6**	**4988,09**	-0,64%
CM203	5	4497,2	5	4465,8	5	4562,6	5	4486,5	**5**	**4458,33**	-0,17%
CM204	5	4359,8	5	4344	5	4399,4	5	4356,9	**5**	**4328,71**	-0,35%
CM205	4	3884,1	**4**	**3827,8**	4	3907,7	4	3896,8	4	3869,86	+1,10%
CM206	4	3767,7	**4**	**3713,2**	4	3791,1	4	3733,4	4	3724,89	+0,31%
CM207	4	4009,7	4	3963,7	4	4007,5	4	3963,7	**4**	**3950,30**	-0,34%
CM208	4	3788,1	**4**	**3749,7**	4	3783,7	4	3756,8	4	3754,26	+0,12%
RCM101	10	3098,9	10	3081,2	10	3093,3	10	3080,6	**10**	**3062,01**	-0,60%
RCM102	10	3222,6	10	3188,3	10	3194,8	10	3184,3	**10**	**3121,67**	-1,97%
RCM103	10	3174,3	10	3150,4	10	3171	10	3148,3	**10**	**3131,75**	-0,53%
RCM104	10	3156,3	10	3144	10	3157,8	10	3141,2	**10**	**3129,03**	-0,39%
RCM105	10	3216,7	10	3207	10	3233,3	10	3208,2	**10**	**3186,82**	-0,63%
RCM106	10	3219,9	**10**	**3181,7**	10	3211,1	10	3191,8	10	3217,77	+1,13%
RCM107	11	3542,4	11	3521,5	11	3525,5	11	3516,5	**11**	**3505,03**	-0,33%
RCM108	11	3614,5	11	3565,2	11	3589,5	11	3566,2	**11**	**3553,54**	-0,33%
RCM201	2	2783,6	2	2783,2	2	2890,1	2	2800,1	**2**	**2695,90**	-3,14%
RCM202	2	2847,1	2	2779,4	2	2838,4	2	2822,9	**2**	**2725,81**	-1,93%
RCM203	2	2721,9	2	2722	2	2814,2	2	2771,7	**2**	**2716,83**	-0,19%
RCM204	2	2726,5	2	2708,5	2	2758,9	2	2716,0	**2**	**2696,79**	-0,43%
RCM205	2	2754,5	2	2754,5	2	2796,5	2	2756,0	**2**	**2718,76**	-1,30%
RCM206	2	2812,7	2	2803,3	2	2822,7	**2**	**2725,0**	2	2748,13	+0,85%
RCM207	3	3764,2	3	3761,5	3	3789,9	3	3757,1	**2**	**2889,80**	-23,08%
RCM208	2	2791,4	2	2742,7	2	2759,4	2	2735,1	**2**	**2730,92**	-0,15%
Melhores:	1/48		10/48		0/48		6/48		31/48		-0,77%

Fonte: Próprio autor

método proposto e comparado ao melhor conhecido anteriormente.

Além da medida de RPD absoluto, também foram analisados individualmente o RPD para cada grupo de instância entre a implementação realizada e os melhores resultados dos outros trabalhos, que podem ser vistos no diagrama de caixa representado pela Figura 10.

É importante ressaltar que atualmente não se tem conhecimento sobre a existência de um único algoritmo capaz de encontrar a melhor solução para todas as instâncias

Figura 10 – Diagrama de Caixa do RPD

Fonte: Próprio autor

analisadas. Portanto, é comum que cada algoritmo encontrado na literatura proveja melhores resultados para instâncias de *benchmark* com determinadas características, em detrimentos de outras. Considerando essa reciproca, para comparar o desempenho do método proposto em determinados cenários com relação aos métodos encontrados na literatura, a Figura 10 apresenta uma comparação do RPD médio e desvio padrão levando em consideração as diferentes características do *benchmark*, que inclui o tipo de distribuição geográfica, o tamanho do horizonte de tempo e o número de janelas de tempo.

Na Figura 10 é possível perceber que diferentes características dos grupos de instâncias acabam afetando diretamente o resultado. As instâncias agrupadas (CM1 e CM2) foram as que o método proposto menos obteve melhoria nas otimizações, inclusive tendo aumento médio de 0.39% no custo das soluções. Já para as instâncias randômicas e randômica agrupadas (RM1, RM2, RCM1 e RCM2) foi possível constatar, na média, uma diminuição no custo das soluções de 1,35%, mesmo que ainda seja possível detectar pioras para algumas instâncias dessas classes. Essas diferenças de resultados podem ser atribuídas aos desafios específicos impostos pelas características de cada classe de instância, conforme relatado na Seção 4.2 Ainda, observando a última caixa, referente às instâncias RCM2, é possível constatar uma grande diminuição no valor mínimo, sem que o valor mediano fosse sensivelmente alterado. Isso ocorreu devido à redução de custo de aproximadamente 23% para a instância RCM207, causada pela redução de 1 veículo obtida pelo método proposto em comparação ao melhor resultado encontrado na literatura.

Para garantir estatisticamente a significância das melhorias observadas com relação aos outros trabalhos comparados, foram aplicados testes de significância estatística de Wilcoxon. A partir da aplicação desses testes é possível refutar a hipótese nula e garantir estatisticamente a confiança e significância dos números obtidos. Ao aplicar o teste de significância estatística de Wilcoxon aos custos calculados pela função objetivo no método proposto e o realizado por Belhaiza, Hansen e Laporte (2014), foi obtido um $p-valor =$

$1,35 \times 10^{-5}$. Ao se comparar com o método proposto por Belhaiza, M'Hallah e Ben Brahim (2017), foi obtido um $p - valor = 1,75 \times 10^{-3}$. Por fim, ao se comparar com o método E-AVNS proposto por Hoogeboom et al. (2020), foi obtido um $p - valor = 1,60 \times 10^{-4}$. Portanto, com base na comparações numéricas e estatísticas com outros trabalhos, é possível afirmar que existe significância estatística suficiente para assegurar que houve uma melhora nos resultados com relação aos demais métodos encontrados na literatura.

4.5 ANÁLISE DO TEMPO DE EXECUÇÃO

Como uma das contribuições deste trabalho é prover uma solução eficiente para a análise das janelas de tempo, esta seção avalia os benefícios alcançados com a aplicação da abordagem (Δ-*Evaluation*) no cálculo incremental das janelas de tempo. O emprego da avaliação incremental (Δ-*Evaluation*) das janelas de tempo visa diminuir a complexidade computacional e aumentar a performance do algoritmo.

A avaliação das contribuições advindas do uso do cálculo (Δ-*Evaluation*) será feita de forma simulada utilizando como base o algoritmo de otimização proposto neste trabalho. Sobre ele serão aplicadas três formas de cálculo das janelas de tempo, sendo elas: i) cálculo completo dos tempos, chamado de T-Full; ii) cálculo incremental dos tempos utilizando (Δ-*Evaluation*) apenas para a análise de factibilidade das janelas de tempo (conforme proposto por Hoogeboom et al. (2020)), chamada T-Incr Hg.; iii) cálculo incremental dos tempos utilizando (Δ-*Evaluation*) não só para a análise de factibilidade, mas também para o cálculo dos tempos da rotas, proposta neste trabalho, chamada de T-Incr Br.

A decisão em utilizar o mesmo algoritmo de otimização como base para a execução dos três experimentos foi tomada para que diferenças intrínsecas à diferentes abordagens de otimização não influenciassem os tempos de execução dos experimentos. Ainda, não foi possível utilizar qualquer resultado de tempo apresentado nos trabalhos relacionados, pois diferenças de hardware, de linguagens de programação, detalhes de implementação, entre outras, inviabilizariam a comparação direta do tempo de execução dos experimentos.

Os três experimentos foram executados no mesmo hardware descrito na Seção 4.1. A bateria de teste foi composta por execuções sequenciais do algoritmo, alternando entre instâncias do problema a cada iteração, visando minimizar variáveis de concorrência e trocas de contextos do sistema. Como o resultado e tempo de execução de uma instância são consistentes, dados os mesmos parâmetros iniciais, a cada execução é fornecida uma semente diferente ao gerador de números aleatórios, visando avaliar diversificados comportamentos do algoritmo. A sequência de testes encerra quando todas as instâncias utilizadas forem executadas 10 vezes para cada uma das implementações.

Como o processo de avaliação das janelas de tempo dentro do algoritmo de otimização ocorre de forma determinística (sem envolver a geração de números aleatórios que influencia a meta-heurística), e que o gerador de números aleatórios utilizado pelo

algoritmo gera uma sequência de números previsível (podendo ser reproduzível dada a mesma semente), é possível afirmar que a alteração da forma de avaliação das janelas de tempo não afeta a solução final gerada pelo algoritmo. Desta forma, o resultado da solução, que foi abordado na Seção 4.4, não será abordado nesta seção, sendo aqui avaliado apenas o tempo de execução.

A Tabela 10 apresenta as médias dos tempos das 10 execuções obtidas para cada instância executada em cada tipo de implementação do cálculo das janelas de tempo. Cada linha da tabela fornece informações sobre os resultados obtidos para uma dada instância. A primeira coluna indica o nome da instância avaliada. A segunda coluna informa a média e desvio padrão de tempo de execução, em segundos, das instâncias considerando o experimento (i), aquele em que a implementação do algoritmo realiza o cálculo completo das janelas de tempo da rota a cada nova solução (coluna T-Full). A terceira coluna informa a média e desvio padrão de tempo de execução, em segundos, das instâncias considerando o experimento (ii), aquele em que a implementação do algoritmo realiza o cálculo incremental das janelas de tempo a cada modificação de rota (coluna T-Incr Hg.), conforme (HOOGEBOOM et al., 2020), e que necessita do recálculo completo das janelas de tempo para efetivar uma nova solução a cada iteração. Em seguida, a quarta coluna, chamada "*Speedup 1*", indica a proporção relativa observada entre tempos de cálculo das soluções de forma completa e de forma incremental com "T-Incr Hg.". A quinta coluna informa a média e desvio padrão de tempo de execução, em segundos, das instâncias considerando o experimento (iii), aquele em que a implementação do algoritmo realiza o cálculo incremental das janelas de tempo a cada modificação de rota proposto neste trabalho (coluna T-Incr Br.). Em seguida, a sexta coluna, chamada "*Speedup 2*", indica a proporção relativa observada entre tempos de cálculo das soluções de forma completa e de forma incremental com "T-Incr Br.". A sétima coluna, chamada "*S. Diff*", apresenta a diferença entre as duas implementações do calculo incremental analisadas. Por fim, a última linha da tabela apresenta a diferença média dos valores de "*Speedup*" entre as implementações.

