O problema roteamento de veículos, com coleta e entrega de encomendas, janela de tempo e requisições dinâmicas (DPDPTW) é geralmente encontado em empresas de *disk*-entrega como UberEats e iFood. O objetivo nesse problema é otimizar uma frota de veículos para fazer o melhor atendimento possível dos clientes, visando reduzir o custo operacional e atender todas restrições do problema. O DPDPTW é um problema dinâmico, atualmente essa classe apresenta diversas oportunidades de melhoria e existem poucos trabalhos que propõem soluções. Portanto, o trabalho em tela visa propor uma abordagem heurística de busca larga adaptativa em vizinhança (ALNS) como solução de otimização combinatorial para o DPDPTW. A solução proposta executa a otimização online e lida com eventos dinâmicos ao mesmo tempo. Dado a falta de um benchmark do DPDPTW, a avaliação da solução proposta foi feita contrastando o problema com e sem dinamicidade, para analisar o impacto na otimização. Os resultados apontam que a solução é capaz de realizar a otimização, porém a urgência dos atendimentos é o fator de maior impacto no custo das soluções encontradas.

Orientador: Fabiano Baldo

Coorientador: Rafael Stubs Parpinelli

Joinville, 2020

ANO
2020

JOÃO PEDRO SCHMITT | PROPOSTA DE ALGORITMO DE BUSCA LARGA ADAPTATIVA EM VIZINHANÇA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA, JANELA DE TEMPO E REQUISIÇÕES DINÂMICAS

UDESC

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

DISSERTAÇÃO DE MESTRADO

PROPOSTA DE ALGORITMO DE BUSCA LARGA ADAPTATIVA EM VIZINHANÇA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA, JANELA DE TEMPO E REQUISIÇÕES DINÂMICAS

JOÃO PEDRO SCHMITT

JOINVILLE, 2020

UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO EM COMPUTAÇÃO APLICADA

JOÃO PEDRO SCHMITT

PROPOSTA DE ALGORITMO DE BUSCA LARGA ADAPTATIVA EM VIZINHANÇA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA, JANELA DE TEMPO E REQUISIÇÕES DINÂMICAS

JOINVILLE

2020

JOÃO PEDRO SCHMITT

PROPOSTA DE ALGORITMO DE BUSCA LARGA ADAPTATIVA EM VIZINHANÇA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA, JANELA DE TEMPO E REQUISIÇÕES DINÂMICAS

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Dr. Fabiano Baldo
Coorientador: Dr. Rafael Stubs Parpinelli

JOINVILLE

2020

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Setorial do CCT/UDESC,

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

Schmitt, João Pedro
Proposta de Algoritmo de Busca Larga Adaptativa em Vizinhança para o Problema de Roteamento de Veículos com Coleta e Entrega, Janela de Tempo e Requisições Dinâmicas / João Pedro Schmitt. -- 2020.
158 p.

Orientador: Fabiano Baldo
Coorientador: Rafael Stubs Parpinelli
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2020.

1. problema do roteamento de veículos. 2. coleta e entrega. 3. requisições dinâmicas. 4. heurística. 5. busca larga adaptativa em vizinhança. I. Baldo, Fabiano. II. Parpinelli, Rafael Stubs. III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. IV. Titulo.

Proposta de Algoritmo de Busca Larga Adaptativa em Vizinhança para o Problema de Roteamento de Veículos com Coleta e Entrega, Janela de Tempo e Requisições Dinâmicas

por

João Pedro Schmitt

Esta dissertação foi julgada adequada para obtenção do título de

Mestre em Computação Aplicada

Área de concentração em "Ciência da Computação",
e aprovada em sua forma final pelo

CURSO DE MESTRADO ACADÊMICO EM COMPUTAÇÃO APLICADA
DO CENTRO DE CIÊNCIAS TECNOLÓGICAS DA
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Prof. Dr. Fabiano Baldo
CCT/UDESC
(Orientador/Presidente)

Prof. Dr. Rafael Stubs Parpinelli
CCT/UDESC (Coorientador)
Membro da Banca Examinadora

Profa. Dra. Aurora Trinidad Ramirez Pozo
INF/UFPR
Membro da Banca Examinadora

Prof. Dr. Omir Correia Alves Junior
CCT/UDESC
Membro da Banca Examinadora

Joinville,SC, 13 de março de 2020.

Dedico essa pesquisa para minha mãe, meu padrasto, minha namorada, meus professores e demais colegas que estiveram ao meu lado durante toda minha vida acadêmica. Também, em especial, dedico essa pesquisa para meu falecido pai.

AGRADECIMENTOS

Gostaria de deixar meus agradecimentos, primeiramente a Deus, por me dar força para empreitar essa jornada. À minha mãe, Sandra, e meu padrasto, Paulo, por todo apoio e suporte durante minha vida acadêmica. À minha namorada, Denise, pela paciência nas incansáveis horas de estudo. À meus colegas, Jacques, Manfred e Fábio Dippold pelo incentivo para iniciar no programa de Mestrado. À meu orientador, Fabiano Baldo, e meu co-orientador, Rafael Parpinelli, pelas muitas horas de discussão e orientação. E por fim a WEG, por ter proporcionado a flexibilidade necessária para realização do meu mestrado.

"*Se eu vi mais longe, foi por estar sobre ombros de gigantes*"
(Isaac Newton)

RESUMO

O VRP é uma área de estudo bastante abrangente, geralmente abordado como parte dos sistemas de gerenciamento de transporte (*Transport Management System* – TMS). Atualmente, o modelo *last-mile* vem recebendo muita atenção dentro dos TMS's por tratar das entregas de última milha direta ao consumidor. Neste modelo, a qualquer momento podem ser recebidas ligações dos clientes sem aviso prévio requisitando coletas ou entregas para o mesmo dia. Esse tipo de solicitação é conhecida como requisição dinâmica. Na literatura, o problema de coleta e entrega de encomendas com janela de tempo e requisições dinâmicas (DPDPTW) muito se assemelha ao modelo de distribuição *last-mile*. O DPDPTW é uma variação do problema PDPTW no qual é adicionada a característica de dinamicidade no recebimento das requisições. O PDPTW é um problema largamente abordado na literatura, diferentemente do DPDPTW, onde poucos trabalhos são encontrados. O desenvolvimento de soluções computacionais para o VRP não é trivial dada a explosão combinatorial necessária para calcular a solução do problema. Estudos apontam as vantagens da utilização da heurística de busca larga adaptativa em vizinhança (ALNS) em problemas de otimização combinatorial. Portanto, este trabalho tem a seguinte pergunta de pesquisa: como o ALNS pode ser usado para encontrar soluções do DPDPTW, minimizando o número de veículos e o custo total do roteamento? Com base nisso, esse trabalho propõem o desenvolvimento de um modelo de otimização baseado no ALNS capaz de resolver o DPDPTW, chamado de ALNS-DPDP. Para avaliar o modelo proposto, experimentos foram realizados com dois *benchmarks*, um aplicado ao PDPTW e outro ao DPDPTW. Na primeira análise, sob o cenário estático (PDPTW), o modelo proposto teve em média um desvio de 2.27% no número de veículos e de -2.36% no custo total. Esse resultado é considerado compatível e adequado com a literatura, pois as melhores soluções da literatura não foram encontradas por um único algoritmo. Na segunda análise, sob o cenário dinâmico (DPDPTW), pode-se perceber que as soluções encontradas pelo modelo sofrem um acréscimo de até 108% no número de veículos e de até 72% no custo total a medida que o grau de dinamicidade aumenta. Esse resultado foi impactado principalmente pelo aumento do grau de urgência das requisições. Essa análise permitiu identificar limitações na otimização a longo prazo, pois as rotas tendem a ser desviadas para atender as requisições dinâmicas. Uma última contribuição deste trabalho é a construção de um *benchmark* para o problema DPDPTW que poderá servir de base para estudos futuros.

Palavras-chaves: problema do roteamento de veículos, coleta e entrega, janelas de tempo, requisições dinâmicas, heurística, busca larga adaptativa em vizinhança, busca local.

ABSTRACT

The VRP is a broad study field, usually tackled as part of transport management systems (TMS). Nowadays, the last-mile model has been received a lot of attention in the TMS systems, mainly because of its focus on the delivery of products directly to the final client. Besides that, in this model clients can make a call at any time requesting a pickup or delivery on the same day, without previous notification. This kind of request is known as dynamic request. In the literature, the last-mile model can be framed as the pickup and delivery problem with time windows and dynamic requests (DPDPTW). The DPDPTW is an extension of the PDPTW where the dynamic requests are considered. The PDPTW problem has received considerable attention on the literature, differently from the DPDPTW that has not received much attention yet. The development of solutions to VRP problems is not a trivial task because it has to deal with a combinatorial explosion of calculations to find a suitable solution for the problem. Studies highlight the advantages of the adaptive large neighborhood search (ALNS) method to deal with such combinatorial problems. Therefore, the research question addressed in the study is: is it possible to apply the ALNS to solve the DPDPTW problem, minimizing the number of vehicles and the total cost of the solution? Based on this assumption, this study proposes the development of a model based on ALNS to solve the DPDPTW, named here as ALNS-DPDP. To validate the proposed model, two benchmarks were executed, the first one applied to the PDPTW and the second one applied to the DPDPTW. In the first analysis, related to the static scenario (PDPTW), the results found indicate that the proposed method has a deviation of 2.27% in the number of vehicles and -2.36% of the total cost related to the best-known solutions from the literature. It allows us to conclude that the ALNS-DPDP is competitive with the literature because in that there is no one algorithm that found all the best solutions. In the second analysis, related to the dynamic scenario (DPDPTW), the results indicate that the number of vehicles increases by up to 108% and the total cost up to 72% when the dynamic degree is increased. This result is mainly impacted by the increment of the urgency factor of dynamic requests. Besides that, the results allowed us to identify limitations related to the long term optimization of the route, because when dynamic requests are announced the vehicles tend to take big deviations to attend the new client. Finally, this work also contributes by constructing a benchmark that can be used to develop methods applied to the DPDPTW problem.

Key-words: vehicle routing problem, pickup and delivery, time windows, dynamic requests, heuristics, adaptive large neighborhood search, local search.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

VRP	*Vehicle Routing Problem*
TMS	*Transport Management Systems*
TaaS	*Transport-as-a-Service*
TSP	*Travelling Salesman Problem*
PDPTW	*Pickup and Delivery Problem with Time Windows*
DPDPTW	*Dynamic Pickup and Delivery Problem with Time Windows*
ALNS	*Adaptive Large Neighborhood Search*
CVRP	*Capacited Vehicle Routing Problem*
VRPTW	*Vehicle Routing Problem with Time Windows*
PVRP	*Periodic Vehicle Routing Problem*
DARP	*Dial-a-Ride Problem*
VRPPDTW	*Vehicle Routing Problem with Pickup and Delivery and Time Windows*
DVRP	*Dynamic Vehicle Routing Problem*
GPS	*Global Positioning System*
LNS	*Large Neighborhood Search*
SA	*Simulated Annealing*
MPDPTW	*Multi-Pickup and Delivery Problem with Time Windows*
LS	*Local-Search*
RPPDPTW	*Rich Pickup and Delivery Problem with Time Windows*
ACO	*Ant Colony Optimization*
GGA	*Grouping Genetic Algorithm*
ALNS-DPDP	*Adaptive Large Neighborhood Search for Dynamic Pickup Delivery Problem*
ALNS-NV	*Adaptive Large Neighborhood Search for Number of Vehicles*
ALNS-TC	*Adaptive Large Neighborhood Search for Total Cost*
TDD	*Test-driven development*

PC	*Personal Computer*
LC	Instância com distribuição dos nós clusterizada
LRC	Instância com distribuição dos nós semi-clusterizada
LR	Instância com distribuição dos nós aleatória
MV	*Moving vehicle*
CPU	*Central Processing Unit*
GAP	Diferença relativa entre dois valores
BSF	*Best-so-far*
SD	*Standard Deviation*

LISTA DE SÍMBOLOS

R	Requisição do cliente
M	Número de requisições
P^+	Nós de coletas das requisições dos clientes, dado que $P^+ = \{1, 2, ..., M\}$
P^-	Nós de entrega das requisições dos clientes, dado que $P^- = \{M + 1, M + 2, ..., 2M\}$
P_i^+	Nó de coleta de uma dada requisição, dado que $1 \leq i \leq M$
P_{M+i}^-	Nó de entrega de uma dada requisição, dado que $1 \leq i \leq M$
P	Todos os nós das requisições, dado que $P = P^+ \cup P^-$
D	Nó do depósito, dado que $D = \{0, 2M + 1\}$
N	Conjunto total de nós (pontos de coleta, entrega e depósito), dado que $N = \{0, 1, 2, ..., M, M + 1, M + 2, ..., 2M, 2M + 1\}$
G_i^x	Latitude do nó, dado que $i \in N$
G_i^y	Longitude do nó, dado que $i \in N$
q_i	Demanda da requisição, dado que $q_i = -q_{i+M}$ para $i \in P^+$
e_i	Janela de início de um nó, dado que $i \in N$
l_i	Janela de término de um nó, dado que $i \in N$
V	Frota de veículos, dado que $V = \{1, 2, ..., \|V\|\}$
v	Veículo, $v \in V$
t_{ij}	Tempo de viajem entre nós, dado que $i, j \in N$
c_{ij}	Custo de viajem entre nós, dado que $i, j \in N$
s_i	Tempo de serviço, dado que $i \in N$
X_{ij}^v	Variável binária dado que $v \in V, i, j \in N$. O valor é 1 se o arco (i, j) está sendo usado pelo veículo v, caso contrário é 0
T_i	Tempo de início do atendimento de um nó, dado que $i \in N$.
T_0^v	Tempo de partida de um veículo do depósito, dado que $v \in V$
T_{2M+1}^v	Tempo de retorno de um veículo para o depósito, dado que $v \in V$
C	Capacidade de carga dos veículos
Y_i	Variável de carga ao término de um atendimento, dado que $i \in P$