Na Tabela 10 é possível constatar que, independente do método usado, o tempo de processamento é inversamente proporcional ao número de veículos da solução que está sendo calculada. Isto é, soluções com um número pequeno de veículos, como é o caso das instâncias do grupo RM2 e RCM2, acabam exigindo mais tempo de processamento (aproximadamente 170 segundos), enquanto instâncias que requerem um número maior de veículos, como é o caso das instâncias CM1, acabam exigindo menos tempo de processamento (aproximadamente 30 segundos). Isso se deve ao fato de que rotas mais longas necessitam de menos veículos, o que eleva o número de possibilidades de alocações das múltiplas janelas de tempo, fazendo com que sejam necessários mais cálculos de janelas de tempo. Portanto, mesmo com o uso do Δ-*Evaluation*, o processamento continua sendo computacionalmente mais elevado do que realizar avaliações de rotas menores.

Tabela 10 – Tempo de execução do algoritmo (segundos)

	T-Full		T-Incr (Hg.)		Speedup 1	T-Incr (Br.)		Speedup 2	S. Diff
cm101	39,09	± 4,70	29,74	± 3,81	1,3143	29,64	± 3,70	1,3190	0,0047
cm102	28,49	± 0,64	21,94	± 0,45	1,2985	21,83	± 0,43	1,3052	0,0067
cm103	32,77	± 5,21	24,26	± 3,89	1,3096	24,90	± 3,94	1,3162	0,0066
cm104	26,70	± 6,84	20,38	± 5,62	1,3101	20,26	± 5,22	1,3176	0,0075
cm105	27,29	± 0,82	20,86	± 0,67	1,3081	20,74	± 0,78	1,3157	0,0076
cm106	28,35	± 0,50	21,55	± 0,62	1,3152	21,51	± 0,60	1,3183	0,0031
cm107	24,36	± 0,71	18,54	± 0,81	1,3134	18,43	± 0,93	1,3217	0,0083
cm108	25,77	± 0,48	19,26	± 0,75	1,3376	19,14	± 0,69	1,3462	0,0086
cm201	74,34	± 1,01	56,05	± 1,54	1,3263	55,81	± 1,72	1,3321	0,0058
cm202	54,48	± 0,72	41,42	± 1,23	1,3155	40,81	± 1,41	1,3350	0,0195
cm203	62,88	± 1,28	47,81	± 2,16	1,3152	47,53	± 2,11	1,3230	0,0078
cm204	57,95	± 1,17	43,75	± 1,43	1,3287	43,53	± 1,46	1,3314	0,0027
cm205	72,19	± 1,57	55,82	± 2,56	1,2934	55,48	± 2,14	1,3012	0,0078
cm206	69,66	± 1,29	53,15	± 1,70	1,3113	52,84	± 1,75	1,3183	0,0007
cm207	61,66	± 1,29	47,67	± 1,41	1,2938	47,30	± 1,23	1,3036	0,0098
cm208	56,69	± 0,29	43,64	± 0,94	1,2991	43,43	± 0,90	1,3055	0,0064
rcm101	35,40	± 2,43	27,27	± 1,36	1,2984	27,11	± 1,86	1,3055	0,0071
rcm102	32,57	± 5,35	25,20	± 4,12	1,2920	24,90	± 3,97	1,3081	0,0161
rcm103	33,88	± 8,36	26,93	± 6,47	1,3035	25,81	± 6,55	1,3124	0,0089
rcm104	33,23	± 5,21	25,59	± 3,78	1,2988	25,43	± 3,79	1,3067	0,0079
rcm105	30,26	± 5,08	23,27	± 3,56	1,3005	23,07	± 3,74	1,3115	0,0110
rcm106	25,88	± 1,71	19,95	± 1,85	1,2973	19,81	± 1,58	1,3067	0,0094
rcm107	22,25	± 0,44	16,97	± 0,36	1,3111	16,80	± 0,37	1,3240	0,0129
rcm108	27,40	± 5,77	20,72	± 4,73	1,3226	20,58	± 4,40	1,3314	0,0088
rcm201	253,91	± 22,88	226,40	± 17,90	1,1215	224,94	± 18,21	1,1286	0,0071
rcm202	220,08	± 40,65	194,78	± 40,63	1,1299	194,26	± 38,87	1,1329	0,0030
rcm203	205,78	± 5,95	182,35	± 5,93	1,1285	181,82	± 5,84	1,1317	0,0032
rcm204	176,08	± 2,56	156,54	± 2,89	1,1296	155,35	± 2,85	1,1335	0,0039
rcm205	161,02	± 3,38	142,84	± 3,53	1,1273	141,84	± 3,47	1,1352	0,0079
rcm206	153,64	± 3,33	134,75	± 3,47	1,1253	136,05	± 3,36	1,1293	0,0040
rcm207	81,59	± 26,44	66,84	± 26,35	1,2272	66,24	± 26,05	1,2318	0,0046
rcm208	131,83	± 2,43	115,71	± 1,22	1,1393	115,51	± 1,70	1,1413	0,0020
rm101	45,51	± 11,25	34,78	± 9,14	1,3085	34,68	± 8,52	1,3122	0,0037
rm102	34,52	± 0,48	26,34	± 0,38	1,3104	26,24	± 0,43	1,3155	0,0051
rm103	38,75	± 8,11	30,00	± 6,01	1,2917	29,88	± 5,96	1,2966	0,0049
rm104	35,58	± 4,70	26,99	± 3,55	1,3182	26,91	± 3,60	1,3219	0,0037
rm105	28,70	± 2,64	22,08	± 1,80	1,2997	22,00	± 1,84	1,3044	0,0047
rm106	27,02	± 3,08	20,22	± 2,57	1,3361	20,17	± 2,45	1,3397	0,0036
rm107	23,23	± 2,20	17,56	± 1,78	1,3222	18,50	± 1,58	1,3277	0,0055
rm108	22,66	± 0,40	16,21	± 0,29	1,3940	16,11	± 0,30	1,4064	0,0124
rm201	269,71	± 53,69	238,85	± 53,76	1,1292	238,40	± 50,93	1,1313	0,0021
rm202	208,42	± 3,14	184,00	± 3,36	1,1266	184,22	± 3,45	1,1314	0,0048
rm203	181,87	± 3,27	160,22	± 3,32	1,1351	159,80	± 3,40	1,1381	0,0030
rm204	163,13	± 3,42	142,90	± 2,25	1,1416	142,75	± 2,41	1,1428	0,0012
rm205	149,24	± 4,91	131,74	± 4,06	1,1320	131,28	± 3,96	1,1368	0,0048
rm206	144,13	± 3,06	127,85	± 3,77	1,1273	127,48	± 3,57	1,1306	0,0033
rm207	137,97	± 2,10	121,51	± 2,42	1,1354	121,25	± 2,87	1,1378	0,0024
rm208	129,56	± 1,98	113,16	± 3,23	1,1449	112,66	± 3,09	1,1500	0,0051
Média					1,2539			1,2605	0,0066

Fonte: Próprio autor

Com relação à diminuição do tempo de execução do cálculo das janelas de tempo proporcionado pelo algoritmo proposto (T-Incr Br.) em relação ao cálculo completo (T-Full), foi possível constatar redução de tempo na utilização do cálculo incremental para todas as instâncias avaliadas. Entretanto, nem todos os grupos de instâncias obtiveram o mesmo nível de ganho, pois o *speedup* varia entre cerda de 1,12 a 1,40. Analisando a Tabela 10, é possível constatar que os maiores ganhos proporcionais de velocidade (as instâncias que mais diminuíram o tempo de execução) foram nas instâncias com rotas

menores e menos rotas, chegando em média a 1,32 de aceleração para as instâncias do tipo C, enquanto que para as instâncias RM2 e RCM2 foi constatada uma aceleração média de 1,15.

De forma geral, foi possível constatar um fator de aceleração nos cálculos de aproximadamente 1,2539 para as instâncias ao utilizar a estratégia proposta por (HOOGEBOOM et al., 2020) (T-Incr Hg.), e de 1,2605 ao utilizar a estratégia proposta neste trabalho (T-Incr Br.). Embora esta diferença média de 0.0066 seja pequena, representando menos de 1% de diferença na média, ela pode ser explicada pelo fato de que as duas estratégias foram implementadas sobre o mesmo algoritmo base, e também elas não possuem grandes diferenças de abordagem para o cálculo das janelas de tempo. Portanto, a mudança mais significativa é que o método proposto neste trabalho dispensa o cálculo extra dos tempos das rotas na modificação de uma rota pelos operadores. Assim, ele economiza algumas avaliações de custo da rota.

Embora tenha sido notado uma redução significativa no tempo de execução do algoritmo ao adotar o cálculo incremental, de acordo com resultados preliminares, era esperado um ganho de performance consideravelmente maior. Contudo, no decorrer do desenvolvimento deste trabalho acabaram sendo realizadas algumas mudanças estruturais no método de otimização, para melhorar a qualidade da solução final, que acabaram prejudicando a diminuição do tempo de execução do algoritmo. Uma das adaptações realizadas foi a inclusão de um segundo algoritmo ALNS utilizado na redução de veículos. Isso fez com que fosse necessário realizar mudanças no modo de funcionamento do Δ-*Evaluation*, fazendo com que a cada iteração seja necessário realizar o cálculo completo da avaliação da solução para a minimização do veículo. Esse cálculo completo é necessário para o cálculo das penalidades por quebra de restrições da solução. Entretanto, esse cálculo pode ser otimizado, mas ainda não foi feito devido à limitação de tempo de desenvolvimento do trabalho.