Γ	Custo operacional aplicado para cada veículo em uso.
$T^{arrival}$	Tempo de chegada em um nó
T^{depart}	Tempo de partida de um nó
T^{wait}	Tempo de espera em um nó
T^{slack}	Tempo de folga de um nó
T_{start}	Início do horizonte de tempo do dia de trabalho
T_{end}	Término do horizonte de tempo do dia de trabalho
T_{wd}	Tempo do horizonte do dia de trabalho, $T_{wd} = T_{end} - T_{start}$
T_{setup}	Tempo de setup de um novo veículo
NV	Número de veículo
TC	Custo total
FC	Factibilidade
H_v	Rota do veículo v
$T_{current}$	Tempo corrente
T^{latest}	Último momento válido para anunciar uma requisição
A_i	Horário de anunciação de uma requisição, dado que $i < M$
Φ_{dod}	Medição geral do grau de dinamicidade
Φ_{edod}	Medição do grau de dinamicidade efetiva
Φ_{edodtw}	Medição do grau de dinamicidade efetiva com janela de tempo
ϱ	Fator de requisições dinâmicas
r_i	Tempo de reação do nó i
a	Fator de urgência
φ	Fator de requisições a-priori
ϱ	Fator de requisições dinâmicas, $\varrho = 1 - \varphi$
F_{ts}	Número de fatias de tempo do dia de trabalho
T_{ts}	Tempo das fatias de tempo
S	Solução do problema
S'	Solução do problema nova
S_{best}	Melhor solução global

θ_1	ALNS peso 1
θ_2	ALNS peso 2
θ_3	ALNS peso 3
τ	Temperatura do ALNS
γ	Taxa de resfriamento do ALNS
I	Instância do problema
O^{rem}	Operadores de remoção
O^{ins}	Operadores de inserção
O^{noise}	Operadores de ruído
$\omega_{O^{rem}}$	Peso dos operadores de remoção
$\omega_{O^{ins}}$	Peso dos operadores de inserção
$\omega_{O^{noise}}$	Peso do operador de ruído
$\omega_{O_i^{rem}}$	Peso dos operadores de remoção i
$\omega_{O_i^{ins}}$	Peso dos operadores de inserção i
$\omega_{O_i^{noise}}$	Peso do operador de ruído i
k	Segmento de iterações para atualizar pesos ALNS
$u_{O^{rem}}$	Contagem de uso dos operadores de remoção
$u_{O^{ins}}$	Contagem de uso dos operadores de inserção
$u_{O^{noise}}$	Contagem de uso dos operadores de ruído
$\phi_{O^{rem}}$	Pontuação dos operadores de remoção
$\phi_{O^{ins}}$	Pontuação dos operadores de inserção
$\phi_{O^{noise}}$	Pontuação dos operadores de ruído
ρ	Fator de reação
y	Número de requisições para remoção
χ	Operador Shaw, peso do fator distância
χ'	Operador Shaw, peso do fator tempo de chegada
χ''	Operador Shaw, peso do fator carga
χ'''	Operador Shaw, peso do fator veículos
ψ_i	Quantidade de veículos que atendem uma determinada carga

ε	Tolerância do custo da solução inicial. Usado pelo cálculo da temperatura inicial
ζ	Nível de ruído, determina o fator de variação usado pelos operadores de inserção.
ξ	Controle de remoção, define um número teto de requisições que podem ser removidas.
i_{max}	Número total de iterações
ϑ	Parâmetro de penalização por requisição não atendida.
μ	Média
σ	Desvio padrão

SUMÁRIO

1 INTRODUÇÃO

O problema de roteamento de veículos (*Vehicle Routing Problem* - VRP) refere-se a um problema de otimização combinatorial, onde para um dado conjunto de requisições de serviços dos clientes, busca-se alocar uma frota de veículos que irá fazer os atendimentos da forma mais otimizada possível (HASLE; KLOSTER, 2007). No VRP, a otimização do roteamento deve ser feita atendendo às restrições do problema. Restrições são imposições que podem variar dependendo dos meios físicos, das regulamentações do governo e das limitações das empresas (CORDEAU et al., 2007). Um exemplo de restrição é a capacidade máxima de carga que um veículo pode transportar.

O VRP é uma área de estudo bastante abrangente devido aos diferentes objetivos de otimização e ao número de restrições que podem ser consideradas no problema. Alguns exemplos de objetivos comumente encontrados são: redução da distância percorrida, redução do número de veículos utilizados, redução do tempo de viagem, redução dos tempos de espera, redução dos atrasos, redução da emissão de gás carbônico e aumento do grau de satisfação dos clientes. Em relação às restrições, alguns exemplos comumente encontrados são: capacidade de carga máxima dos veículos, janela de tempo para atendimento dos clientes, tempo de serviço gasto no cliente e jornada de trabalho máxima dos motoristas (CORDEAU et al., 2007).

Os VRPs são geralmente abordados como parte dos sistemas de gerenciamento de transporte (*Transport Management System* - TMS). Esses sistemas vêm recebendo muita atenção de empresas como Uber, Convoy, Transfix e Loadsmart. O aprofundamento dos avanços no desenvolvimento dos TMS's deu início a um novo ramo de serviços, os chamados TaaS (*Transport-as-a-Service*). De acordo com Frost & Sullivan, é esperado que os TaaS ultrapassem a receita de US$52 bilhões em 2025 – muito acima dos US$18 bilhões em 2018 (MCCREA, 2019b).

Alguns estudos mostram os ganhos reais advindos da adoção dos TMS's. Segundo Ballerstedt, líder de logística da empresa Sensata Technology, a adoção de um TMS permitiu a empresa minimizar custos com transporte em 60% à 70%, mantendo a satisfação dos clientes (MCCREA, 2019a). Esses ganhos são resultados da capacidade do TMS em calcular rotas e agendamentos agrupando entregas para minimizar os custos operacionais. Outros ganhos percebidos estavam relacionados ao gerenciamento da logística, ao melhor uso das filiais e a redução do trabalho de planejamento desenvolvido pelos funcionários. Todos esses ganhos, quando somados, reduziram o custo operacional da cadeia de suprimentos da Sensata em 2.2 milhões de dólares

(MCCREA, 2019a).

Com a evolução constante das cidades e dos sistemas de comércio eletrônico, o número de encomendas transportadas vem crescendo consideravelmente, exigindo melhores sistemas de roteamento. Banker (2018) comenta sobre o modelo de distribuição *last-mile*, onde as coletas e entregas são realizadas diretamente aos clientes finais e geralmente no mesmo dia. O modelo *last-mile* vem recebendo muita atenção dentro dos TMS's por tratar da entrega de última milha direta ao consumidor, ele é comumente encontrado em sistemas de *e-commerce* onde os produtos, após comprados, são coletados em um centro de distribuição local da cidade e entregues na casa do cliente ou em facilitadores (como o Amazon Locker) o mais rápido possível. Outro exemplo são os sistemas de disque-entrega, onde o entregador deve coletar o produto em um determinado estabelecimento e entregar no local do cliente (ex: Uber Eats e IFood).

As dificuldades do comércio baseado em *last-mile* estão predominantemente relacionadas à dinamicidade do dia-à-dia da operação, (ex: aparecimento de última hora de requisições dos clientes). Diferente do modelo de processamento estático (em lote – *batch*), onde as requisições do dia são planejadas para serem entregues no dia seguinte, no modelo de roteamento dinâmico do *last-mile* é exigido um processamento contínuo 24/7 (PSARAFTIS, 1995; MCCREA, 2019a). Nesse contexto, os sistemas buscam mecanismos de comunicação com o ambiente real para manterem-se atualizados acerca da operação durante todo o dia de trabalho. Com base nessas informações, os sistemas podem ser capazes de atualizar as rotas quando eventos inesperados acontecerem. Esses eventos contemplam, mas não estão limitados, a situações tais como: acidentes de trânsito, engarrafamentos, novas solicitações de coleta ou cancelamento de entregas (BANKER, 2018).

Os problemas dinâmicos vêm crescendo devido aos avanços tecnológicos da internet móvel e dos sistemas de geolocalização, que permitem que os sistemas de VRP sejam atualizados sobre o estado dos veículos, do ambiente e das requisições. Nesse ambiente, conforme os dados vão sendo recebidos, os sistemas buscam se atualizar para procurar por melhores soluções de roteamento durante o processo logístico. Dessa forma, quando novos roteamentos são encontrados, os sistemas notificam os motoristas em tempo real a cerca das novas rotas que devem ser executadas. Um exemplo de dinamicidade são as requisições dinâmicas, onde os clientes solicitam atendimento sem aviso prévio e o sistema deve ser capaz de atualizar a rota para atender as novas demandas (LARSEN, 2000).

Em relação aos estudos em VRP, Braekers, Ramaekers e Van Nieuwenhuyse (2016) apresentam uma análise da literatura em relação a quantidade de trabalhos e das características abordadas, onde foram considerados mais de 300 artigos. Os

resultados apontam que: 90% dos estudos consideraram a capacidade de carga dos veículos, 37% as janelas de tempo dos clientes, 16% o uso de veículos heterogêneos e somente 2.45% a entrada dinâmica de requisições (BRAEKERS; RAMAEKERS; Van Nieuwenhuyse, 2016). Com base nesses resultados, e na busca por sistemas mais eficientes em problemas de *last-mile*, pode-se perceber diversas oportunidades de melhoria no contexto do roteamento dinâmico.

Uma derivação do VRP que contém diversas características do modelo de distribuição *last-mile*, e também é foco desse estudo, é o problema de coleta e entrega de encomendas com janela de tempo e requisições dinâmicas, também conhecido na literatura pelo acrônimo DPDPTW (*Dynamic Pickup and Delivery Problem with Time Windows*) (PANKRATZ, 2005; PUREZA; LAPORTE, 2008). Esse problema é a versão dinâmica do PDPTW (*Pickup and Delivery Problem with Time Windows*). Por se tratar de um caso mais complexo de solução, o DPDPTW apresenta poucos trabalhos relacionados encontrados na literatura. Nesse tipo de problema as soluções comumente propostas buscam minimizar dois objetivos: i) o número de veículos e ii) o custo total do roteamento. Ambos objetivos seguem uma ordem de prioridade, onde minimizar o número de veículos tem prioridade sobre o custo total das rotas, ou seja, é aceitável aumentar o custo total para reduzir o número de veículos (Li; Lim, 2001).

Soluções para o problema de VRP não são triviais devido à complexidade computacional advinda da explosão combinatorial necessária para calcular a solução do problema. Para caracterizar melhor sua complexidade, o VRP enquadra-se como uma generalização do problema do caixeiro viajante (*Travelling Salesman Problem* - TSP), onde a principal diferença está no fato do TSP fazer a otimização de uma única rota, enquanto o VRP permite a otimização de *n* rotas (LARSEN, 2000). A complexidade do TSP e do VRP, quando tratados como problemas de otimização combinatorial no campo da ciência da computação, é classificada como NP-Difícil (CORMEN, 2009).

Atualmente, métodos exatos não são aplicáveis a grandes instâncias de problemas de TSP e VRP por causa do elevado tempo computacional para encontrar uma solução. A literatura apresenta métodos exatos que são válidos apenas para problemas de VRP com número reduzido de instâncias. Por esse motivo, diversos estudos fazem a aplicação de meta-heurísticas na busca de uma solução de qualidade para o problema. As meta-heurísticas são uma classe de métodos desenvolvidos de forma agnóstica ao problema, elas servem de *template* para guiar o desenvolvimento de heurísticas específicas (TALBI, 2009). Segundo Braekers, Ramaekers e Van Nieuwenhuyse (2016), aproximadamente 70% dos estudos sobre VRP consideram a aplicação de meta-heurísticas, 17% dos estudos consideram aplicação de métodos exatos, 9% heurísticas clássicas e o restante métodos com soluções de tempo real e simulações.

No contexto de *last-mile*, diversos estudos já abordaram a aplicação de heurísticas e meta-heurísticas para a otimização do problema do PDPTW (Li; Lim, 2001; CARABETTI et al., 2010; TCHOUPO et al., 2017; PISINGER; ROPKE, 2007; LUTZ, 2014), enquanto poucos abordaram o problema do DPDPTW (PANKRATZ, 2005; PUREZA; LAPORTE, 2008; DRIDI; ALAIA; BORNE, 2015). Em relação as abordagens, encontra-se a aplicação de métodos construtivos e de melhoramento. Os métodos construtivos tem um pior desempenho por causa da forma que a solução é otimizada, por meio da inserção de nós sequencialmente durante a construção. No PDPTW, a inserção de uma requisição em uma rota só pode ser feita se ambos nós de coleta e entrega forem inseridos. Em métodos construtivos, avaliar a inserção de um novo nó requer computar a factibilidade de que a rota em construção poderá ser completada com sucesso, isso requer um elevado tempo computacional. Diferente dos métodos de melhoramento, que a cada inserção avaliam a factibilidade da rota já completa ser modificada de forma factível. Ainda em relação aos métodos de melhoramento, existem estratégias populacionais, que requerem a avaliação de N possibilidades por iteração, e não populacionais, que avaliam uma solução por vez.

Dentre os estudos mais relevantes, encontram-se a aplicação algoritmos de melhoramento não populacional como recozimento simulado (*simulated annealing*) (Li; Lim, 2001) e busca larga adaptativa (PISINGER; ROPKE, 2007; LUTZ, 2014), algoritmos construtivos como colônia de formiga (CARABETTI et al., 2010; TCHOUPO et al., 2017), e algoritmos de melhoramento populacional como algoritmos genéticos (PANKRATZ, 2005; DRIDI; ALAIA; BORNE, 2015). Para o problema PDPTW, vários estudos apontam que algoritmos de melhoramento não populacionais como o algoritmo de busca larga adaptativa (ALNS) são a escolha indicada pelos resultados apresentados e o bom desempenho computacional. Entretanto, poucos estudos abordam o problema do DPDPTW. Portanto, dado ao desempenho dos algoritmos de melhoramento não populacionais no PDPTW, este trabalho assume como hipótese que a heurística ALNS pode ser mais adequada para encontrar soluções otimizadas no DPDPTW.

Com base no exposto acima, e sabendo que os problemas de VRP não apresentam soluções triviais, a pergunta de pesquisa que se pretende responder é: como a heurística de busca larga adaptativa (ALNS) pode resolver o problema DPDPTW, minimizando o número de veículos e o custo total do roteamento?

1.1 OBJETIVO

Esse trabalho tem como objetivo desenvolver um modelo baseado na heurística de otimização combinatorial de busca larga adaptativa (ALNS) para resolver o problema de roteamento de veículos com coleta e entrega, janela de tempo e re-

quisições dinâmicas (DPDPTW). Esse modelo tem como função objetivo minimizar o número de veículos e o custo total da solução, de forma hierárquica, priorizando o primeiro objetivo em relação ao segundo.

1.1.1 Objetivos Específicos

Baseado no objetivo geral, este trabalho apresenta os seguintes objetivos específicos:

- Revisão bibliográfica;
- Definir funções objetivo, para minimização do número de veículos e do custo total do problema DPDPTW;
- Modelar restrições do problema DPDPTW considerando propriedades inerentes as requisições dinâmicas;
- Definir um método de anunciação das requisições dinâmicas para geração dos cenários de teste;
- Definir um método de roteamento dinâmico dos veículos para realizar o atendimento dos clientes durante o dia de trabalho;
- Definir restrições que gerenciem as otimizações que podem ser feitas em função dos clientes visitados ao longo do tempo;
- Definir uma modelagem utilizando o algoritmo ALNS que seja aplicável ao problema em questão;
- Realizar testes com um *benchmark* estático para avaliar o desempenho do algoritmo em problemas estáticos;
- Realizar testes com um *benchmark* dinâmico para avaliar o desempenho do algoritmo em problemas dinâmicos;

1.2 METODOLOGIA DE PESQUISA

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

1.2.1 Caracterização Metodológica

De acordo com a classificação metodológica encontrada na literatura, o tipo de pesquisa relacionado a este trabalho é de cunho exploratório onde busca-se desenvolver algo presumivelmente melhor, pois será proposto um método computacional diferente para a resolução de um problema da literatura (MARCONI; LAKATOS, 2003). Do ponto de vista do paradigma da pesquisa científica, esse trabalho é classificado como tecnocatra, pois utiliza a avaliação de experimentos empíricos para compreender as contribuições da solução, ou seja, as informações são analisadas *a posteriori* (EDEN, 2007). Enquanto que as abordagens racionalistas e naturalistas buscam conhecimento *a priori* por meio de deduções de raciocínio ou por dedução formal e experimentação científica rigorosa.

Quanto ao procedimento técnico adotado, este trabalho se enquadra como uma Pesquisa-Ação, ou seja, a investigação e a experimentação são aplicadas de forma cíclica onde cada iteração gera informações aplicadas a evolução da próxima iteração (TRIPP, 2005). Em relação a forma de avaliação da pesquisa, utiliza-se o método indutivo, onde o pesquisador busca por padrões em casos particulares que explicam ou se aplicam a todos os casos isolados análogos ao analisado (MORIN; MOIGNE, 2000). Por fim, a pesquisa pode se enquadrar no nível de maturidade 2 (WAZLAWICK, 2017), pois propõe um método diferente e busca conhecimento empírico *a posteriori* para validação da proposta.

1.2.2 Procedimento Metodológico

O procedimento metodológico adotado inicia com uma revisão bibliográfica dos conceitos básicos do problema de roteamento de veículo e, em seguida, se especializa para o problema de coleta e entrega, com janela de tempo e requisições dinâmicas. Na sequência, inicia-se a revisão referente ao método de solução, nesse caso, é feita uma pesquisa sob heurísticas de otimização baseadas no conceito de busca larga em vizinhança. Nessa etapa, busca-se detalhar as principais características e como tais heurísticas são aplicadas ao problema de roteamento de veículos. Ao passo que a revisão bibliográfica é conduzida, experimentações com os algoritmos das heurísticas encontradas são realizadas sob instâncias de *benchmark* para o PDPTW propostas por Li e Lim (2001), disponível em Sintef (2019).

Após a revisão da literatura, o modelo proposto é concebido por meio da técnica evolucionária de Pesquisa-Ação. Dessa forma, a cada iteração o modelo é refinado e avaliado. Os resultados dos experimentos de cada iteração servem de insumo para a próxima iteração. De forma geral, o desenvolvimento da solução inicia com a definição da função objetivo e as restrições de dinamicidade aplicadas ao problema. Na sequência, é criado um mecanismo para simular a anunciação das requisições di-

nâmicas. Ainda, é feito a aplicação de restrições temporais para suportar o roteamento online dos veículos. Posteriormente, é modelada a proposta de solução do DPDPTW usando como base a heurística ALNS. Para melhorar o desempenho da otimização dos dois objetivos do DPDPTW (minimização de veículos e minimização de custo) o modelo é composto por dois ALNS's, um para a otimização de cada objetivo do problema. Por fim, é concebido o algoritmo principal, nomeado de *solver*, responsável por gerenciar a operação do dia, que inclui: a aplicação das restrições temporais, o controle da anunciação das requisições dinâmicas, a otimização do número de veículos e a otimização do custo total, os dois últimos realizados pelos ALNS's.

Para a avaliação dos resultados, é realizada uma comparação entre os resultados da literatura (SINTEF, 2019) com os resultados dos *benchmarks* PDPTW. As comparações entre a literatura e os resultados do PDPTW são usadas para avaliar se o método proposto apresenta resultados compatíveis com os apresentados na literatura. Ainda, é realizada a avaliação do método proposto para o cenário DPDPTW. Essa segunda comparação avalia se o método suporta adequadamente o roteamento dinâmico. Essas comparações entre a literatura e os resultados do DPDPTW servem para analisar as propriedades de dinamicidade e as diferenças entre o cenário estático e dinâmico.

As métricas utilizadas para avaliar os resultados são a média e o desvio padrão das variáveis de otimização: número de veículos, factibilidade e custo total. As métricas de dinamicidade usadas para avaliar as diferenças entre o PDPTW e o DPDPTW são a urgência de atendimento das requisições e a porcentagem de requisições dinâmicas. Por fim, as métricas para avaliar o desempenho do algoritmo são o custo da melhor solução encontrada até o momento, da solução local e da temperatura do algoritmo de *simulated annealing*.

Dois *benchmarks* são utilizados para realizar os experimentos, um para o PDPTW e outro para o DPDPTW. O *benchmark* utilizado para a comparação do PDPTW é o proposto por Li e Lim (2001), disponível em Sintef (2019). Já o *benchmark* utilizado para avaliar o DPDPTW foi construído por meio da adaptação do *benchmark* de Li e Lim (2001), adicionando as características temporais propostas pelo método de (PANKRATZ, 2005). Com as adaptações incluídas no *benchmark* de Li e Lim (2001) é possível simular as requisições dinâmicas necessárias para avaliar o DPDPTW.

1.3 ESTRUTURA DO TRABALHO

Esse trabalho está organizado da seguinte forma. O capítulo 2 apresenta uma revisão dos conceitos necessários para o desenvolvimento dessa pesquisa. Nele é revisada a bibliografia sobre o problema de coleta e entrega com janela de tempo, sobre

a dinamicidade no roteamento de veículos e sobre o algoritmo de busca larga adaptativa. O capítulo 3 apresenta o método proposto utilizando o algoritmo ALNS aplicado ao problema de coleta e entrega, com janela de tempo e requisições dinâmicas. O capítulo 4 apresenta os resultados dos experimentos realizados, incluindo a adaptação de uma base de *benchmark* da literatura para simular as requisições dinâmicas. Por fim, o capítulo 5 apresenta as conclusões e propostas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo visa apresentar o problema de roteamento de veículos (VRP), suas aplicações práticas no mundo real e os métodos mais usados em sua resolução. Mais especificamente, a seção 2.1 apresenta as definições gerais relacionadas ao VRP, suas aplicações e como essa classe de problemas está organizada. A seção 2.2 detalha a versão do VRP selecionada para esse trabalho, no caso o PDPTW. A seção 2.3 define a dinamicidade no PDPTW e sua forma de aplicação. A seção 2.4 apresenta o método *Adaptive Large Neighborhood Search* (ALNS). A seção 2.5 revisa os operadores de busca local utilizados para melhorar a solução por meio da intensificação da busca. Por fim, a seção 2.7 faz uma revisão dos trabalhos relacionados encontrados na literatura à cerca do problema em questão.

2.1 O PROBLEMA DO ROTEAMENTO DE VEÍCULOS (VRP)

A importância do VRP nasce no contexto da logística, onde as empresas de distribuição têm como fator chave de sucesso o bom planejamento das rotas de seus veículos de entrega. De forma geral, o VRP considera que os veículos partem de um ponto comum, denominado depósito, e ao término do expediente de trabalho voltam para ele. O roteamento foca em atender um conjunto de entregas que são requisitadas pelos clientes. Geralmente, as requisições de entregas possuem as seguintes informações: localização geográfica, demanda, período da janela de tempo de atendimento, entre outras. Dessa forma, métodos para resolução do VRP buscam encontrar as melhores rotas possíveis para os veículos a partir das requisições dos clientes.

O VRP é um problema estudado nos campos da pesquisa operacional e otimização combinatorial. Dantzig e Ramser (1959) apresentaram a primeira versão do VRP aplicada ao contexto de despacho de caminhões. O objetivo do problema era encontrar uma solução que, dado um conjunto de clientes a serem visitados, pudesse atender a todos os clientes no menor custo possível. Nessa formulação, uma solução composta de vários veículos deveria atender as demandas dos clientes respeitando a capacidade máxima de carga dos veículos.

O problema do VRP tem diversas aplicações práticas, com por exemplo: entrega de correspondências, entrega de gasolina a postos de combustíveis, serviço de coleta de lixo, serviços de transporte de passageiros, serviços de disque-entregas, prestação de serviços, roteamento de robôs, entre outros. Dado a ampla diversidade de aplicações do VRP, diversas variações já foram propostas e estão presentes na literatura (ADEWUMI; ADELEKE, 2016). A seguir são apresentadas as principais:

- CVRP (*Capacited Vehicle Routing Problem*), VRP com restrição de capacidade de carga dos veículos. Nesse modelo todos os veículos possuem uma mesma capacidade de carga. Em alguns casos os veículos podem possuir capacidades diferentes, quando isso acontece o problema é definido como CVRP com frota heterogênea (ADEWUMI; ADELEKE, 2016);

- VRPTW (*Vehicle Routing Problem with Time Windows*), uma extensão do CVRP com restrição de janela de tempo para atendimento dos clientes. Nessa versão existem duas variações, a primeira define que todos os clientes devem ser atendidos na janela definida enquanto na segunda atrasos nos atendimentos são permitidos. Nesse último modelo, os atrasos geralmente são convertido em penalizações no custo operacional das soluções, por diminuir o índice de satisfação dos clientes (ADEWUMI; ADELEKE, 2016);

- PVRP (*Periodic Vehicle Routing Problem*), caracteriza o problema de roteamento periódico de veículos. Esse é um modelo em que uma agenda periódica é considerada e agendamentos são feitos considerando mais de um dia de roteamento. Geralmente, busca-se definir uma agenda para a semana de entregas de um veículo. Um exemplo prático é o sistema de entrega de gás para aquecimento residencial, em que periodicamente os entregadores precisam entregar gás aos mesmos clientes (ADEWUMI; ADELEKE, 2016);

- PDPTW ou VRPPDTW (*Vehicle Routing Problem with Pickup and Delivery and Time Windows*), define o problema de coleta e entrega de produtos com janela de tempo. Nessa versão, cada requisição de cliente possui um local de coleta e um local de entrega do produto. Esse modelo é diferente por permitir que os veículos façam a coleta da carga em locais pré-definidos pelos clientes. Entretanto, assim como nos VRPTW, toda coleta/entrega deve ser atendida respeitando capacidade de carga e janela de tempo do local de atendimento. Um exemplo prático desse tipo de problema são os serviços de disque-entrega como Uber Eats e iFood (DUMAS; DESROSIERS; SOUMIS, 1991; Li; Lim, 2001; BERBEGLIA et al., 2007);

- DVRP (*Dynamic Vehicle Routing Problem*), contextualiza o problema de VRP dinâmico. Nessa abordagem nem todos os dados a respeito do problema são conhecidos *a priori*, assim, ao longo do processo de otimização, novas informações são inseridas no problema e o modelo de solução precisa se adaptar as novas condições. Diversos modelos de dinamicidade já foram propostos na literatura, alguns exemplos são: variações nas condições de tráfego, entrada de requisições dinâmicas e demandas estocásticas (ADEWUMI; ADELEKE, 2016; LARSEN, 2000).

O trabalho em tela foca em atacar as dificuldades da indústria no que diz respeito ao modelo de *last-mile*, que na literatura pode ser enquadrado com as restrições de coleta e entrega de produtos, janela de tempo, capacidade de carga e entrada de requisições dinâmicas. Por meio da revisão da literatura, o modelo de VRP que mais se assemelha ao considerado neste trabalho é o problema de coleta e entrega com janela de tempo e requisições dinâmicas, ou DPDPTW (PANKRATZ, 2005). Esse modelo é uma variação do problema de coleta e entrega com janela de tempo estático, ou PDPTW (DUMAS; DESROSIERS; SOUMIS, 1991; Li; Lim, 2001; PANKRATZ, 2005; PARRAGH; DOERNER; HARTL, 2008; LUTZ, 2014; BERBEGLIA et al., 2007). O acrônimo DPDPTW, do inglês *Dynamic Pickup and Delivery Problem with Time Windows*, será usado durante todo o trabalho. As seções seguintes irão detalhar em profundidade a formulação do problema, primeiramente falando sobre a versão estática e em seguida introduzindo as requisições dinâmicas.

2.2 O PROBLEMA DE COLETA E ENTREGA COM JANELA DE TEMPO (PDPTW)

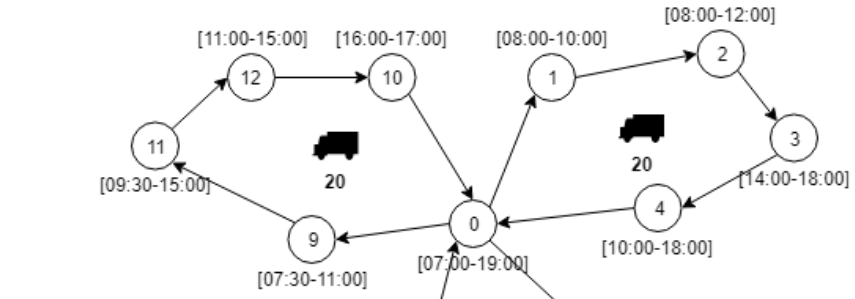
A versão do PDPTW proposta por Dumas, Desrosiers e Soumis (1991) apresenta o problema como uma generalização do problema do VRPTW. No VRPTW as requisições são coletadas no depósito (origens) e os pontos de entregas são as localidades dos clientes (destinos). Além disso, todas as requisições devem ser atendidas dentro dos limites das janelas de tempo e das capacidades de carga dos veículos. Na proposta do PDPTW, modifica-se o VRPTW para que os locais de coleta sejam definidos pelos clientes, ou seja, o veículo não é carregado no depósito com a carga do cliente mas sim no local de coleta. Para suportar essa nova condição são adicionadas duas novas restrições: 1) a restrição de paridade, em que para cada ponto de entrega deve existir um ponto de coleta capaz de suprir a demanda especificada e 2) a restrição de precedência, em que cada coleta deve preceder a entrega nas rotas dos veículos (DUMAS; DESROSIERS; SOUMIS, 1991).