Por fim, é importante observar os desvios padrão dos tempos de execução. Para cada linha representada na tabela, a coleta de tempos foi feita considerando a execução do algoritmo com os mesmos parâmetros aplicadas às mesmas instâncias, com a exceção da semente provida ao gerador de números aleatórios do algoritmo. Essa escolha de utilização de sementes distintas foi feita para que fosse realmente possível avaliar o tempo médio de execução do algoritmo para a instância, independente da semente selecionada. Portanto, embora seja o mesmo algoritmo executado várias vezes, as iterações foram executadas de forma diferente (seleção de operadores diferentes, trabalhando com soluções diferentes, entre outras variáveis), o que pode afetar diretamente o tempo de execução do algoritmo. Sendo assim, é possível constatar uma correlação entre o desvio padrão da coluna “T-Full” e da coluna “T-Incr Hg.”, e “T-Incr Br.”, haja vista que as mesmas sementes foram utilizadas em ambas as execuções, de forma que as soluções avaliadas são idênticas.

4.6 ANÁLISE DO ALGORITMO

Além da análise quantitativa apresentada nas Seções 4.4 e 4.5, também foi realizada uma análise qualitativa do comportamento do algoritmo durante a execução da otimização. Portanto, esta seção avalia as características dos diferentes cenários presentes nas instâncias de *benchmark* e como os parâmetros do algoritmo afetam seu comportamento.

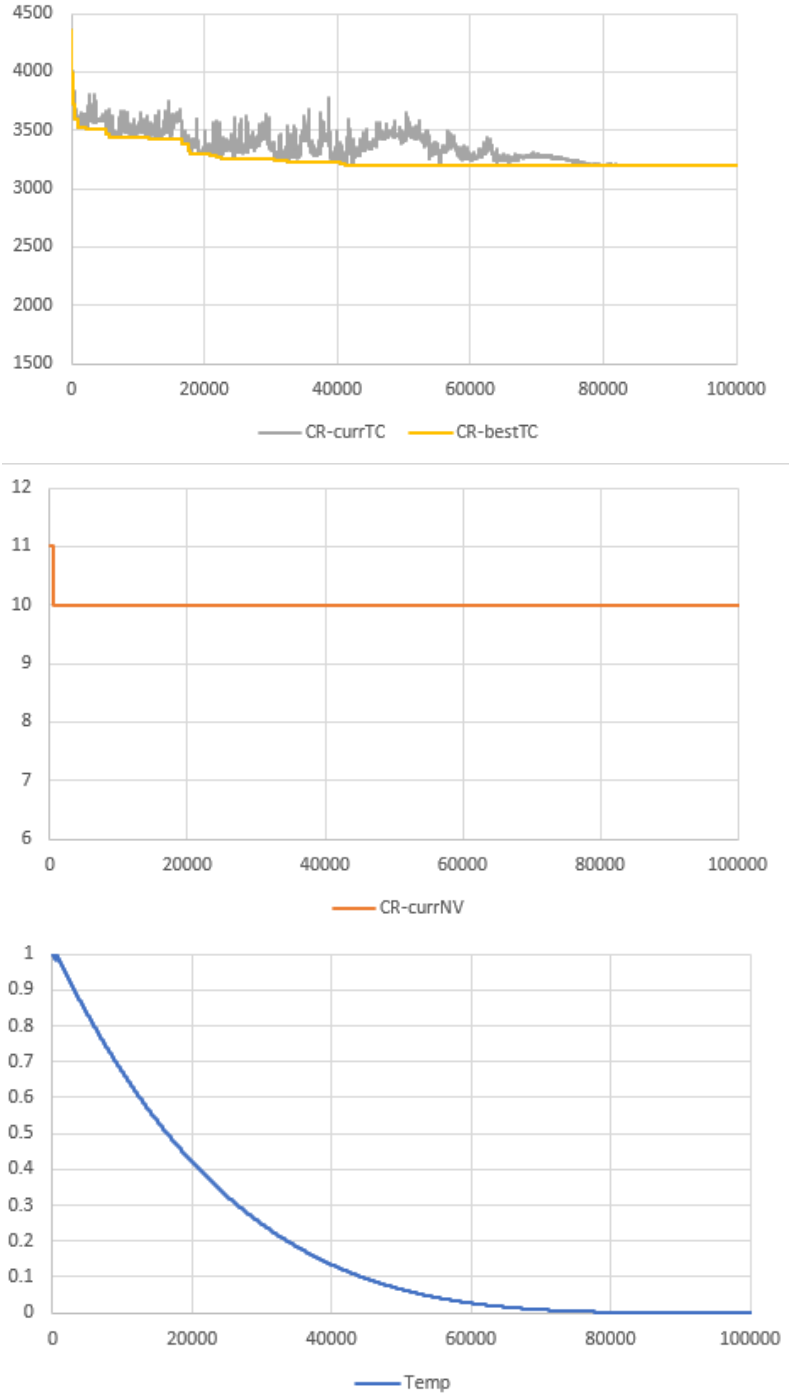
Para operacionalizar esta análise, durante cada iteração do algoritmo foram coletadas informações, tais como: o identificador da execução, o número da iteração corrente, o numero máximo de iterações, o tempo de execução da iteração, o tempo acumulado de execução, o número de soluções distintas avaliadas, a temperatura do sistema, o custo da solução corrente, o custo da melhor solução até o momento, a pontuação dos operadores, os operadores selecionados, as estatísticas de utilização de memória, entre outros. A partir da tabulação e análise dessas informações foi possível verificar as ações do algoritmo a cada iteração e gerar gráficos para facilitar a interpretação de sua operação.

Para realizar uma análise minuciosa sobre o comportamento do método proposto, são detalhados os resultados obtidos na execução do algoritmo em instâncias específicas de *benchmark*. Mais especificamente, foram selecionadas uma instância com resultado mediano para representar o típico comportamento médio do algorítimo, a instância com melhor desempenho e a com pior desempenho nos resultados apresentados na Seção 4.4. O intuito dessa seleção é analisar as vantagens e carências do método proposto. Portanto, são analisadas a instância CM105, pertencente ao grupo mais mediano de resultados comparado com a literatura, a instancia RCM207 que obteve o melhor resultado em comparação à literatura, incluindo redução do número de veículos, e também a instância CM102 que foi a instância com pior resultado obtido pelo método proposto, ou seja, a que ele teve maior dificuldade em ser resolvida.

As Figuras 11, 12 e 13 ilustram graficamente as curvas de convergência do custo da solução (linha cinza para a solução corrente e linha amarela para a melhor solução descoberta pelo algoritmo até o momento), número de veículos (linha laranja) e temperatura do sistema (linha azul) de uma execução do algoritmo (eixo vertical) em função do número de iterações (eixo horizontal) para as instâncias CM105, CM102 e RM107, respectivamente. Nelas é possível observar observar os diferentes comportamentos do algoritmo dependendo das características da instância utilizada.

Na Figura 11, com relação à curva de convergência da minimização do custo da solução, é possível constatar o típico comportamento do algoritmo. Ela, a heurística de construção inicial produz uma solução factível razoavelmente boa, que rapidamente é melhorada nas primeiras iterações do algoritmo. Com o passar das iterações, devido à alta temperatura do sistema, o valor da solução corrente oscila entre melhoras e pioras, conforme representado pela linha cinza. Entretanto, com o passar das iterações ela tende a se estabilizar, diminuindo a amplitude das oscilações e convergindo para uma solução de