Outro fator importante do VRPTW também presente no PDPTW é o tempo de serviço. Essa informação representa o tempo que o veículo leva para atender o cliente depois que chegar no ponto de entrega ou coleta. O tempo de serviço geralmente representa o tempo necessário para estacionar o veículo, descarregar ou carregar os produtos e realizar os trâmites burocráticos do atendimento (LARSEN, 2000). O tempo de serviço não é considerado dentro dos limites de atendimento da janela de tempo, ou seja, se a janela de tempo de um cliente termina as 10:00 AM e o veículo chega no cliente as 10:00 AM e precisa de 10 minutos para atender o cliente, esse será um atendimento válido (mesmo terminando as 10:10 AM).

A Figura 1 apresenta um exemplo do PDPTW para um cenário fictício. Nesse exemplo, a solução é composta por três veículos, todos com capacidade de carga má-

xima igual a 20 unidades. A tabela “Requisições” apresenta os IDs das requisições dos clientes com os respectivos pontos de coleta e entrega. Cada requisição possui uma demanda associada. Essa demanda é expressa por um valor positivo na coleta, para indicar a carga adicionada ao veículo, e por um valor negativo na entrega, para indicar a carga removida do veículo. Na Figura 1 também são apresentadas as janelas de tempo nos nós de atendimento. A solução apresentada considera que todos atendimentos respeitam as janelas especificadas. É possível notar que as requisições podem ser arranjadas de várias formas. Por exemplo, a requisição de ID 1 é coletada e entregue antes de coletar e entregar a requisição de ID 2, já as requisições de IDs 3 e 4 são coletadas e mantidas no veículos para então serem entregues na sequência. Essas mesclas de requisições são permitidas contanto que as restrições de paridade, precedência, capacidade e janela de tempo sejam atendidas.

Figura 1 – Exemplo de solução de roteamento para o problema PDPTW. A tabela e os vértices contém informações das requisições dos clientes. A solução é composta por três veículos homogêneos com capacidade de carga de 20.



Requisições		
ID Req	Coleta Nó/Carga	Entrega Nó/Carga
1	1/+20	2/-20
2	3/+15	4/-15
3	5/+5	6/-5
4	7/+10	8/-10
5	9/+10	10/-10
6	11/+10	12/-10

Fonte: Imagem gerada pelo autor.

2.2.1 Formulação matemática

A formulação matemática do PDPTW, apresentada a seguir, segue a mesma notação e terminologia usada por Dumas, Desrosiers e Soumis (1991). Essa formulação, assume que o problema é composto de um único depósito e uma frota de veículos com capacidade homogênea. Para facilitar a visualização, a Tabela 1 apresenta todas as variáveis do problema.

O PDPTW é formulado por meio da notação aplicada à problemas de otimização. A Equação 2.1a representa a função objetivo (*fitness*) sujeita às restrições do

Tabela 1 – Variáveis da formulação do PDPTW

Variáveis	**Descrição**
R	Requisições dos clientes.
M	número de requisições
P^+	Nós de coletas das requisições dos clientes, dado que $P^+ = \{1, 2, ..., M\}$.
P^-	Nós de entrega das requisições dos clientes, dado que $P^- = \{M+1, M+2, ..., 2M\}$.
P_i^+	Nó de coleta de uma dada requisição, dado que $1 \leq i \leq M$.
P_{M+i}^-	Nó de entrega de uma dada requisição, dado que $1 \leq i \leq M$.
P	Todos os nós das requisições, dado que $P = P^+ \cup P^-$.
D	Nó do depósito, dado que $D = \{0, 2M+1\}$.
N	Conjunto total de nós (pontos de coleta, entrega e depósito), dado que $N = \{0, 1, 2, ..., M, M+1, M+2, ..., 2M, 2M+1\}$
G_i^x	Latitude do nó, dado que $i \in N$.
G_i^y	Longitude do nó, dado que $i \in N$.
q_i	Demanda da requisição, dado que $q_i = -q_{i+M}$ para $i \in P^+$.
e_i	Janela de início de um nó, dado que $i \in N$.
l_i	Janela de término de um nó, dado que $i \in N$.
V	Frota de veículos, dado que $V = \{1, 2, ..., \|V\|\}$.
t_{ij}	Tempo de viajem entre nós, dado que $i, j \in N$.
c_{ij}	Custo de viajem entre nós, dado que $i, j \in N$.
s_i	Tempo de serviço, dado que $i \in N$.
X_{ij}^v	Variável binária dado que $v \in V, i, j \in N$. O valor é 1 se o arco (i, j) está sendo usado pelo veículo v, caso contrário é 0.
T_i	Tempo de início do atendimento de um nó, dado que $i \in N$.
T_0^v	Tempo de partida de um veículo do depósito, dado que $v \in V$.
T_{2M+1}^v	Tempo de retorno de um veículo para o depósito, dado que $v \in V$.
C	Capacidade de carga dos veículos
Y_i	Variável de carga ao término de um atendimento, dado que $i \in P$.
Γ	Custo operacional aplicado para cada veículo em uso.
A_i	Horário de anunciação de uma requisição, dado que $i < M$.

problema. Como o objetivo é reduzir o número de veículos e o custo das rotas, trata-se de um problema de minimização bi-objetivo. Um detalhe em relação ao custo das rotas, é que ele pode ser expresso em função do somatório do tempo de viagem ou da distância percorrida por todas as rotas.

$$\min \quad \sum_{v\in V}\sum_{i\in N}\sum_{j\in N} c_{ij}X_{ij}^{v} + \Gamma \sum_{v\in V}\sum_{j\in P^{+}} X_{0j}^{v} \tag{2.1a}$$

s.t.

$$\sum_{v\in V}\sum_{j\in N} X_{ij}^{v} = 1,\ i \in P^{+}, \tag{2.1b}$$

$$\sum_{j\in N} X_{ij}^{v} - \sum_{j\in N} X_{ji}^{v} = 0,\ i \in P, v \in V, \tag{2.1c}$$

$$\sum_{j\in P^{+}} X_{0j}^{v} \leq 1,\ v \in V, \tag{2.1d}$$

$$\sum_{i\in P^{-}} X_{i,2M+1}^{v} \leq 1,\ v \in V, \tag{2.1e}$$

$$\sum_{j\in N} X_{ij}^{v} - \sum_{j\in N} X_{j,M+i}^{v} = 0,\ i \in P^{+}, v \in V, \tag{2.1f}$$

$$T_i + s_i + t_{i,M+i} \leq T_{M+i},\ i \in P^{+}, \tag{2.1g}$$

$$X_{ij}^{v} = 1 \Rightarrow T_i + s_i + t_{ij} \leq T_j,\ i,j \in P, v \in V, \tag{2.1h}$$

$$X_{0i}^{v} = 1 \Rightarrow T_0^{v} + t_{0i} \leq T_i,\ i \in P^{+}, v \in V, \tag{2.1i}$$

$$X_{i,2M+1}^{v} = 1 \Rightarrow T_i + s_i + t_{i,2M+1} \leq T_{2M+1}^{v},\ i \in P^{-}, v \in V, \tag{2.1j}$$

$$e_i \leq T_i \leq l_i,\ i \in P, \tag{2.1k}$$

$$e_0 \leq T_0^{v} \leq l_0,\ v \in V, \tag{2.1l}$$

$$e_{2M+1} \leq T_{2M+1}^{v} \leq l_{2M+1},\ v \in V, \tag{2.1m}$$

$$X_{ij}^{v} = 1 \Rightarrow Y_i + q_j = Y_j,\ i \in N, j \in P^{+}, v \in V, \tag{2.1n}$$

$$X_{ij}^{v} = 1 \Rightarrow Y_i - q_j = Y_j,\ i \in N, j \in P^{-}, v \in V, \tag{2.1o}$$

$$X_{0j}^{v} = 1 \Rightarrow Y_0 + q_j = Y_j,\ j \in P^{+}, v \in V, \tag{2.1p}$$

$$Y_0 = 0,\ q_i \leq Y_i \leq C,\ i \in P^{+}, \tag{2.1q}$$

$$X_{ij}^{v} \in \{0,1\},\ i,j \in N, v \in V \tag{2.1r}$$

Como a Equação 2.1a é formulada para minimizar dois objetivos, 1) o número de veículos e 2) o custo total do roteamento, a prioridade dos objetivos depende do valor definido para a variável Γ, onde um valor grande irá priorizar o número de veículos, enquanto um valor pequeno irá priorizar o custo do roteamento. A definição do valor de Γ pode variar conforme o problema e deve ser escolhido de acordo com as características do problema. Ainda, a Equação 2.1a deve ser minimizada respeitando às seguintes restrições:

- Restrição 2.1b: todas as coletas devem ser atendidas uma única vez;

- Restrição 2.1c: todo nó associado a uma coleta ou entrega de um veículo deve possuir uma aresta de entrada e saída. Ou seja, um nó de requisição nunca será nem o primeiro nem o último nó de uma rota;
- Restrição 2.1d: todo veículo com rota atribuída deve partir do depósito para um nó de coleta;
- Restrição 2.1e: todos os veículos com rota devem chegar no depósito a partir de um nó de entrega;
- Restrição 2.1f: restrição de paridade, uma requisição deve ser coletada e entregue pelo mesmo veículo;
- Restrição 2.1g: restrição de precedência, toda coleta deve preceder a respectiva entrega;
- Restrição 2.1h, 2.1i e 2.1j: as visitas devem ser atendidas conforme o agendamento;
- Restrição 2.1k, 2.1l e 2.1m: as janelas de tempos devem ser respeitadas;
- Restrição 2.1n, 2.1o e 2.1p: a carga dos veículos deve ser respeitada conforme o agendamento;
- Restrição 2.1q: todas as coletas/entregas devem respeitar as capacidades de carga do veículo;
- Restrição 2.1r: define o domínio de X;

A formalização descrita até o momento somente considera problemas estáticos em que todas informações são conhecidas *a priori* a execução da otimização. A seguir será apresentado o conceito de dinamicidade das requisições dos problemas de VRP.

2.3 ROTEAMENTO DE VEÍCULOS COM REQUISIÇÕES DINÂMICAS

O problema de roteamento de veículos com requisições dinâmicas é um cenário de roteamento sujeito as seguintes características. No início da operação do depósito, os veículos iniciam a operação com rotas previamente calculadas a partir de requisições conhecidas *a priori*. Na sequência, ao longo da operação, requisições dinâmicas podem ser anunciadas, no mundo real essas requisições chegam a uma central de atendimento ou agendamento. Com base nisso, o objetivo do roteamento de veículos com requisições dinâmicas é otimizar a frota de veículos para suportar essas novas requisições, como essas requisições não são conhecidas *a priori*, a central

precisa decidir, em tempo real, se irá reaproveitar veículos que já estão em operação ou alocar novos veículos para realizar esses novos atendimentos (PSARAFTIS, 1995).

Larsen (2000) define a solução de um VRP dinâmico não como um conjunto de rotas, mas sim como uma política que descreve como as rotas devem evoluir conforme a entrada de informações no problema. De forma geral, existem duas formas de tratar os problemas dinâmicos, método *a priori* ou método em tempo real. No método *a priori* são aplicados conhecimentos probabilísticos, com base em registros históricos, para tentar prever o futuro e adaptar o roteamento. Enquanto no método em tempo real as informações são recebidas de forma aleatória, e somente quando uma nova informação é revelada ao sistema ele tenta adaptar o roteamento para incluir o atendimento das novas requisições (LARSEN, 2000).

O estudo da dinamicidade é possível graças aos sistemas de comunicação e geolocalização, em que o avanço tecnológico desse tipo de tecnologia torna sua aplicação mais barata e eficaz (PSARAFTIS, 1995). Essas tecnologias são capazes de coletar informações de geoposicionamento dos veículos e enviar esses dados via internet móvel para os centros de processamento, que por sua vez utilizam as novas informações para buscar rotas mais otimizadas e notificar os veículos de forma online acerca das atualizações (LARSEN, 2000).

Assim como o problema de VRP estático, o problema de VRP dinâmico também é classificado como NP-Difícil. Essa propriedade se mantém porque o problema dinâmico é considerado um caso especial do problema estático. Os problemas dinâmicos pressupõem que as condições serão alteradas em algum momento, dessa forma, se um problema que não sofre alterações é resolvível por meio de um algoritmo de complexidade polinomial, então significa que o problema dinâmico pode ser resolvido em tempo polinomial aplicando o mesmo algoritmo, toda vez que o problema muda (LARSEN, 2000).

Geralmente, o estudo de problemas dinâmicos na literatura considera cenários simulados em detrimento dos cenários reais, isso é importante pois as simulações permitem um estudo mais profundo a cerca das propriedades do problema (LARSEN, 2000). Entretanto, é de suma importância que tais cenários simulados sejam capazes de modelar as características reais, caso contrário, os métodos resolutivos não serão aplicáveis nos problemas práticos do mundo real.

Como esse trabalho foca no problema de coleta e entrega com requisições dinâmicas, a seção seguinte descreve com maior detalhes as suas características.

2.3.1 Requisições dinâmicas

As requisições são solicitações de clientes por serviços. No caso do problema DPDPTW, uma requisição está associada a coleta de um produto em um dado local e a entrega desse produto no local do cliente. Além disso, a coleta e a entrega do produto estão condicionadas as restrições impostas pelo cliente, como: janela de tempo, tempo de serviço e demanda do produto. Uma requisição é dita estática se ela é conhecida *a priori* e ela é dita dinâmica se ela é conhecida *a posteriori*.

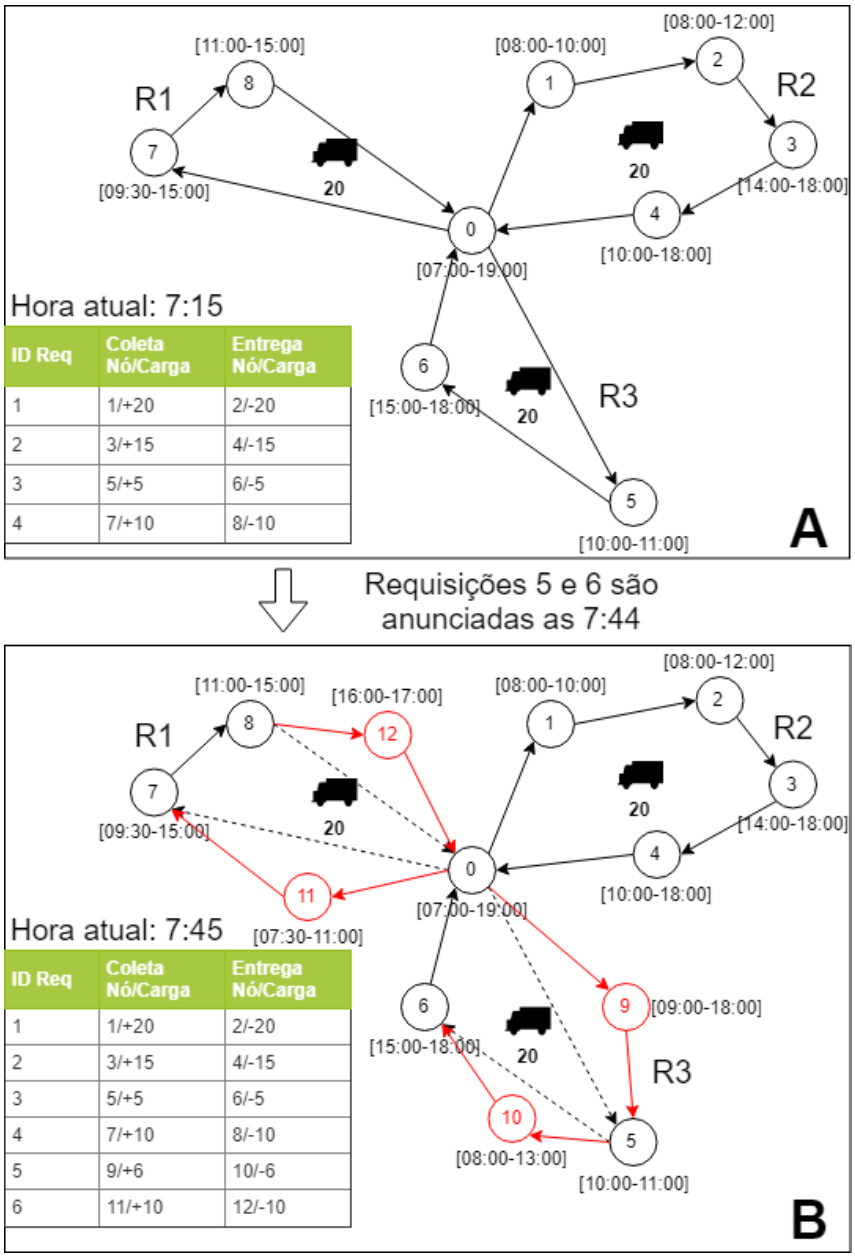
Formalmente, dado uma requisição $i \in R$ (ver Tabela 1), defina $T_{start} = e_D$ como o horário de início da operação do depósito, $T_{end} = l_D$ como o horário de término da operação do depósito e A_i como horário de anunciação de uma requisição i. Uma requisição i é definida como dinâmica (ou conhecida *a posteriori*) se $A_i > 0 \wedge T_{start} < A_i < T_{end}$. E uma requisição i é dita estática (ou conhecida *a priori*) se $A_i \leq T_{start}$. Defina $T_{wd} = T_{end} - T_{start}$ como horizonte do dia de trabalho e $T_{current}$ como o tempo corrente da otimização durante o horizonte do dia de trabalho, onde $T_{start} \leq T_{current} \leq T_{end}$, uma requisição dinâmica i é anunciada para o problema no exato momento que $A_i > T_{current}$.

Como as requisições dinâmicas são anunciadas ao sistema ao longo do horizonte do dia de trabalho, alterando a instância do problema, é requerido que os sistemas de roteamento sejam capazes de se adaptar a nova versão do problema. Por exemplo, suponha que no início do dia de trabalho de uma empresa de distribuição de produtos são conhecidas k requisições *a priori*. Com base nessas requisições um sistema de roteamento determina as melhores rotas para a frota de veículos. Porém, ao longo do dia em um determinado horário, novas requisições foram recebidas e essas alteram o problema em mãos, nesse momento o sistema de roteamento precisa atualizar as rotas de atendimento da frota em operação para atender as novas requisições.

A Figura 2 retrata a adaptação de rotas para problemas dinâmicos. Nessa figura, o roteamento encontrado para os veículos às 7:15 da manhã (figura A) era composto pelas requisições de ID 1, 2, 3 e 4 alocado das rotas R1, R2 e R3. Porém, às 7:44 da manhã duas novas requisições de ID 5 e 6 são anunciadas ao problema, o que torna a solução atual sub-ótima. Nesse momento, o algoritmo de otimização deve alocar as novas requisições para serem atendidas. Nesse exemplo, as 7:45 da manhã a requisição 5 foi alocada na rota R3 e a requisição 6 foi alocada na rota R1, criando uma nova solução (figura B). Se os veículos em operação não pudessem ser reaproveitados, o algoritmo de otimização teria de alocar um novo veículo, definindo uma nova rota, até atender todas as requisições dinâmicas.

Segundo Larsen (2000), nesse tipo de problema os eventos de curto prazo devem ser priorizados. Um evento de curto prazo é uma requisição com urgência de

Figura 2 – Exemplo de roteamento do DPDPTW. A Figura A apresenta a solução vigente às 7:15 horas, enquanto a Figura B apresenta a solução vigente às 7:45, alterado para anteder as requisições dinâmicas 5 e 6 anunciadas as 7:44.



Fonte: Imagem gerada pelo autor.

atendimento alta, ou seja, o início do atendimento não pode ser postergado mais pra frente. Como existe pouca informação sobre o futuro e o mesmo é incerto, é mais interessante focar nos eventos de curto prazo já que os de longo prazo poderão sofrer modificações ou serem re-priorizados quando aparecerem novos eventos de curto prazo.

Um item importante para problemas dinâmicos é a medição do grau de dinamicidade. Essas medidas permitem que seja possível entender melhor as características do problema e ajustar o sistema para diferentes cenários. A seção 2.3.1.1 apresenta

algumas medidas de dinamicidade.

2.3.1.1 Medição geral do grau de dinamicidade

A medição geral do grau de dinamismo das requisições, definida pela Eq. 2.2, considera um valor $0 \leq \Phi_{dod} \leq 1$ que define a porcentagem de requisições dinâmicas que são apresentadas ao sistema ao longo do horizonte de planejamento (LARSEN, 2000). Essa medida também é conhecida como *degree of dynamism* ou Φ_{dod}.

$$\Phi_{dod} = \frac{\textit{Número de requisições dinâmicas}}{\textit{Número total de requisições}} \tag{2.2}$$

Embora a medição Φ_{dod} (Eq. 2.2) expresse de forma adequada a estimativa do grau de dinamismo, essa medida não considera um fator importante do problema, que é o momento de anunciação das requisições ao sistema. Não considerar o tempo de anunciação indica que as requisições que entram próximo do começo ou do final da execução possuem o mesmo grau de dinamismo. Entretanto, quanto mais cedo as requisições são apresentadas, mais tempo o sistema de roteamento terá para encontrar melhores soluções (LARSEN, 2000).

2.3.1.2 Medição do grau de dinamicidade efetiva

Larsen (2000) propõem a medição do grau de dinamismo efetivo, ou *effective degree of dynamism* (Φ_{edod}), que considera o tempo de anúncio de cada requisição. Para isso defina: $[T_{start}, T_{end}]$ como o horizonte de tempo do dia de trabalho, T_i como o horário que o veículo inicia o atendimento do nó i, A_i como o horário que uma requisição é anunciada para o problema, φ como todas as requisições conhecidas *a priori*, onde $A_i \leq 0, i \in \varphi$, ϱ como todas as requisições dinâmicas onde $A_i > 0 \wedge T_i \leq T_{end} \wedge A_i \leq T_i, i \in \varrho$, e, por fim, $|R| = \varphi + \varrho$ representa o número total de requisições do problema. A medida do grau de dinamismo efetivo é definida pela Eq. 2.3.

$$\Phi_{edod} = \frac{1}{|R|} \sum_{i=1}^{\varrho} (\frac{T_i}{T_{end}}) \tag{2.3}$$

O resultado da Eq. 2.3 é interpretado da seguinte forma. Se Φ_{edod} for igual a zero indica que o problema não possui nenhuma dinamicidade, ou seja, é estático. Se Φ_{edod} for igual a um indica que o grau de dinamicidade é máximo, portanto todas as requisições são reveladas no último momento disponível do horizonte de tempo. A aplicação do Φ_{edod} é válida para cenários que não possuem janela de tempo.

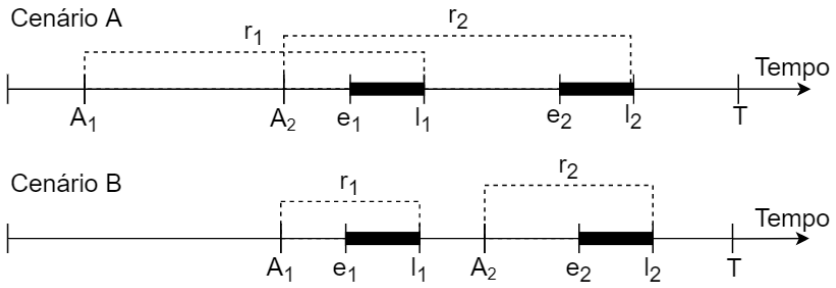
2.3.1.3 Medição do grau de dinamicidade efetiva com janela de tempo

Para resolver o problema da medição do grau de dinamicidade com janela de tempo, Larsen (2000) propõem uma medida que adiciona o tempo de reação (Equação 2.4). O tempo de reação indica quanto tempo existe entre o horário de anunciação e o término da janela de tempo da requisição, ou seja, quanto tempo há disponível para otimizar aquela requisição a partir do momento que ela foi anunciada. Essa medição considera a diferença de tempo entre o término da janela de tempo da coleta ($l_i, i \in R$) e o tempo de anúncio da requisição ($A_i, i \in R$).

$$r_i = l_i - A_i \tag{2.4}$$

Os exemplos da Figura 3 demonstram dois cenários com diferentes tempos de reação (r_1 e r_2) de duas requisições (requisições 1 e 2). No cenário A o tempo de reação é maior e no cenário B o tempo de reação é menor.

Figura 3 – Gráfico do tempo de reação, variável r, usada na medição do grau de dinamicidade. O cenário A tem um tempo de reação maior e o cenário B tem um tempo de reação menor.



Fonte: Imagem adaptada de (LARSEN, 2000) pelo autor.

Segundo Larsen (2000), com base no tempo de reação é possível medir a dinamicidade do problema. Quanto maior o tempo de reação médio do problema menor é a dinamicidade, pois o otimizador tem mais tempo ao longo da execução para otimizar a solução. E quanto menor a média do tempo de reação maior a dinamicidade, porque o otimizador tem menos tempo para otimizar a solução. A Equação 2.5 apresenta o cálculo do Φ_{edodtw}, essa equação calcula a média geral do tempo de reação do problema.

$$\Phi_{edodtw} = \frac{1}{|R|} \sum_{i=1}^{\varrho} (1 - \frac{r_i}{T_{end}}) \tag{2.5}$$

2.3.1.4 Simulação de dinamicidade

O estudo das requisições dinâmicas pode ser feito com cenários reais ou simulados. Em cenários reais são usados dados do dia a dia extraídos por meio de dispositivos de GPS (*Global Positioning Systems*) e serviços de mapeamento como

Google e Waze, enquanto que em cenários simulados a entrada de requisições é modelada por meio de distribuições probabilísticas, como a distribuição de Poisson ou a distribuição aleatória uniforme (LARSEN, 2000).

Para ambientes de requisições dinâmicas simuladas, Pankratz (2005) propõem um *framework* capaz de criar os cenários experimentais utilizando o conceito de horizonte de tempo da simulação. Esse conceito basicamente considera um tempo de início e término, discreto ou contínuo, para determinar quando as requisições dinâmicas serão geradas e anunciadas. Nesse *framework* o horário de inicio e de término da janela de tempo do depósito são usados para definir o intervalo do horizonte de tempo da simulação. No método de tempo discreto o *framework* avança conforme as iterações do algoritmo, enquanto no método contínuo o relógio do computador é usado para a contagem do tempo. Em ambos os casos o tempo da simulação é proporcional ao horizonte de tempo da simulação. Por exemplo, se o depósito possuí uma janela de tempo que começa as 8AM e termina as 5PM então o horizonte de tempo é de 9H, assim a simulação será ajustada para executar considerando esse horizonte de tempo. Em instâncias de benchmark, as janelas de tempo do depósito são dadas como parte do problema, no mundo real as mesmas são definidas de acordo com as políticas das empresas.

Para criar os cenários de *benchmark* do DPDPTW, Pankratz (2005) propõem a adaptação das instâncias de *benchmark* do PDPTW. Essa adaptação adiciona o atributo chamado de tempo de anúncio A_i. O tempo de anúncio define o horário que as requisições serão anunciadas para o problema. Entretanto, o valor de A_i deve ser definido de forma que seja possível atender a requisição i, ou seja, se $A_i = T_{end} - \epsilon$, para ϵ muito pequeno, será impossível alocar um veículo para atender A_i respeitando as janelas de tempo da requisição e do depósito. Para resolver esse problema, (PANKRATZ, 2005) define que A_i deve respeitar um tempo máximo para cada requisição, T_i^{latest}, de forma que se $A_i = T_i^{latest}$, no pior caso um novo veículo partindo do depósito será capaz de fazer o atendimento dentro da janela de tempo e retornar ao depósito.