Figura 11 – CM105 - Medio



Fonte: Próprio autor

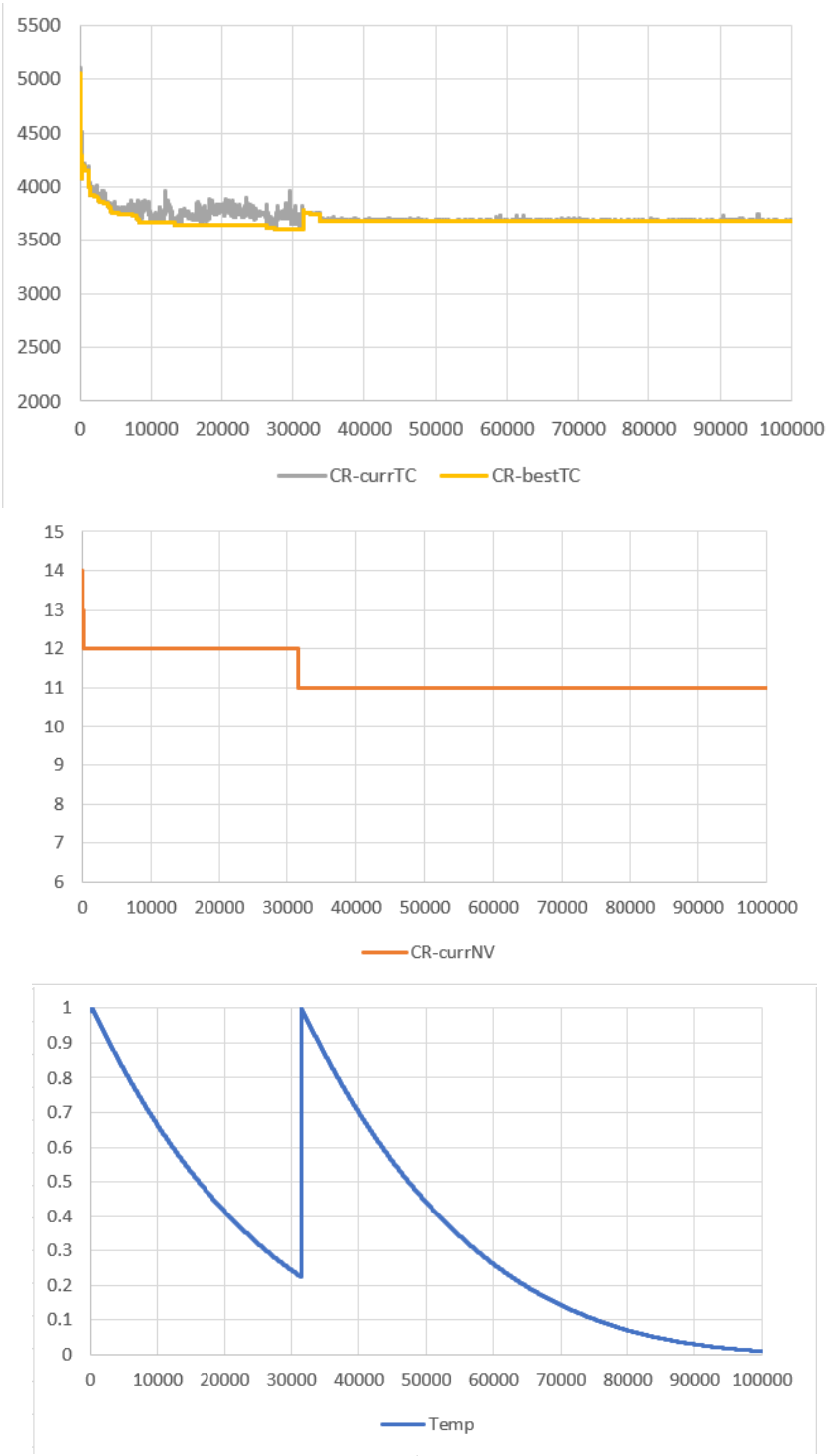
menor custo. Dessa forma, ao final do processo a solução trabalhada tente a coincidir com a melhor solução conhecida até o momento (linha amarela).

Ainda, analisando a Figura 11, mas agora com relação à curva de convergência do número de veículos, é possível perceber que a heurística de construção da solução inicial chegou a uma solução razoavelmente boa, com apenas um veículo a mais do que a melhor solução conhecida. Na sequência, o algoritmo logo em suas primeiras iterações foi capaz

de reduzir o número de veículos utilizados na solução.

Por fim, analisando a Figura 11, agora com relação à curva de temperatura do algoritmo, é possível perceber seu decrescimento típico. Ou seja, o sistema inicia com a temperatura máxima que decai conforme o passar das iterações. Isso permite uma maior exploração do espaço de busca no início do processo de otimização e uma maior intensificação da solução ao final do processo de otimização.

Figura 12 – CM102 - Pior



Fonte: Próprio autor

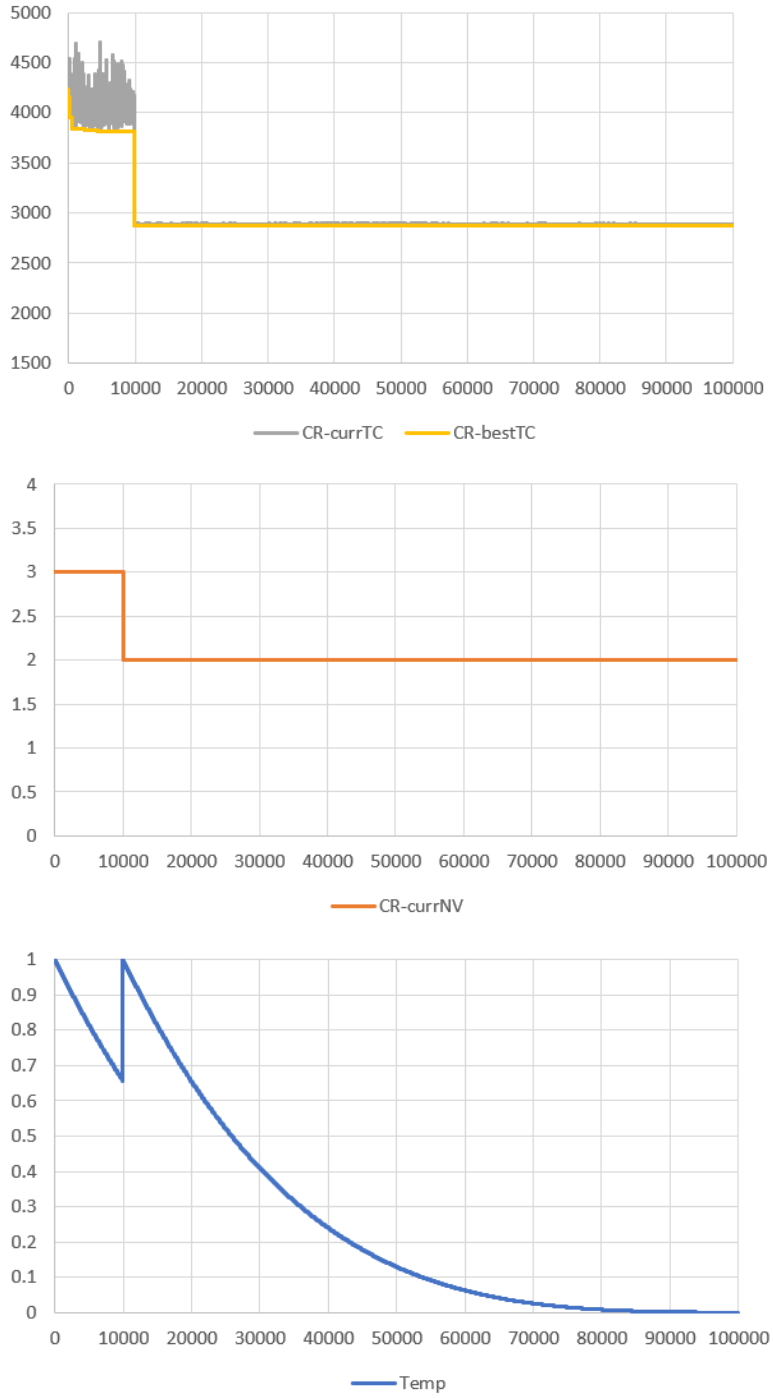
Analisando a Figura 12, com relação à curva de convergência da minimização do custo da solução, é possível constatar o típico comportamento esperado do algoritmo em seu início. Nela a heurística de construção inicial produz uma solução factível que rapidamente é melhorada nas primeiras iterações do algoritmo. Porém, a taxa de melhoramento acaba diminuindo com o passar das iterações. Contudo, próximo à iteração de número 30000 o minimizador de número de veículos foi capaz de encontrar uma nova solução factível com um número reduzido de veículos, mas com custo maior do que a solução atual. Este aumento de custo da solução, inclusive com relação à melhor solução global, é aceitável e desejável devido à natureza hierárquica da função objetivo, que sempre privilegia soluções com o menor número de veículos possível. Este fato ajuda a explicar a dificuldade de melhoramento desta instância, dado que a minimização do número de veículos e do custo da solução devem se encontrar distantes uma da outra no espaço de busca, dado que a minimização do número de veículos não acarretou na diminuição do custo da rota, para este caso. Além disso, também é importante notar a saturação da nova solução adotada pelo minimizador de custos, pois mesmo com o reaquecimento da temperatura do sistema, poucas soluções com custo maior são aceitas. Isso é ocasionado pela dificuldade em encontrar outras soluções competitivamente comparáveis na sua vizinhança.

Ainda analisando a Figura 12, com relação à curva de convergência do número de veículos, é possível perceber que a heurística de construção da solução inicial chegou a uma solução com um número elevado de veículos. Portanto, já no início da execução do algoritmo o número de veículos foi reduzido de 14 para 12 nas primeiras iterações. Além disso, próximo à iteração 30000 foi encontrada uma nova solução que minimizava o número de veículos. Entretanto, essa redução de veículo não foi trivial, pois necessitou de diversas iterações e da redução da temperatura do sistema à cerca de 20% da temperatura inicial para que fosse encontrada.

Por fim, analisando a Figura 12 com relação à curva de temperatura do algoritmo, é possível perceber seu comportamento de decaimento. Entretanto, perto da iteração de número 30000 foi realizado o reaquecimento da temperatura causada pela descoberta de uma nova solução com um número menor de veículos. Portanto, isso permitiu que o algoritmo pudesse explorar novamente o espaço de busca.

Analisando a Figura 13, com relação à curva de convergência da minimização do custo da solução, é possível constatar que a solução inicial é rapidamente melhorada nas primeiras iterações do algoritmo, apresentando pouca melhora na melhor solução encontrada entre as iterações 1000 e 10000, enquanto o algoritmo continua explorando o espaço de busca. Próximo à iteração de número 10000 o minimizador de número de veículos foi capaz de encontrar uma nova solução factível com um número reduzido de veículos que também reduziu consideravelmente o custo da solução. Após este momento foi possível notar a saturação da nova solução adotada pelo minimizador de custos. Portanto, embora a temperatura do sistema tenha sido reaquecida, poucas soluções com custo maior

Figura 13 – RCM207 - Melhor



Fonte: Próprio autor

foram aceitas. Isso reforça a dificuldade de encontrar outras soluções competitivamente comparáveis na sua vizinhança, fazendo com que o custo da solução atual se encontre bem próximo da melhor solução global.

Ainda, analisando a Figura 13 com relação à curva de convergência do número de veículos, é possível perceber que a heurística de construção da solução inicial chegou a uma solução com um número de veículos reduzido, e que a meta-heurística não foi capaz

de trivialmente reduzi-lo. Além disso, próximo à iteração 10000 foi encontrada uma nova solução que minimizava o número de veículos quando a temperatura do sistema ainda se encontrava relativamente alta, com cerca de 70% da temperatura inicial.