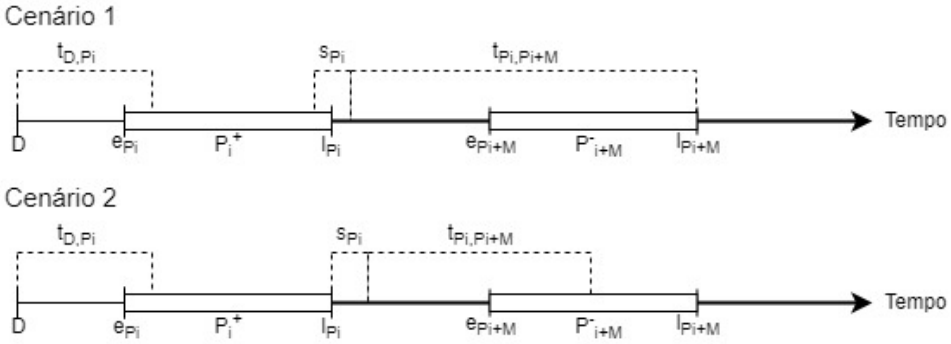
O cálculo do T^{latest} é feito da seguinte forma, dado P como conjunto de todas as coletas e entregas (conforme Tabela 1), $i \in R$ como uma dada requisição, l_{P_i} e $l_{P_{i+M}}$ como a janela de término dos nós de coleta e entrega da requisição i, s_{P_i} como o tempo de serviço da coleta da requisição i, D como o índice do depósito e $t_{P_iP_j}$ o tempo de viajem entre dois nós P_i e P_j, o valor de T_i^{latest}, para cada requisição i, pode ser calculado conforme Equação 2.6.

$$T_i^{latest} = min\{l_{P_i}, l_{P_{i+M}} - t_{P_iP_{i+M}} - s_{P_i}\} - t_{D,P_i}, \quad i \in R \tag{2.6}$$

A Figura 4 apresenta graficamente dois cenários para a Equação 2.6. No *Ce-*

nário 1 é representado o caso onde não existe folga entre os términos das janela de tempo (l_{P_i} não é o valor mínimo), enquanto no *Cenário 2* é representado o caso inverso, em que existe uma folga entre a janela de tempo da entrega (l_{P_i} é o valor mínimo). Nessa definição, considerar os tempos de viagens entre os nós garante que um veículo seja sempre capaz de atender a requisição, mesmo que para isso deva-se alocar um novo veículo. Entretanto, a alocação de novos veículos geralmente leva a dois problemas, um maior custo operacional da solução e o esgotamento da frota de veículos. Nesse caso, adicionar veículos para atender a nova demanda deve ter menor prioridade em relação a alocação das novas requisições a veículos que já estão arranjados (PANKRATZ, 2005).

Figura 4 – Cenários considerados no cálculo de T_i^{latest}. O cenário 1 demonstra o uso da janela de término da entrega ($l_{P_{i+M}}$), enquanto o cenário 2 demonstra o uso da janela de término da coleta (l_{P_i}).



Fonte: Imagem criada pelo autor.

O cálculo do T^{latest} é essencial para a geração do *benchmark* do DPDPTW. Uma vez calculado seu valor, Pankratz (2005) define duas metodologias para gerar os conjuntos de benchmark P1 e P2, ambos com propriedades diferentes (PANKRATZ, 2005). No conjunto P1 as instâncias são variadas em termos de urgência de atendimento, onde o tempo de anuncio de cada requisição (A_i) é atribuído conforme Equação 2.7. Nessa equação o fator a define o grau de urgência do atendimento, respeitando $0 \leq a \leq 1$. Nesse contexto, quanto mais próximo de 1 o fator a menor será o tempo de reação (Equação 2.4) e logo as requisições são mais urgentes. Entretanto, quanto mais próximo de zero o fator a maior será o tempo de reação. Se $a = 0$ o conjunto se comporta como na versão estática do problema onde todas as requisições são conhecidas *a priori*.

$$A_i = a \times T_i^{latest}, \quad \forall i \in P^+ \tag{2.7}$$

Para o conjunto P2 é variado o número de requisições conhecidas *a priori*. Nesse conjunto um fator φ define a porcentagem de requisições estáticas, enquanto as demais são definidas como requisições dinâmicas (fator $\varrho = 1 - \varphi$). As requisições

estáticas são anunciadas no tempo $A_i = 0$, enquanto as requisições dinâmicas são anunciadas no tempo $A_i = T_i^{latest}$ (atribuídas com o grau máximo de urgência, onde $a = 1$). Para reduzir o viés aleatório da distribuição normal, Pankratz (2005) sugere que sejam geradas 10 instâncias de teste para cada valor de φ.

Para gerenciar a anunciação das requisições dinâmicas Veen et al. (2013) propõem o conceito de fatias de tempo, usadas para discretizar os momentos que o sistema de roteamento deve checar por requisições dinâmicas. Durante o avanço temporal, em toda transição entre fatias de tempo, o valor de $T_{current}$ é atualizado e o mecanismo de dinamismo anuncia novas requisições em que o $A_i < T_{current}$. Formalmente, a fórmula define T_{wd} como horizonte de tempo do dia de trabalho (dado por $T_{wd} = T_{end} - T_{start}$), esse valor pode ser um valor contínuo (relógio do computador, ex: 60 segundos) ou um valor discreto (número de iterações dos algoritmos, ex: 1.000), define também F_{ts} como o número de fatias de tempo, onde $0 < F_{ts} \leq T_{wd}$. É importante observar que quanto mais próximo de T_{wd} for F_{ts}, mais real será a simulação. Nesse método, cada fatia de tempo possui um tempo dado por $T_{ts} = T_{wd}/F_{ts}$ unidades de tempo, ou seja, o problema será divido em F_{ts} problemas estáticos com duração de T_{ts}.

O tempo de simulação da dinamicidade precisa ser escalado para que todas as requisições sejam anunciadas no tempo correto. Um forma de atender essa questão é calcular a escala temporal associada às variáveis das requisições (como janela de tempo, tempo de serviço, tempo de viajem, etc.) em relação ao valor do dia de trabalho, T_{wd}. O cálculo dessas escalas geralmente leva em consideração o tempo da janela de tempo do depósito. Dado que o depósito é o ponto de partida e retorno dos veículos, ele define os limites temporais de trabalho dos veículos. Logo, as escalas de tempo das requisições podem ser adaptadas conforme Equação 2.8.

$$scale(t) = \frac{(l_D - e_D)t}{T_{wd}} \tag{2.8}$$

2.3.2 Veículos em movimento

Em problemas dinâmicos um fator importante é a relação da movimentação do veículo em função do tempo. Em um ambiente simulado, considera-se que os veículos estão viajando pelas rotas conforme os tempos de viagem, atendimento e serviço da simulação. Dessa forma, é necessário manter o rastreamento dos pontos que os veículos já visitaram, estão visitando e ainda irão visitar, para que novas requisições sejam alocadas de forma factível, considerando a posição geográfica dos veículos com base no tempo que passou (LARSEN, 2000).

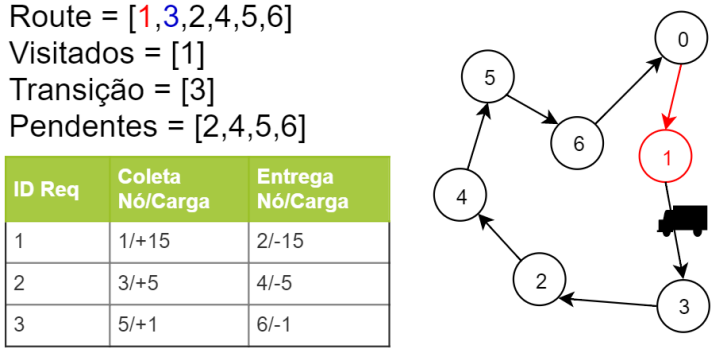
Geralmente, os veículos partem do depósito com rotas pré-definidas pelo sis-

tema de roteamento. Essas rotas são construídas com base nas requisições conhecidas *a priori*. Em um ambiente de simulação extremo em que todas as requisições são dinâmicas, os veículos aguardam até a primeira requisição dinâmica ser anunciada para gerar a primeira rota (NECULA; BREABAN; RASCHIP, 2017). A partir do momento que o primeiro veículo sai do depósito, se novas requisições dinâmicas forem anunciadas, em um primeiro momento, o sistema de roteamento irá tentar re-arranjá-las nas rotas já atribuídas aos veículos em operação, caso isso não seja possível, então o sistema de roteamento irá alocar novos veículos.

O veículo em movimento impõem a restrição de fixar os nós por ele já visitado. Isso quer dizer que, quando um veículo terminou de visitar um nó (vértice) da sua rota, aquele nó não poderá ser realocado para outro veículo, nem em outra posição da rota, porque ele já foi atendido (NECULA; BREABAN; RASCHIP, 2017). Além disso, no caso do PDPTW, quando um ponto de coleta é atendido por um dado veículo, aquele veículo deverá ser o mesmo a fazer a entrega daquela carga, ou seja, depois que uma coleta for feita a respectiva requisição não pode ser realocada para outro veículo.

A Figura 5 apresenta um exemplo em que o sistema de roteamento determinou a seguinte rota para o veículo: $[1, 3, 2, 4, 5, 6]$ (a visita será feita nessa sequência). Como o veículo está avançando pela rota ao longo do tempo, ele já visitou o nó 1 e está a caminho do nó 3. Dessa forma, o nó 1 e 3 não podem ser re-arranjados, seja na mesma ou em outras rotas. Como esse exemplo trata do problema PDPTW, e os nós 2 e 4 são os pontos de entrega dos nós 1 e 3, eles não podem ser re-arranjados para outras rotas, porém, podem ser re-arranjados na mesma rota respeitando os limites da parte pendente. Finalmente, como os nós 5 e 6 não foram visitados e representam a requisição 3, esses nós podem ser re-arranjados na parte pendente da mesma rota ou em outra rota.

Figura 5 – Exemplo de otimização online com veículos em movimento. Nesse exemplo, somente os nós pendentes 2, 4, 5 e 6 podem ser re-arranjados porque não foram visitados e o veículo não está a caminho para realizar a visita (transição).



Fonte: Imagem criada pelo autor.

Com o passar do tempo o número de nós a serem roteados pelo sistema reduz, o que torna também a busca pela solução menos complexa. Entretanto, duas

novas complexidades são adicionadas. Primeiro, escolhas ruins feitas no começo do roteamento podem impactar no custo futuro dos veículos dado que um nó já visitado não pode ser realocado. Segundo, o tempo de resposta do sistema de roteamento deve ser menor para poder responder às mudanças enquanto há tempo antes dos nós serem visitados (SCHMITT; BALDO; PARPINELLI, 2018).

2.3.3 Dificuldades da otimização de problemas dinâmicos

Um dos fatores chave no sucesso dos sistemas de roteamento aplicados no contexto de problemas dinâmicos, é sua capacidade de se manter atualizado apesar do surgimento de novos eventos. Caso o sistema não se mantenha atualizado, os algoritmos combinatoriais usados por eles podem encontrar soluções ruins para o problema, ou mesmo não achar soluções factíveis (LARSEN, 2000).

Os problemas dinâmicos implicam que os sistemas sejam capazes de se adaptar com baixo tempo de resposta. Essa característica é importante porque em cenários com entrada de novas requisições, acidentes de trânsito, mudanças de tráfego, cancelamento de requisições ou outros eventos inesperados, pode haver pouco tempo para tomar uma decisão e atualizar as rotas dos veículos (LARSEN, 2000).

2.4 HEURÍSTICA DE BUSCA LARGA ADAPTATIVA EM VIZINHANÇA (ALNS)

As heurísticas de busca local (LS - *Local Search*) são geralmente usadas para modificar pequenas partes da solução com o objetivo de melhoramento. No problema de roteamento de veículos, essas pequenas modificações alteram a ordem de visita de um determinado cliente ou trocam o veículo de atendimento de uma determinada requisição (PISINGER; ROPKE, 2007).

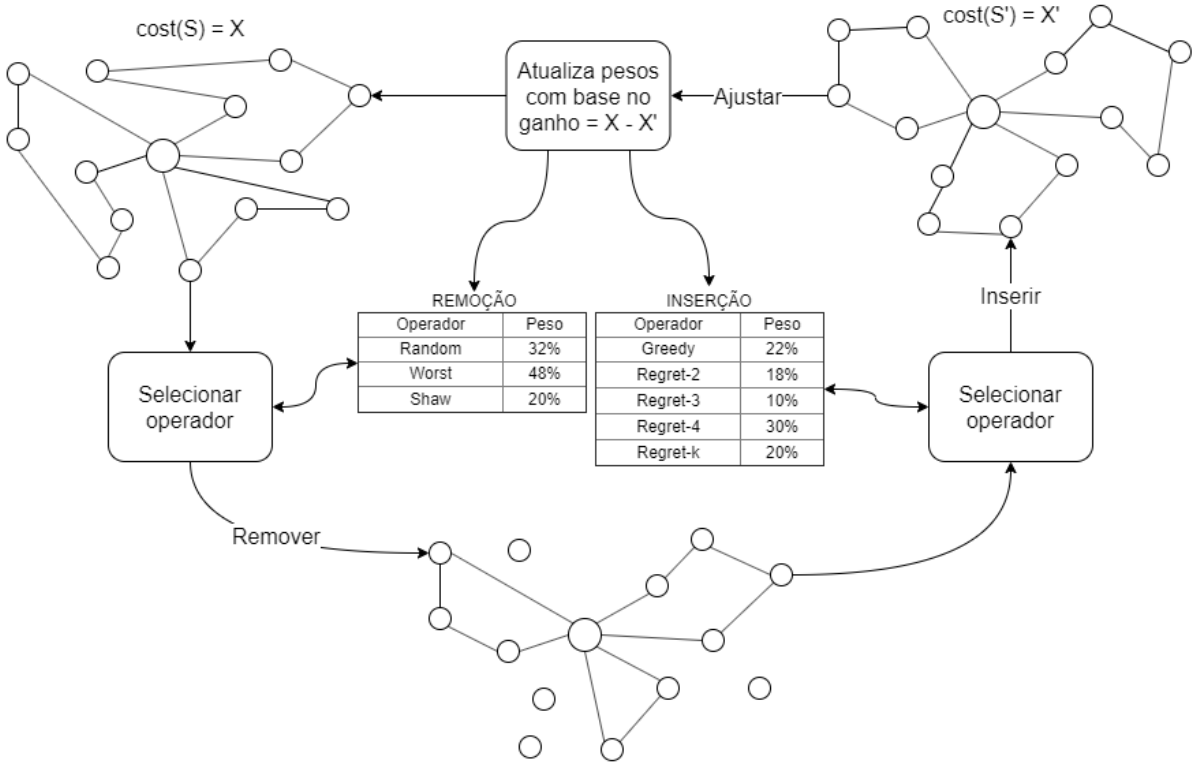
Entretanto, quando as modificações são pequenas o algoritmo tem mais dificuldade em explorar o espaço de busca e, nesses casos, eles tendem a ficar presos em ótimos locais. Para superar essa dificuldade, Shaw (1998) propôs o algoritmo de busca larga em vizinhança (LNS - *Large Neighborhood Search*). Esse algoritmo é diferente dos demais por modificar em maior escala as soluções, em que as alterações variam na faixa de 30% à 40% da solução (PISINGER; ROPKE, 2007).