Por fim, analisando a Figura 13 com relação à curva de temperatura do algoritmo, é possível perceber seu comportamento de decaimento, onde perto da iteração de número 10000 foi realizado o reaquecimento da temperatura causada pela descoberta de uma nova solução com um número menor de veículos. Portanto, isso permitiu ao algoritmo explorar novamente o espaço de busca.

4.7 CONSIDERAÇÕES SOBRE A ANÁLISE DOS RESULTADOS

Este capítulo relatou os experimentos realizados para validação do método proposto e sua respectiva implementação. O primeiro passo para realizar a análise foi a elaboração do protocolo de experimentação, destacando os tipos de análise a serem realizadas e os procedimentos para sua execução, coleta de resultados e posterior avaliação dos resultados.

As principais contribuições pretendidas a partir do método proposto era gerar um algoritmo baseado na meta-heurística ALNS e no cálculo incremental (Δ-*Evaluation*) das janelas de tempo capaz de ser competitivo com outros métodos existentes na literatura, tanto em relação à qualidade das soluções geradas, quanto no desempenho computacional do algoritmo.

Para avaliar a qualidade das soluções geradas, foi realizada a execução do algoritmo implementado sobre o método proposto em instâncias de *benchmark* para o VRPMTW proposto por Belhaiza, Hansen e Laporte (2014), que é um *dataset* bem difundido para avaliação deste tipo de problema. De posse dos dados para teste e com a metodologia de avaliação definida, foram coletados os resultados obtidos a partir da execução do modelo proposto e comparados com os resultados obtidos por outros trabalhos na literatura, nominalmente Belhaiza, Hansen e Laporte (2014), Belhaiza, M'Hallah e Ben Brahim (2017) e Hoogeboom et al. (2020).

Durante a análise dos resultados, foi possível perceber que em parte significativa das instâncias do *benchmark* a implementação do algoritmo proposta foi capaz de obter melhores soluções que as alcançadas pelos trabalhos da literatura. De fato, a implementação proposta foi capaz de encontrar melhores soluções para 31 das 48 instâncias avaliadas, proporcionando, em média, redução de 0,93% do custo da função objetivo, sem aumentar em nenhuma instância o número de veículos necessários. Para validar a relevância estatística do resultado foram aplicados testes de significância estatísticos de Wilcoxon comparando o resultado do trabalho proposto com os outros existentes na literatura. Portanto, pela análise estatística foi possível observar resultados significativamente melhores e comprovar a superioridade do algoritmo proposto.

Com relação a execução da meta-heurística de melhoramento, por meio da análise

do gráfico da curva de convergência, foi possível verificar um comportamento semelhante ao esperado para esse tipo de meta-heurística, tanto com relação ao decaimento da temperatura do sistema, quanto com relação a evolução da solução sendo otimizada. Foi possível constatar a importância do correto controle de temperatura durante a execução do algoritmo. Isso permitiu uma busca com comportamento mais exploratório no início da execução e mais intensiva ao final do processo de otimização.

Ainda, foi possível verificar a eficiência da otimização simultânea de duas soluções. Enquanto um ALNS tenta minimizar exclusivamente o número de veículos de uma solução infactível e assim obter uma solução factível, o outro ALNS foca na minimização do custo total da solução factível com o menor número de veículos geradas pelo primeiro ALNS. Portanto, o segundo ALNS necessariamente utiliza uma solução factível e tenta continuamente melhorar essa solução buscando reduzir o valor da função objetivo, sem foco direto na redução do número de veículos. Essa abordagem foi essencial para evitar conflitos de interesse entre objetivos possivelmente conflitantes. Isso é explicado porque a redução do número de veículos não é necessariamente acompanhada de uma rota com duração menor, tendo em vista que uma rota com um número reduzido de veículos pode acabar saturando a disponibilidade e aumentar o custo das rotas. Desta forma, foi possível especializar o funcionamento de cada um dos mecanismos de forma a tornar a redução de seu objetivo específico mais efetivo.

De forma geral, foi possível perceber que o algoritmo consegue obter bons resultados para a maioria dos cenários. Isso pode ser justificado pelo fato do algoritmo proposto possuir um número considerável de operadores, de forma que estes podem ser utilizados para lidar com as diversas características do problema. Ainda, esse bom desempenho também está associado ao mecanismo adaptativo de seleção de operadores, capaz de priorizar os operadores que agregam mais benefícios à solução dependendo do cenário sendo otimizado.

Contudo, ainda é possível constatar que o algoritmo apresenta oportunidades de melhoras. Esse fato pode ser constatado na Figura 10, em que é possível verificar que o algoritmo não apresenta desempenho tão atrativo para os casos de instâncias com horizonte de tempo comprido. Isso pode ser justificado pelo fato de que embora o horizonte de tempo seja estendido, o valor de variáveis que são influenciadas pelo tempo, como penalidade por quebra de restrição temporal, por exemplo, apresentam valores fixos, ou seja, não são proporcionais ao contexto do cenário que está sendo otimizado.

Além disso, alguns parâmetros do algoritmo são compartilhados entre os dois ALNS que estão contidos no método proposto. Esses parâmetros compartilhados são: os operadores, os pesos dos operadores e a temperatura do sistema, que são aplicados tanto para a modificação e avaliação da solução minimizada por veículo e da solução minimizada por custo. Como cada ALNS tem um objetivo específico e independente, é de se esperar que uma escolha especializada de operadores para cada uma das situações poderia levar a

obtenção de resultados melhores e mais cedo pelo algoritmo.

Durante os testes, os valores utilizados para os parâmetros não passaram por um processo formal de ajuste. Portanto, isso significa que foram adotados valores de referência na literatura (que podem acabar não sendo a melhor alterativa para o método proposto), ou então utilizados valores definidos empiricamente por meio de amostragem. Embora o ALNS seja um algoritmo adaptativo capaz de definir automaticamente alguns valores para determinados parâmetros em tempo de execução de acordo com suas constatações, o próprio mecanismo adaptativo do algoritmo também implica na definição manual prévia de parâmetros. Portanto, com a aplicação de um processos adequado e efetivo de ajustes de parâmetros assume-se que seria possível melhorar a qualidade dos resultados.

Com relação ao desempenho do algoritmo levando em consideração o método Δ-*Evaluation*, embora agregue bastante complexidade no desenvolvimento do algoritmo, ele permitiu ganhos significativos de desempenho com relação ao tempo de execução do algoritmo, sem comprometer a qualidade do resultado, reduzindo em cerca de um quarto o tempo de execução do algoritmo. Contudo, é possível dizer que durante o desenvolvimento do trabalho o aspecto de otimização do tempo de execução acabou sendo penalizado em favor da melhora da minimização do custo da solução. Isso porque a avaliação de soluções infactíveis, necessária no processo para alcançar a minimização de veículos, acabou gerando entraves que prejudicaram a aplicação do método Δ-*Evaluation* nestes contextos. Portanto, este aspecto acabou penalizando significativamente a redução de tempo obtida.

5 CONCLUSÃO

Dentro da área de operações logísticas, o problema de roteamento de veículos, denominado VRP pela literatura, é um dos mais relevantes, pois um bom planejamento das rotas dos veículos é fundamental para a diminuição dos custos da operação e aumento da competitividade de empresas de transporte ou que precisem escoar sua produção. O problema de roteamento de veículos com suporte a múltiplas janelas de tempo, comumente chamado de VRPMTW, permite expandir o contexto de restrições do VRP e atender diversas necessidades de empresas da área logística, desde transportes de longa duração com horizontes compreendendo vários dias de viagem, a operações de curta duração onde o cliente possui vários possíveis períodos para realização da entrega no mesmo dia.

Embora parte significativas dos trabalhos encontrados na literatura que abordam o VRP lidam com janelas de tempo, relativamente poucos trabalhos tratam múltiplas janelas de tempo para um mesmo serviço. Portanto, o problema VRPMTW endereçado neste trabalho tem como função objetivo a minimização hierárquica de duas variáveis: i) o número de veículos e ii) o custo total de viagem dos veículos, incluindo tempo de viagem. Ainda, foram definidas as seguintes restrições a serem respeitadas: limite de capacidade de carga máxima dos veículos, limite da quantidade veículos da frota, múltiplas janelas de tempo de chegada ao cliente e tempo de serviço no cliente.

O VRPMTW é a associação de dois problemas de otimização combinatória: 1º) o problema de roteamento de veículos e 2º) o problema de agendamento com múltiplas janelas de tempo, onde o segundo, geralmente, é tratado como um subproblema do primeiro. Para resolver esse problema foi proposto um método de otimização baseado na meta-heurística ALNS para geração das sequências de atendimentos de serviço, associado a programação dinâmica utilizando técnica de cálculo incremental (Δ-*Evaluation*) para avaliação rápida das modificações feitas nas rotas, calculando os tempos de atendimento dos serviços e minimização do tempo de viagem.

Para tratar adequadamente a otimização dos dois objetivos a serem otimizadas (número de veículos e custo total da solução), que são possivelmente conflitantes, o método proposto se baseia em um algoritmo ALNS que otimiza duas soluções simultaneamente de forma especializada para cada objetivo específico, uma dedicada a minimização dos veículos e outra dedicada a minimização do custo total. Essa abordagem fez com que a otimização das instâncias *benchmark* utilizadas tivessem um resultado melhor do que se a minimização das duas variáveis fosse feita de forma conjunta. Essa situação foi constatada em testes preliminares durante o desenvolvimento do algoritmo que implementa o método proposto.

A avaliação dos resultados foi realizada por meio da aplicação de avaliações quantitativas e qualitativas. As avaliações quantitativas tiveram como objetivo avaliar, primeiro, se o algoritmo que implementa o método proposto é capaz de gerar soluções melhores

que as obtidas pelos trabalhos mais relevantes encontrados na literatura e, segundo, se a implementação feita neste trabalho do cálculo incremental (Δ-*Evaluation*) para as janelas de tempo tinha tempo de execução menor que as implementações feitas pelos trabalhos relacionados. Por fim, a avaliação qualitativa tinha por objetivo analisar o comportamento do método por meio da análise minuciosa dos resultados alcançados em algumas instâncias do *benchmark* de testes.