O algoritmo LNS é baseado no conceito de relaxamento e re-otimização. Basicamente, o algoritmo parte de uma solução base, aplica uma heurística de remoção para remover um conjunto de requisições e depois aplica uma heurística de inserção para tentar re-inserir as requisições em posições melhores da solução. Portanto, durante uma iteração do processo de otimização o algoritmo destrói e repara a solução. Caso a nova solução seja melhor ela substitui a anterior (SHAW, 1998).

Nesse processo de geração de novas soluções, um ponto importante é a exploração do espaço de busca. A exploração no LNS é feita por meio da heurística do *Simulated Annealing* (SA) (KIRKPATRICK; GELATT; VECCHI, 1983). Isso permite que soluções de pior custo possam ser aceitas como novas soluções. A ideia é que soluções de pior custo possam abrir atalhos para áreas mais promissoras do espaço de busca (ver Seção 2.4.5).

Pisinger e Ropke (2007) propõem uma extensão ao LNS, a heurística de busca em largura adaptativa, ou ALNS (*Adaptive Large Neighborhood Search*). A principal diferença é que enquanto o LNS usa uma única heurística de remoção e inserção, o ALNS pode usar múltiplas heurísticas de cada tipo. A seleção dos operadores é feita com base em um critério de sucesso. A Figura 6 demonstra um esquema básico de funcionamento do ALNS. Vale ressaltar que essa versão do algoritmo, até onde se conhece, não foi proposta para problemas dinâmicos.

Figura 6 – Funcionamento básico do algoritmo de busca larga adaptativa, ALNS. Demonstra o ciclo de remoção e re-inserção de requisições com objetivo de otimização. As tabelas centrais representam os operadores de remoção e inserção ponderados pelo ganho.



Fonte: Imagem criada pelo autor.

Durante uma iteração do processo de otimização o ALNS seleciona um operador de remoção e um operador de inserção. Essa seleção é feita de forma probabilista com base em um vetor de pesos dos operadores, em que a probabilidade segue uma distribuição uniformemente variada. Nesse processo, quando uma nova solução é gerada, um contador de uso e uma pontuação são dados aos operadores utilizados.

Porém, a atualização dos pesos não ocorre a cada iteração, um segmento de tempo define quantas iterações devem ocorrer antes de atualizar os pesos, o tamanho desse segmento deve ser alto (aproximadamente 100 iterações) para não afetar o desempenho do algoritmo. A importância desse mecanismo é favorecer as heurísticas que resultam em melhores soluções ao longo do tempo (PISINGER; ROPKE, 2007). Outro fator importante é que diferentes problemas requerem diferentes estratégias de otimização, então os operadores selecionados devem ter estratégias diferentes para que o ALNS possa se adaptar aos diferentes cenários (LUTZ, 2014).

Uma visão geral do ALNS é apresentada no Algoritmo 1 (PISINGER; ROPKE, 2007; LUTZ, 2014). No início, o algoritmo cria uma solução inicial S (linha 1) e a define como melhor solução (linha 2). Em seguida, o algoritmo inicializa o *Simulated Annealing* junto com os vetores de pesos, pontuações e contagens de uso dos operadores (linha 3). A cada iteração do algoritmo (linha 4), é feita uma cópia da solução atual (linha 5), definido o número de requisições para remoção (linha 6), selecionado um operador de remoção, inserção e ruído, conforme Equação 2.9 (linhas 7 à 9), e aplicado o processo de remoção e a inserção (linhas 10 e 11). Se a nova solução for aceita (linha 12), então a solução local S é atualizada. Se a solução for uma nova melhor (linha 14), então S_{best} é atualizada (linha 15). Com base na solução gerada, a pontuação dos operadores é acrescida conforme o seguinte critério (linha 18): 1) soma-se θ_1 se a nova solução for uma melhor global, 2) soma-se θ_2 se a solução for uma melhor local e 3) soma-se θ_3 se a solução for pior mas aceita pelo SA. Se um determinado número de iterações foram executadas (linha 19) atualiza-se os pesos dos operadores conforme Equação 2.10 (linha 20). Por fim, executa-se o controle de temperatura do *Simulated Annealing* (linha 22).

O algoritmo ALNS utiliza a seleção por roleta para definir os operadores a serem aplicados em cada iteração. A seleção é feita utilizando os pesos dos operadores de remoção (O^{rem}), inserção (O^{ins}) e de ruído (O^{noise}), conforme Equação 2.9. Esse procedimento utiliza o vetor de pesos, onde cada índice do vetor corresponde a um operador e o valor numérico daquela posição indica o peso daquele operador. Nesse processo, o peso é usado para favorecer operadores que foram bons no passado e que deveriam ser usados mais vezes no futuro (LUTZ, 2014).

$$p(i) = \frac{w_i}{\sum_{j=1}^{k} w_j} \tag{2.9}$$

A atualização dos pesos dos operadores é feita de forma periódica a cada segmento k de iterações. Nesses cálculos são usadas as informações de contagem de uso ($u_{O^{rem}}$, $u_{O^{ins}}$ e $u_{O^{noise}}$) e a pontuação ($\phi_{O^{rem}}$, $\phi_{O^{ins}}$ e $\phi_{O^{noise}}$) dos operadores armazenadas durante o último segmento. Esses valores são combinados de forma a

Algoritmo 1 Algoritmo de busca larga adaptativa em vizinhança (ALNS).

Require: I: Instância do problema, γ: taxa de resfriamento
1: Crie uma solução inicial S
2: $S_{best} \leftarrow S$
3: Inicializar controles do método de aceitação e pesos dos operadores
4: **while** Critério de parada $<>$ verdadeiro **do**
5: $S' \leftarrow$ cópia da solução S
6: $y \leftarrow$ gerar número de requisições para remoção
7: $i \leftarrow$Selecionar operador de remoção Eq. 2.9 de O^{rem}
8: $j \leftarrow$Selecionar operador de inserção Eq. 2.9 de O^{ins}
9: $n \leftarrow$Selecionar uso de ruído Eq. 2.9 de O^{noise}
10: Remova y requisições de S' usando o operador O_i^{rem}
11: Insira as requisições removidas em S' usando o operador O_j^{ins} e O_n^{noise}
12: **if** $accept(S', S)$ **then**
13: $S = S'$
14: **if** $cost(S') < cost(S_{best})$ **then**
15: $S_{best} \leftarrow S'$
16: **end if**
17: **end if**
18: Ajustar pontuações dos operadores O_i^{rem} e O_j^{ins}
19: **if** $updateWeights()$ **then**
20: Ajustar pesos de todos operadores $\omega_{O^{rem}}$ e $\omega_{O^{ins}}$ de acordo com 2.10
21: **end if**
22: Executar rotinas de controle do método de aceitação
23: **end while**

estimar os pesos considerando o histórico de uso do último segmento com o histórico dos segmentos anteriores. Esses dois tipos de históricos são balanceados por um fator de reação ρ, que determina o equilíbrio entre o passado e o presente (LUTZ, 2014). A Equação 2.10 demonstra o cálculo de atualização dos pesos, onde ω_{O_i} representa o peso do operador i, dado que $i \in \omega_{O^{rem}} \vee \omega_{O^{ins}} \vee \omega_{O^{noise}}$, ρ representa o fator de reação, tal que $0 < \rho < 1$, ϕ_{O_i} representa a pontuação do operador i e u_{O_i} representa a contagem de uso do operador i.

$$w(i) = \begin{cases} (1-\rho)w_{O_i} + \rho\frac{\phi_{O_i}}{u_{O_i}}, & \text{se } u_{O_i} > 0 \\ (1-\rho)w_{O_i}, & \text{se } u_{O_i} = 0 \end{cases} \tag{2.10}$$

Embora o sistema adaptativo de pesos do ALNS seja útil para selecionar os melhores operadores durante a sua execução, isso não é suficiente para um bom desempenho do algoritmo. Tanto no LNS quanto no ALNS, existem operadores que são construídos pensando no problema de roteamento de veículo, ou seja, utilizam variáveis e medidas do problema para executar as inserções e remoções. No ALNS diversas variáveis são dadas *a priori* pelo problema, como posicionamento geográfico

e demanda de carga, e outras são calculadas a partir de medidas do problema como tempo de partida e tempo de chegada dos veículos nos nós. As variáveis e as medidas podem ser combinadas para encontrar similaridades de forma mais eficiente na frota de veículos. Segundo Shaw (1998), um bom exemplo de relação é a combinação do posicionamento geográfico entre os nós a serem visitados. Esse tipo de relação resulta em remoções e inserções mais eficientes. Em respeito as medidas calculadas, a seguir serão apresentadas as principais medidas do VRP e em seguida será apresentado os principais operadores.

2.4.1 Medição dos agendamentos de tempo

Em respeito as medidas calculadas, nessa seção serão apresentadas as medidas de tempo de chegada, partida, espera e folga, necessárias para a implementação dos operadores usados no ALNS para o VRPTW. Serão apresentadas as medidas de: tempo de chegada, tempo de partida, tempo de espera e tempo de folga (SAVELSBERGH, 1992; LUTZ, 2014; NACCACHE; CÔTÉ; COELHO, 2018; AZIEZ; CôTé; COELHO, 2019).

2.4.1.1 Tempo de chegada, tempo de partida e tempo de espera

As medidas de tempo são usadas para indicar o momento que o veículo chega, espera, e parte de um determinado local de atendimento. Elas são necessárias durante a execução do ALNS para validar se as restrições de janela de tempo estão sendo atendidas ou se existe espaço para alocação de novas requisições. Além disso, o indexação desses valores reduz a complexidade de execução dos operadores, porque com os valores pré-calculados os algoritmos podem testar a permutação e a inserção de novos nós nas rotas mais rapidamente (LUTZ, 2014).

O cálculo dessas medidas é realizado separadamente para cada veículo, assumindo a rota pela qual um dado veículo irá passar. O cálculo do tempo de chegada, variável $T^{arrival}$, e do tempo de partida, variável T^{depart}, para cada nó de uma dada rota é dado pelas Equações 2.11 e 2.12, respectivamente. Onde i representa índice do nó na rota do veículo, $t_{i-1,i}$ representa o tempo de viajem entre dois nós, e_i representa o início da janela de tempo e s_i representa o tempo de serviço. Esses valores são calculados de forma encadeada, ou seja, para calcular o $T_i^{arrival}$ de um dado nó i é necessário o tempo de partida T_{i-1}^{depart} do nó anterior $i-1$ (LUTZ, 2014).

$$T_i^{arrival} = T_{i-1}^{depart} + t_{i-1,i} \tag{2.11}$$

$$T_i^{depart} = \max\{T_i^{arrival}, e_i\} + s_i \tag{2.12}$$

Além dos tempos de chegada e partida, a medida do tempo de espera também é utilizada pelos operadores. Essa medida indica o tempo que um veículo aguarda até iniciar o atendimento de um determinado nó quando ele chega no local antes do inicio da janela de tempo. O cálculo desse tempo é dado pela Equação 2.13, onde i representa o nó de visita, e_i é o início da janela de tempo e $T_i^{arrival}$ o tempo de chegada. A medida do tempo de espera é importante porque ela indica ociosidade do veículo, ou seja, lacunas de tempo que possibilitam a inserção de novas requisições na rota (LUTZ, 2014).

$$T_i^{wait} = \max\{0, e_i - T_i^{arrival}\} \tag{2.13}$$

2.4.1.2 Tempo de folga

O calculo do tempo de folga é usado no processo combinatorial para validar se um nó de coleta ou entrega pode ser alocado em outra sequência de uma rota. O tempo de folga é uma medida T^{slack} calculada para cada nó i de uma dada rota. Essa folga, representa o valor máximo que o tempo de partida T^{depart} pode ser atrasado de forma que o veículo consiga atender todos os próximos clientes sem atrasar nenhum término das janelas de tempo. Essa folga indica um espaço de tempo flexível que pode ser usado para alocar novas requisições em diferentes posições da rota de um veículo (SAVELSBERGH, 1992).

O cálculo do tempo de folga para cada nó de uma dada rota requer que os tempos de partida (T^{depart}) e espera (T^{wait}) já tenham sido calculados. Se essa condição for verdadeira, o tempo de folga pode ser calculado percorrendo a rota na direção reversa (de trás pra frente) uma única vez. Para um dado nó i, o tempo de folga é definido formalmente pela Equação 2.14, onde T_{i+1}^{wait} é o tempo de espera do próximo nó, T_{i+1}^{slack} é o tempo de folga do próximo nó, l_i é o término da janela de tempo, T_i^{depart} é o tempo de partida e s_i é o tempo de serviço (SAVELSBERGH, 1992).

$$T_i^{slack} = T_{i+1}^{wait} + \min\{T_{i+1}^{slack}, l_i - T_i^{depart} + s_i\} \tag{2.14}$$

2.4.2 Operadores de remoção

Os operadores de remoção são usados pelo ALNS para remover requisições das rotas de uma solução. O princípio básico é buscar remoções que possam abrir espaço para inserções mais eficientes. Os operadores de remoção utilizados pelo ALNS geralmente dependem que um número y seja gerado para indicar quantas requisições serão removidas. Existem diversas estratégias para gerar esse número. Em (ROPKE; PISINGER, 2006) é apresentado uma formulação aplicada ao problema do PDPTW,

onde y é gerado aleatoriamente conforme a Equação 2.15, dado que $|R|$ representa o número de requisições e ξ é um parâmetro de controle.

$$4 \leq y \leq min(100, \xi|R|) \tag{2.15}$$

Diversos operadores já foram propostos na literatura utilizando diferentes heurísticas para o ALNS. Nas próximas subseções serão apresentados os principais operadores de remoção encontrados na literatura.