Sobre a análise quantitativa de desempenho na geração de melhores soluções, foi possível constatar que o método proposto obteve 31 melhores soluções no conjunto de 48 instâncias testadas, sendo que o segundo trabalho melhor colocado (trabalho de Belhaiza, M'Hallah e Ben Brahim (2017)) obteve apenas 10 melhores resultados. Esses resultados foram obtidos com significância estatística confirmada pela aplicação do teste estatístico de Wilcoxon. Ainda, é importante destacar que o método desenvolvido no trabalho em tela teve uma melhora de 0,93% na média dos resultados quando comparadas todas as instâncias do *benchmark* utilizado nos testes. Entretanto, foi possível notar que o método não teve bons resultados para instâncias com distribuição geográfica randômicas e agrupadas com horizonte de tempo de operação curto.

Sobre a análise quantitativa do tempo de processamento do cálculo incremental (Δ-*Evaluation*) para as janelas de tempo, foi possível constatar que a utilização da técnica de programação dinâmica para o cálculo dos tempos obteve um desempenho consideravelmente melhor do que quando o cálculo dos tempos é feito de forma completa, obtendo, em média, uma diminuição de 26,05% no tempo necessário para o cálculo dos tempos. Entretanto, quanto comparada com a técnica de cálculo incremental proposta na literatura, a forma de cálculo implementada no trabalho em tela obteve uma diminuição no tempo, em média, de apenas 0,66%. Esse valor poderia ter sido maior se não tivesse sido implementados vários cálculos extras necessários para a política de penalização das soluções infactíveis geradas pelo ALNS de minimização de veículos.

Sobre a análise qualitativa, foi possível perceber que na maioria dos casos os ALNSs contidos no método se tiveram um comportamento conforme o esperado. Ou seja, a solução corrente tinha uma alta variabilidade de custo durante a fase inicial, em que a temperatura do sistema estava mais quente, enquanto que essa variabilidade diminuía sensivelmente a medida que o sistemas se resfriava no decorrer das iterações. Entretanto, em alguns casos foi possível perceber que quanto o número de veículos diminuía, mesmo com o reaquecimento do sistema, as soluções não voltaram a ter uma alta variabilidade. Em vários casos e diminuição do número de veículos já era suficiente para fazer com que a solução calculada fosse melhor que a encontrada na literatura. Entretanto, para outros casos a diminuição do número de veículos acabou gerando uma solução atratora local que restringiu a área de busca do algoritmo, fazendo com que não conseguisse alcançar o custo mínimo conhecido, como, por exemplo, nas instâncias agrupadas e randômicas de horizonte de tempo curto.

Portanto, com base nas análises realizadas sobre o método proposto, é possível observar que ele apresenta resultados satisfatório na otimização de instâncias do VRPMTW e que a aplicação de uma técnica de programação dinâmica reduz significativamente o tempo necessário para o cálculo e análise das múltiplas janelas de tempo. Em comparação com outras abordagens encontradas na literatura, pode-se dizer que o trabalho em tela apresenta resultados promissores.

5.1 TRABALHOS FUTUROS

Com base dos resultados obtidos e nas experiências adquiridas durante o desenvolvimento deste trabalho, é possível elencar possibilidades de melhorias ou avanços nos seguintes trabalhos futuros:

- **Melhorar o desempenho do algoritmo nas instâncias com horizonte de tempo curtos e posições agrupadas e randomizadas**: Uma das possíveis soluções para essa melhoria é a desvinculação dos parâmetros utilizados na minimização das duas soluções. Hoje, tanto a minimização de veículos quanto a minimização de custo utilizam os mesmos operadores, pesos dos operadores e temperatura do sistema. Com a utilização isolada de parâmetros para cada ALNS, é possível que o método consiga se especializar mais para cada uma das soluções e alcançar melhores resultados.

- **Permitir que o método consiga manipular soluções infactíveis para reduzir o custo total da solução**: Outra melhoria relacionada à obtenção de menores custos está relacionada a implementação de métodos de manipulação de soluções infactíveis também no ALNS que faz a redução de custo. A implementação pode ser baseada na implementação já feita no ALNS de redução de veículos.

- **Aprimorar a forma de cálculo do Δ-*Evaluation* para soluções infactíveis**: A forma de cálculo do Δ-*Evaluation* implementada tem foco em otimização de soluções factíveis, de forma que para as situações em que a solução é infactível necessita que o cálculo dos tempos tenha que ser calculado completamente. Isso é necessário para poder estimar a penalidade aplicada a solução. Esse cálculo pode ser aprimorado e ter seu tempo reduzido se essas informações sobre a penalidade também fossem calculadas utilizando programação dinâmica.

- **Aprimorar a parametrização das variáveis do método proposto**: Essa melhoria visa melhorar os valores de parâmetros utilizados e também reduzir a complexidade de parametrização do algoritmo, visto que o ALNS por si contém muitos parâmetros. Para isto é necessário, por exemplo, a incorporação de técnicas *online*

de auto-parametrização em que o próprio algoritmo ajusta seus parâmetros durante a execução, ou também técnicas *offline* robustas (em contraste à abordagem empírica deste trabalho) para definição de valores de parâmetros fixos do algoritmo.

- **Frota heterogênea e *pickup-delivery***: Os principais avanços sugeridos para o trabalho em tela compreende a compatibilização do método para lidar com outras variantes do do VRP além do VRPMTW, tais como: i) a utilização de frota heterogênea e ii) a realização de coletas e entregas *pickup-delivery*. Essas alterações necessitariam de adaptações significativas no método proposto, o que implicam na reestruturação de grande parte da abordagem proposta neste trabalho.

REFERÊNCIAS

ADEWUMI, Aderemi Oluyinka; ADELEKE, Olawale Joshua. A survey of recent advances in vehicle routing problems. **International Journal of System Assurance Engineering and Management**, Springer Nature, v. 9, n. 1, p. 155–172, jun. 2016. Disponível em: <https://doi.org/10.1007/s13198-016-0493-4>. Citado na página 24.

AFIFI, Sohaib; DANG, Duc Cuong; MOUKRIM, Aziz. A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [s.n.], 2013. v. 7997 LNCS, p. 259–265. ISBN 9783642449727. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-642-44973-4_27>. Citado na página 50.

AGHEZZAF, Brahim; FAHIM, Hassan EL. The multi-constraint team orienteering problem with time windows in the context of distribution problems: A variable neighborhood search algorithm. In: **2014 International Conference on Logistics Operations Management**. [S.l.: s.n.], 2014. p. 155–160. Citado 2 vezes nas páginas 44 e 47.

BEHESHTI, Ali Kourank; HEJAZI, Seyed Reza; ALINAGHIAN, Mehdi. The vehicle routing problem with multiple prioritized time windows: A case study. **Computers & Industrial Engineering**, v. 90, p. 402 – 413, 2015. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835215003988>. Citado na página 18.

BELHAIZA, Slim. **Homepage of Slim Belhaiza**. 2014. Acesso: 2020-12-05. Disponível em: <https://faculty.kfupm.edu.sa/MATH/slimb/>. Citado na página 85.

Belhaiza, S. A game theoretic approach for the real-life multiple-criterion vehicle routing problem with multiple time windows. **IEEE Systems Journal**, v. 12, n. 2, p. 1251–1262, 2018. Citado 5 vezes nas páginas 18, 45, 52, 54 e 58.

BELHAIZA, Slim et al. Variable neighborhood search for vehicle routing problem with multiple time windows. **Electronic Notes in Discrete Mathematics**, v. 66, p. 207 – 214, 2018. ISSN 1571-0653. 5th International Conference on Variable Neighborhood Search. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1571065318300738>. Citado 6 vezes nas páginas 17, 18, 33, 34, 55 e 58.

BELHAIZA, Slim; HANSEN, Pierre; LAPORTE, Gilbert. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. **Computers & Operations Research**, v. 52, p. 269 – 281, 2014. ISSN 0305-0548. Recent advances in Variable neighborhood search. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054813002165>. Citado 20 vezes nas páginas 10, 17, 18, 20, 26, 34, 35, 45, 54, 55, 58, 80, 81, 83, 84, 88, 89, 90, 91 e 101.

Belhaiza, S.; M'Hallah, R. A pareto non-dominated solution approach for the vehicle routing problem with multiple time windows. In: **2016 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.: s.n.], 2016. p. 3515–3524. Citado 4 vezes nas páginas 18, 30, 54 e 58.

Belhaiza, S.; M'Hallah, R.; Ben Brahim, G. A new hybrid genetic variable neighborhood search heuristic for the vehicle routing problem with multiple time windows. In: **2017 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.: s.n.], 2017. p. 1319–1326. Citado 9 vezes nas páginas 34, 54, 58, 80, 88, 89, 92, 101 e 105.

BELL, Walter J. et al. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. **Interfaces**, INFORMS, v. 13, n. 6, p. 4–23, 1983. ISSN 00922102, 1526551X. Disponível em: <http://www.jstor.org/stable/25060491>. Citado 4 vezes nas páginas 16, 17, 34 e 52.

Ben Ticha, Hamza et al. Empirical analysis for the vrptw with a multigraph representation for the road network. **Computers & Operations Research**, v. 88, p. 103 – 116, 2017. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054817301636>. Citado 2 vezes nas páginas 30 e 31.

Bitao, P.; Fei, W. Hybrid intelligent algorithm for vehicle routing problem with multiple time windows. In: **2010 International Forum on Information Technology and Applications**. [S.l.: s.n.], 2010. v. 1, p. 181–184. Citado 4 vezes nas páginas 18, 34, 45 e 48.

BRAEKERS, Kris; RAMAEKERS, Katrien; Van Nieuwenhuyse, Inneke. The vehicle routing problem: State of the art classification and review. **Computers and Industrial Engineering**, Elsevier Ltd, v. 99, p. 300–313, 2016. ISSN 03608352. Citado na página 18.

CORDEAU, Jean François et al. Chapter 6 Vehicle Routing. **Handbooks in Operations Research and Management Science**, v. 14, n. C, p. 367–428, 2007. ISSN 09270507. Citado 2 vezes nas páginas 15 e 16.

CORMEN, Thomas H. et al. **Introduction to algorithms**. [S.l.]: MIT Press, 2009. Citado na página 17.

DANILOVA, Anastasia. **Multi-period Vehicle Routing Problem with multiple Time Windows**. Dissertação (Mestrado) — Molde University College, 2018. Citado na página 42.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management Science**, Institute for Operations Research and the Management Sciences (INFORMS), v. 6, n. 1, p. 80–91, out. 1959. Disponível em: <https://doi.org/10.1287/mnsc.6.1.80>. Citado 2 vezes nas páginas 15 e 22.

DONG, Wenbo et al. A tissue p system based evolutionary algorithm for multi-objective vrptw. **Swarm and Evolutionary Computation**, v. 39, p. 310 – 322, 2018. ISSN 2210-6502. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2210650216305867>. Citado na página 41.

DORIGO, Marco; STüTZLE, Thomas. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In: _____. [S.l.: s.n.], 2006. v. 37, p. 250–285. Citado na página 48.

DUMAS, Yvan; DESROSIERS, Jacques; SOUMIS, François. The pickup and delivery problem with time windows. **European Journal of Operational**

Research, Elsevier BV, v. 54, n. 1, p. 7–22, set. 1991. Disponível em: <https://doi.org/10.1016/0377-2217(91)90319-q>. Citado na página 25.

EDEN, Amnon H. Three paradigms of computer science. **Minds and machines**, Springer, v. 17, n. 2, p. 135–167, 2007. Citado na página 19.

ELHASSANIA, Messaoud; JAOUAD, Boukachour; AHMED, Elhilali Alaoui. Solving the dynamic Vehicle Routing Problem using genetic algorithms. In: **Proceedings of 2nd IEEE International Conference on Logistics Operations Management, GOL 2014**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2014. p. 62–69. ISBN 9781479946501. Citado na página 25.

FAVARETTO, Daniela; MORETTI, Elena; PELLEGRINI, Paola. Ant colony system for a vrp with multiple time windows and multiple visits. **Journal of Interdisciplinary Mathematics**, Taylor & Francis, v. 10, n. 2, p. 263–284, 2007. Disponível em: <https://doi.org/10.1080/09720502.2007.10700491>. Citado 5 vezes nas páginas 53, 54, 55, 56 e 58.

FERDI, Imene; LAYEB, Abdesslem. A novel heuristic based simulated annealing for the capacitated location routing problem. In: **ACM International Conference Proceeding Series**. [S.l.]: Association for Computing Machinery, 2016. p. 17–24. ISBN 9781450348768. Citado 2 vezes nas páginas 24 e 50.

FISHER, Marshall L. Optimal solution of vehicle routing problems using minimum k-trees. **Operations Research**, v. 42, n. 4, p. 626–642, 1994. Disponível em: <https://doi.org/10.1287/opre.42.4.626>. Citado na página 58.

GHILAS, Veaceslav; DEMIR, Emrah; Van Woensel, Tom. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. **Computers & Operations Research**, v. 72, p. 12–30, 2016. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054816300144>. Citado 3 vezes nas páginas 44, 45 e 47.

HANSHAR, Franklin T.; OMBUKI-BERMAN, Beatrice M. Dynamic vehicle routing using genetic algorithms. **Applied Intelligence**, v. 27, n. 1, p. 89–99, aug 2007. Disponível em: <https://link.springer.com/article/10.1007/s10489-006-0033-z>. Citado na página 25.

HASLE, G; KLOSTER, O. Industrial vehicle routing problems. **Geometric Modelling, Numerical Simulation and Optimization: Applied Mathematics at SINTEF**, 2007. Citado na página 15.

HOOGEBOOM, Maaike et al. Efficient neighborhood evaluations for the vehicle routing problem with multiple time windows. **Transportation Science**, 01 2020. Citado 12 vezes nas páginas 18, 34, 56, 58, 80, 81, 88, 89, 92, 93, 95 e 101.

HURKAłA, Jarosław. Time-dependent traveling salesman problem with multiple time windows. In: **?** [S.l.: s.n.], 2015. p. 71–78. Citado 4 vezes nas páginas 25, 45, 55 e 58.

KHOUKHI, Saâdia et al. A genetic algorithm for solving a multi-trip vehicle routing problem with time windows and simultaneous pick-up and delivery in a hospital complex. In: **ACM International Conference Proceeding Series**. [S.l.]: Association for Computing Machinery, 2019. p. 76–80. ISBN 9781450366120. Citado 2 vezes nas páginas 25 e 49.

KOK, A. et al. Dynamic programming algorithm for the vehicle routing problem with time windows and ec social legislation. **Journal of Chemical Physics - J CHEM PHYS**, 01 2009. Citado na página 42.

LARSEN, Allan; MADSEN, Oli B G. The dynamic vehicle routing problem. 2000. Citado na página 15.

LARSEN, Rune; PACINO, Dario. Fast delta evaluation for the vehicle routing problem with multiple time windows. **ArXiv**, 05 2019. Citado 5 vezes nas páginas 17, 34, 44, 55 e 58.

LI, Haibing; LIM, Andrew. Local search with annealing-like restarts to solve the vehicle routing problem with time windows. In: **Proceedings of the 2002 ACM symposium on Applied computing - SAC '02**. New York, New York, USA: ACM Press, 2002. p. 560. ISBN 1581134452. Disponível em: <http://portal.acm.org/citation.cfm?doid=508791.508900>. Citado na página 31.

LI, Jiliu et al. Branch-and-price-and-cut for the synchronized vehicle routing problem with split delivery, proportional service time and multiple time windows. **Transportation Research Part E: Logistics and Transportation Review**, v. 140, p. 101955, 2020. ISSN 1366-5545. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1366554520306062>. Citado 4 vezes nas páginas 17, 18, 33 e 34.

LIANG, Chun Hua; ZHOU, Hong; ZHAO, Jian. Vehicle routing problem with time windows and simultaneous pickups and deliveries. In: **IE and EM 2009 - Proceedings 2009 IEEE 16th International Conference on Industrial Engineering and Engineering Management**. [S.l.: s.n.], 2009. p. 685–689. ISBN 9781424436705. Citado na página 24.

LIN, Shih-Wei; YU, Vincent F. A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows. **Applied Soft Computing**, v. 37, p. 632 – 642, 2015. ISSN 1568-4946. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494615005712>. Citado 2 vezes nas páginas 17 e 45.

LINTZMAYER, Carla Negri; MOTA, Guilherme Oliveira. **Análise de Algoritmos e Estruturas de Dados**. [S.l.]: UFABC - Universidade Federal do ABC, 2018. Citado na página 42.

LIU, Ran et al. Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. **European Journal of Operational Research**, v. 230, n. 3, p. 475 – 486, 2013. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221713003585>. Citado 2 vezes nas páginas 15 e 23.

LU, Chung-Cheng; YANG, Cheng-Yao. The performance of genetic algorithms on solving the pickup and delivery vehicle routing problem. **Asian Transport Studies**, v. 2, n. 1, p. 64–79, 2012. Citado na página 16.

LU, Chung-Cheng; YU, Vincent F. Data envelopment analysis for evaluating the efficiency of genetic algorithms on solving the vehicle routing problem with soft time windows. **Computers & Industrial Engineering**, v. 63, n. 2, p. 520 – 529, 2012. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835212000939>. Citado na página 49.

LUTZ, Roman. **Adaptive Large Neighborhood Search, A heuristic for the Rich Pickup and Delivery Problem with Time Windows**. Dissertação (Mestrado) — Ulm University Universitat, Germany, 2014. Citado na página 51.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de metodologia científica**. [S.l.]: 5. ed.-São Paulo: Atlas, 2003. Citado na página 19.

MIRHASSANI, S.A.; ABOLGHASEMI, N. A particle swarm optimization algorithm for open vehicle routing problem. **Expert Systems with Applications**, v. 38, n. 9, p. 11547 – 11551, 2011. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417411004404>. Citado 2 vezes nas páginas 15 e 18.

MIRHASSANI, S.A.; ABOLGHASEMI, N. A particle swarm optimization algorithm for open vehicle routing problem. **Expert Systems with Applications**, v. 38, n. 9, p. 11547 – 11551, 2011. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417411004404>. Citado na página 23.

MONTEMANNI, R.; GAMBARDELLA, L.M. An ant colony system for team orienteering problems with time windows. **Foundations of Computing and Decision Sciences**, Vol. 34, No. 4, p. 287–306, 2009. Citado na página 58.

MORIN, Edgar; MOIGNE, Jean-Louis Le. A inteligência da complexidade. Ed. Fundação Peirópolis, 2000. Citado na página 20.

NACCACHE, Salma; CôTé, Jean-François; COELHO, Leandro C. The multi-pickup and delivery problem with time windows. **European Journal of Operational Research**, v. 269, n. 1, p. 353 – 362, 2018. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221718300717>. Citado na página 16.

Nunes Bezerra, Sinaide et al. A VNS-Based Algorithm with Adaptive Local Search for Solving the Multi-Depot Vehicle Routing Problem. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. Springer Verlag, 2019. v. 11328 LNCS, p. 167–181. ISBN 9783030158422. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-030-15843-9_14>. Citado na página 25.

OKULEWICZ, Michal; MANDZIUK, Jacek. Two-phase multi-swarm PSO and the dynamic vehicle routing problem. In: **IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIHLI 2014: 2014 IEEE Symposium on Computational Intelligence for Human-Like Intelligence, Proceedings**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2014. ISBN 9781479945078. Citado na página 49.

OYOLA, Jorge; ARNTZEN, Halvard; WOODRUFF, David L. The stochastic vehicle routing problem, a literature review, part ii: solution methods. **EURO Journal on Transportation and Logistics**, v. 6, n. 4, p. 349–388, Dec 2017. ISSN 2192-4384. Disponível em: <https://doi.org/10.1007/s13676-016-0099-7>. Citado 2 vezes nas páginas 18 e 42.

OYOLA, Jorge; ARNTZEN, Halvard; WOODRUFF, David L. The stochastic vehicle routing problem, a literature review, part i: models. **EURO Journal on Transportation and Logistics**, v. 7, n. 3, p. 193–221, Sep 2018. ISSN 2192-4384. Disponível em: <https://doi.org/10.1007/s13676-016-0100-5>. Citado 2 vezes nas páginas 18 e 42.

PAN, LJ. Vehicle routing problem with time windows and its algorithm. **Central South University**, 2012. Citado na página 42.

PAULSEN, Niklas; DIEDRICH, Florian; JANSEN, Klaus. Heuristic Approaches to Minimize Tour Duration for the TSP with Multiple Time Windows. In: ITALIANO, Giuseppe F.; SCHMIDT, Marie (Ed.). **15th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2015)**. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. (OpenAccess Series in Informatics (OASIcs), v. 48), p. 42–55. ISBN 978-3-939897-99-6. ISSN 2190-6807. Disponível em: <http://drops.dagstuhl.de/opus/volltexte/2015/5460>. Citado 4 vezes nas páginas 17, 26, 55 e 58.

PESANT, Gilles et al. On the flexibility of constraint programming models: From single to multiple time windows for the traveling salesman problem. **European Journal of Operational Research**, v. 117, n. 2, p. 253 – 263, 1999. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221798002483>. Citado 3 vezes nas páginas 17, 53 e 58.

PISINGER, David; ROPKE, Stefan. A general heuristic for vehicle routing problems. **Computers & Operations Research**, Elsevier BV, v. 34, n. 8, p. 2403–2435, ago. 2007. Disponível em: <https://doi.org/10.1016/j.cor.2005.09.012>. Citado na página 51.

POP, Petrică C.; FUKSZ, Levente; MARC, Andrei Horvat. A variable neighborhood search approach for solving the generalized vehicle routing problem. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. Springer Verlag, 2014. v. 8480 LNAI, p. 13–24. ISBN 9783319076164. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-319-07617-1_2>. Citado na página 50.

PRATAMA, Rayandra; MAHMUDY, Wayan. Optimization of vehicle routing problem with time window (vrptw) for food product distribution using genetics algorithm. **Journal of Information Technology and Computer Science**, v. 2, n. 2, 2017. ISSN 2540-9824. Disponível em: <http://jitecs.ub.ac.id/index.php/jitecs/article/view/16>. Citado na página 31.

PUREZA, Vitória; MORABITO, Reinaldo; REIMANN, Marc. Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the vrptw. **European Journal of Operational Research**, v. 218, n. 3, p. 636 – 647, 2012. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221711010605>. Citado 2 vezes nas páginas 22 e 34.

QURESHI, Ali; TANIGUCHI, Eiichi; YAMADA, Tadashi. An analysis of exact vrptw solutions on its data-based logistics instances. **International Journal of Intelligent Transportation Systems Research**, v. 10, 01 2011. Citado na página 24.

RANCOURT, Marie-Eve; CORDEAU, Jean-François; LAPORTE, Gilbert. Long-haul vehicle routing and scheduling with working hour rules. **Transportation Science**, v. 47, p. 81–107, 02 2013. Citado 4 vezes nas páginas 17, 34, 53 e 58.

RIZZOLI, A. E. et al. Ant colony optimization for real-world vehicle routing problems: From theory to applications. **Swarm Intelligence**, v. 1, n. 2, p. 135–151, 2007. Disponível em: <https://link.springer.com/article/10.1007/s11721-007-0005-x>. Citado na página 49.

Sakakibara, K.; Tamaki, H.; Nishikawa, I. Autonomous distributed approaches for pickup and delivery problems with time windows. In: **SICE Annual Conference 2007**. [S.l.: s.n.], 2007. p. 2639–2642. Citado 4 vezes nas páginas 16, 18, 42 e 48.

SAVELSBERGH, M.W.P. An efficient implementation of local search algorithms for constrained routing problems. **European Journal of Operational Research**, v. 47, n. 1, p. 75 – 85, 1990. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/037722179090091O>. Citado na página 42.

SAVELSBERGH, Martin W. P. The vehicle routing problem with time windows: Minimizing route duration. **ORSA Journal on Computing**, v. 4, n. 2, p. 146–154, 1992. Disponível em: <https://doi.org/10.1287/ijoc.4.2.146>. Citado na página 26.

SCHMITT, Joao Pedro; BALDO, Fabiano; PARPINELLI, Rafael Stubs. A MAX-MIN ant system with short-term memory applied to the dynamic and asymmetric traveling salesman problem. In: **2018 7th Brazilian Conference on Intelligent Systems (BRACIS)**. IEEE, 2018. Disponível em: <https://doi.org/10.1109/bracis.2018.00009>. Citado 3 vezes nas páginas 16, 25 e 60.

SHAW, Paul. Using constraint programming and local search methods to solve vehicle routing problems. In: **Principles and Practice of Constraint Programming — CP98**. Springer Berlin Heidelberg, 1998. p. 417–431. Disponível em: <https://doi.org/10.1007/3-540-49481-2_30>. Citado 2 vezes nas páginas 48 e 51.

SOLOMON, Marius M. Algorithms for the vehicle routing and scheduling problems with time window constraints. **Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), v. 35, n. 2, p. 254–265, abr. 1987. Disponível em: <https://doi.org/10.1287/opre.35.2.254>. Citado 5 vezes nas páginas 53, 58, 83, 84 e 85.

SOUFFRIAU, Wouter; VANSTEENWEGEN, Pieter; BERGHE, Greet Vanden. The multiconstraint team orienteering problem with multiple time windows. **Transportation Science**, v. 47, p. 53–63, 02 2013. Citado 4 vezes nas páginas 17, 50, 54 e 58.

TAO, Zhang et al. A mixed PSO algorithm for the VRPSPD. In: **Chinese Control and Decision Conference, 2008, CCDC 2008**. [S.l.: s.n.], 2008. p. 4017–4021. ISBN 9781424417346. Citado na página 50.

TASAN, A. Serdar; GEN, Mitsuo. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. **Computers & Industrial Engineering**, v. 62, n. 3, p. 755 – 761, 2012. ISSN 0360-8352. Soft Computing for Management Systems. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835211003482>. Citado na página 16.

TILK, Christian; IRNICH, Stefan. Dynamic programming for the minimum tour duration problem. **Transportation Science**, v. 51, n. 2, p. 549–565, 2017. Disponível em: <https://doi.org/10.1287/trsc.2015.0626>. Citado na página 25.

TRICOIRE, Fabien et al. Heuristics for the multi-period orienteering problem with multiple time windows. **Computers & Operations Research**, v. 37, n. 2, p. 351 – 367, 2010. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S030505480900152X>. Citado 7 vezes nas páginas 17, 26, 53, 54, 55, 56 e 58.

TRICOIRE, Fabien et al. Addendum to "heuristics for the multi-period orienteering problem with multiple time windows". **Computers & Operations Research**, v. 40, n. 5, p. 1516 – 1519, 2013. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S030505481200233X>. Citado 4 vezes nas páginas 17, 53, 56 e 58.

TRIPATHI, Mukul; Glenn Kuriger; WAN, Hung-da. **AN ANT BASED SIMULATION OPTIMIZATION FOR VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS**. [S.l.]: Winter Simulation Conference, 2009. ISBN 9781424457717. Citado na página 49.

TRIPP, David. Pesquisa-ação: uma introdução metodológica. **Educação e pesquisa**, SciELO Brasil, v. 31, n. 3, 2005. Citado na página 19.

VIDAL, Thibaut et al. **A Unifying View on Timing Problems and Algorithms**. [S.l.: s.n.], 2011. Citado na página 26.

Wang, Y. et al. Multiperiod coverage path planning and scheduling for airborne surveillance. **IEEE Transactions on Aerospace and Electronic Systems**, v. 54, n. 5, p. 2257–2273, 2018. Citado na página 47.

WAZLAWICK, Raul. **Metodologia de pesquisa para ciência da computação**. [S.l.]: Elsevier Brasil, 2017. v. 2. Citado na página 20.

WEI, Lijun et al. A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. **European Journal of Operational Research**, v. 265, n. 3, p. 843 – 859, 2018. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S037722171730766X>. Citado 2 vezes nas páginas 15 e 23.

WILCOXON, Frank. Individual comparisons by ranking methods. **Biometrics Bulletin**, [International Biometric Society, Wiley], v. 1, n. 6, p. 80–83, 1945. ISSN 00994987. Disponível em: <http://www.jstor.org/stable/3001968>. Citado na página 81.

XU, Zongyan; LI, Haihua; WANG, Yilin. An improved genetic algorithm for vehicle routing problem. In: **Proceedings - 2011 International Conference on Computational and Information Sciences, ICCIS 2011**. [S.l.: s.n.], 2011. p. 1132–1135. ISBN 9780769545011. Citado na página 50.

YU, Bin; YANG, Zhong Zhen. An ant colony optimization model: The period vehicle routing problem with time windows. **Transportation Research Part E: Logistics and Transportation Review**, v. 47, n. 2, p. 166 – 181, 2011. ISSN 1366-5545. Disponível em: <http://www.sciencedirect.com/science/article/pii/S136655451000089X>. Citado 4 vezes nas páginas 15, 16, 22 e 23.