2.4.2.1 Remoção aleatória

A remoção aleatória é a heurística de remoção mais simples, ela recebe um número y de requisições para remoção e uma solução S composta por vários veículos. Com base nesses dados, um processo iterativo se inicia, de forma que a cada iteração a heurística selecione e remova aleatoriamente uma requisição da solução, até que y requisições tenham sido removidas. A ideia dessa heurística é aumentar a exploração do algoritmo facilitando a fuga de ótimos locais, já que a mesma não considera nenhum tipo de critério para selecionar qual requisição deve ser removida (LUTZ, 2014).

O Algoritmo 2 descreve o processo de remoção aleatória. O algoritmo recebe como parâmetro de entrada uma solução S e o número de requisições y para serem removidas. No inicio é criada uma coleção D para armazenar as requisições removidas. Após isso, inicia-se o laço iterativo, em que, a cada iteração, é selecionada uma requisição r aleatoriamente em S, ela é adicionada à coleção D e removida da solução S. O algoritmo continua até que y requisições tenham sido removidas (PISINGER; ROPKE, 2007; NACCACHE; CÔTÉ; COELHO, 2018).

Algoritmo 2 Algoritmo de remoção aleatória (*Random removal*).

Require: Solução S, número de requisições para remoção y
1: $D \leftarrow \{\}$
2: **while** $|D| < y$ **do**
3: $\quad r \leftarrow$ selecione uma requisição aleatoriamente de S
4: $\quad D = D \cup \{r\}$
5: $\quad S = S \setminus \{r\}$
6: **end while**

2.4.2.2 Remoção de pior custo

O operador de pior custo é uma heurística que busca remover requisições que agregam maior incremento no custo da solução, ou seja, requisições que se removidas, resultam em um maior ganho na redução do custo da solução. Nessa heurística,

o ganho da remoção de uma requisição r é dado pela Equação 2.16, onde $cost(S)$ é o custo atual da solução S e $cost(S \setminus \{r\})$ é o custo da solução S sem a requisição r. É importante notar que a remoção da requisição r representa a remoção dos seus nós de coleta e entrega (LUTZ, 2014).

$$gain(S, r) = cost(S) - cost(S \setminus \{r\}), \forall r \in S \tag{2.16}$$

No PDPTW, uma requisição r é composta de um nó de coleta e um nó de entrega, indexados na posição i e j da rota de um veículo, respectivamente. Com base nisso, o cálculo do ganho associado a uma remoção pode ser feito em tempo constante $O(1)$. A Equação 2.17 calcula a remoção diferentemente dependendo se os dois nós são, ou não, adjacentes (LUTZ, 2014).

$$gain(i, j) = \begin{cases} t_{i-1,i} + t_{ij} + t_{j,j+1} - t_{i-1,j+1}, & \text{se } i \text{ e } j \text{ são adj.} \\ t_{i-1,i} + t_{i,i+1} + t_{j-1,j} + t_{j,j+1} - t_{i-1,i+1} - t_{j-1,j+1}, & \text{se } i \text{ e } j \text{ não são adj.} \end{cases} \tag{2.17}$$

Nesse operador, toda vez que uma requisição é removida, o ganho obtido é aplicado sobre a solução para atualizar o seu custo. Esse processo é repetido até que y requisições tenham sido removidas. A ideia desse operador é remover das rotas requisições que estejam desviando muito o veículo do trajeto (ou seja requisições que estão alocadas no lugar errado), e assim realocá-las em outras rotas (ou posições) para reduzir o custo total do roteamento (PISINGER; ROPKE, 2007).

O Algoritmo 3 descreve o processo. Os parâmetros de entrada são a solução S, um número y de requisições a serem removidas e um parâmetro p para adicionar aleatoriedade no processo. O algoritmo inicia criando uma coleção de elementos removidos D (linha 1). Após isso, a cada iteração, é ordenado um vetor L, de forma descendente, com o ganho de remoção de cada requisição r em L (linhas 3-4). Então, é selecionada uma requisição do vetor considerando alguma aleatoriedade adicionada pelo parâmetro p, e então a requisição é removida (linhas 5-6). Esse algoritmo continua até que y requisições sejam removidas da solução (PISINGER; ROPKE, 2007).

Uma variação do operador de remoção de pior custo é usada em Naccache, Côté e Coelho (2018). Na proposta, os autores consideram o problema do MPDPTW (*Multi-Pickup and Delivery Problem with Time Windows*), onde uma requisição pode ter mais de um ponto de coleta. Nesse caso, duas heurísticas para realizar o cálculo da pior remoção são aplicadas. A primeira considera o ganho em função do nó com maior redução no custo da rota enquanto a segunda considera o custo da rota sem uma requisição por inteira.

Algoritmo 3 Algoritmo de remoção de pior custo (*Worst removal*).

Require: Solução S, y, $p \in R_+$
1: $D \leftarrow \{\}$
2: **while** $|D| < y$ **do**
3: $L \leftarrow$ todas requisições de S não em D
4: Ordene L de forma descendente usando a função $gain()$
5: $y \leftarrow$ um número aleatório entre 0 e 1
6: $D = D \cup L[y^p|L|]$
7: Remove de S as requisições de D
8: **end while**

2.4.2.3 Remoção relacional

O operador de remoção relacional é uma heurística que foca em remover requisições com maior grau de similaridade entre si. A ideia é que removendo tais requisições, seria possível fazer trocas entre as alocações e encontrar melhores soluções. Ou seja, essa heurística tenta resolver o problema onde a remoção de requisições muito diferentes não resulta em nenhuma troca, visto que elas só poderiam ser re-inseridas nas mesmas posições. A heurística de remoção relacional de Shaw define uma *medida de relação* $shaw(i, j)$ para duas requisições i e j arbitrárias. Assim, quanto menor o valor de $shaw(i, j)$ maior o grau de relação (PISINGER; ROPKE, 2007).

O cálculo da medida de relação pode ser feito combinando diferentes métricas do problema. Em Pisinger e Ropke (2007) essa medida é calculada conforme a Equação 2.18, onde χ, χ', χ'' e χ''' são pesos usados para cada termo, $t_{P_i^+,P_j^+}$ é a distância entre dois pontos de coleta, $t_{P_{i+M}^-,P_{j+M}^-}$ é a distância entre dois pontos de entrega, $T_{P_i^+}^{arrival}$ é o tempo de chegada em um ponto de coleta, $T_{P_{i+M}^-}^{arrival}$ é o tempo de chegada em um ponto de entrega, $q_{P_i^+}$ é a demanda associada a coleta P_i^+, e $\psi_{P_i^+}$ indica a quantidade de veículos que atendem aquele tipo de carga.

$$
\begin{aligned}
shaw(i,j) = {} & \chi(t_{P_i^+,P_j^+} + t_{P_{i+M}^-,P_{j+M}^-}) + \\
& \chi'(|T_{P_i^+}^{arrival} - T_{P_j^+}^{arrival}| + |T_{P_{i+M}^-}^{arrival} - T_{P_{j+M}^-}^{arrival}|) + \\
& \chi''|q_{P_i^+} - q_{P_j^+}| + \\
& \chi'''(1 - \frac{|\psi_{P_i^+} \cap \psi_{P_j^+}|}{\min\{|\psi_{P_i^+}|, |\psi_{P_j^+}|\}})
\end{aligned} \tag{2.18}
$$

Na Equação 2.18 os pesos possuem as seguintes responsabilidades: χ pondera a proximidade de distância, χ' pondera a similaridade do tempo de chegada, χ'' pondera a similaridade de carga e χ''' pondera a quantidade de veículos que podem

atender ambas as requisições. Porém, os pesos relacionados a cada termo da equação podem variar de problema para problema. Por exemplo, no problema de coleta e entrega com frota homogênea não faz sentido considerar o quarto termo (χ'''), visto que todo os veículos possuem as mesmas características (PISINGER; ROPKE, 2007).

Em Pisinger e Ropke (2007) a remoção relacional é apresentada conforme Algoritmo 4. Esse algoritmo recebe como parâmetro uma solução S, um número y que define o número de requisições a serem removidas e um parâmetro p para adicionar aleatoriedade no processo. Esse algoritmo inicia selecionando uma requisição de referência (linha 1) e adiciona essa requisição ao conjunto de requisições removidas D (linha 2). Após isso ele inicia o laço iterativo (linha 3). A cada iteração, é selecionada uma requisição de referência r de D (linha 4) e calculada a sua similaridade com as requisições em S (linha 5-6), onde $shaw(i, j)$ é definido de acordo com a Equação 2.18. Então, seleciona-se uma requisição do vetor L adicionando algum grau de aleatoriedade com o parâmetro p (linha 7-8). O processo contínua até que y requisições tenham sido removidas.

Algoritmo 4 Algoritmo de remoção relacional (*Shaw removal*).

Require: Solução S, $y \in N$
1: $r \leftarrow$ remova uma requisição selecionada aleatoriamente em S
2: $D \leftarrow \{r\}$
3: **while** $|D| < y$ **do**
4: $r \leftarrow$ uma requisição selecionada aleatoriamente de D
5: $L \leftarrow$ todas requisições de S não em D
6: Ordene L tal qual $i < j \Rightarrow shaw(r, L[i]) < shaw(r, L[j])$
7: $x \leftarrow$ um número aleatório entre 0 e 1
8: $D = D \cup L[x^p|L|]$
9: Remove de S as requisições de D
10: **end while**

2.4.3 Operadores de inserção

Os operadores de inserção são executados pelo ALNS logo após os operadores de remoção. Basicamente, assim que y requisições foram removidas pelo ALNS usando um operador de remoção, o próximo passo é tentar inseri-las novamente de forma mais otimizada. Essa etapa é realizada pelos operadores de inserção, que são heurísticas que buscam melhoramento do custo fazendo inserções mais apropriadas.

Como os operadores de inserção são concebidos para inserir requisições na solução, uma sub-rotina de calculo do custo da inserção é necessária. Essa rotina calcula o aumento de custo que cada requisição r, não atendida, irá causar para cada veículo da solução S. O cálculo de custo provê informação para determinar a próxima inserção a ser feita. O Algoritmo 5 apresenta a sub-rotina em questão para o problema

do PDPTW (PISINGER; ROPKE, 2007; NACCACHE; CÔTÉ; COELHO, 2018). O algoritmo inicia recebendo uma requisição r para inserção e um veículo v, então ele avalia a melhor posição para inserir a coleta de r em v dado por $pos_{P_r^+}$ (linha 1). Se uma posição não pode ser encontrada (linha 2), o algoritmo identifica inserção não possível e retorna (linha 3), caso contrário ele avalia a melhor inserção da entrega de r em v dada por $pos_{P_{r+M}^-}$ (linha 5). Se uma inserção não pode ser encontrada (linha 6) o algoritmo identifica inserção não possível e retorna (linha 7). Se ambos os pontos puderem ser inseridos o algoritmo calcula o custo da nova rota e retorna (linha 9-10).

Algoritmo 5 Algoritmo para avaliar a inserção de uma determinada requisição r em um veículo v

Require: Requisição r para inserção, veículo v para avaliação
1: $pos_{P_r^+} \leftarrow$ melhor posição para inserir P_r^+ no veículo v.
2: **if** $pos_{P_r^+} = null$ **then**
3: Retorne solução não factível
4: **end if**
5: $pos_{P_{r+M}^-} \leftarrow$ melhor posição para inserir P_{r+M}^- no veículo v.
6: **if** $pos_{P_{r+M}^-} = null$ **then**
7: Retorne solução não factível
8: **end if**
9: $cost \leftarrow$ custo de solução S com a requisição r nas posições $pos_{P_r^+}$ e $pos_{P_{r+M}^-}$
10: Retorne $\{pos_{P_r^+}, pos_{P_{r+M}^-}, cost\}$

O Algoritmo 6 calcula a melhor inserção de um nó i em um veículo v. Isso é feito testando diversas posições para um novo nó (variável i) em cada posição da rota do veículo v, até encontrar a melhor posição. O algoritmo inicia configurando os ponteiros (linhas 1-4) e após isso o mesmo entra no laço iterativo de avaliação das posições, esse laço segue percorrendo a rota de forma incremental. Dentro do laço, ele calcula o tempo de chegada no nó i se o mesmo fosse inserido depois da posição corrente (linha 6), se o término da janela de tempo não for respeitado (linha 7) o algoritmo retorna solução não factível. Na sequência, o algoritmo calcula a diferença de tempo considerando o nó i entre os nós $prev$ e $next$ (linhas 10-12). Se essa inserção consome mais tempo do que o tempo ocioso do veículo, o algoritmo continua o laço a partir da próxima posição (linhas 13-17). Porém, se existe espaço para inserir o nó, o algoritmo calcula o acréscimo de tempo daquela inserção (linha 18). Quando o custo daquela inserção é melhor e se a rota continua factível, o algoritmo armazena aquela posição como uma nova melhor posição (linhas 19-25), e, após isso, avança para o próximo nó da rota, testando assim as demais posições (linhas 26-27).

A eficiência da execução do Algoritmo 6 depende do cálculo dos valores das medidas temporais apresentados na Seção 2.4.1. Quando todos esses valores estão calculados e armazenados em estruturas apropriadas, o algoritmo pode avaliar a in-

Algoritmo 6 Algoritmo para determinar a melhor posição de inserção de um nó i em um veículo v

Require: Nó i para inserção, rota R do veículo v para avaliação

$\Delta^* \leftarrow \infty$
$pos^* \leftarrow null$
$prev \leftarrow$ 0 (nó do depósito)
$next \leftarrow$ primeiro nó de coleta da rota v
while $prev <> size(v)$ **do**
    $t \leftarrow$ tempo de chegada em i a partir de R_{prev}
    if $t > l_i$ **then**
        Retorne inserção não factível
    end if
    $t_{next} \leftarrow$ tempo de chegada em R_{next} a partir de R_{prev}
    $t'_{next} \leftarrow$ tempo de chegada em R_{next} a partir de R_{prev} passando por i
    $\epsilon \leftarrow t'_{next} - t_{next}$ ▷ Duração adicionada
    if $t_{next} > l_{R_{next}} \vee \epsilon > T^{slack}_{next}$ **then**
        $prev \leftarrow next$
        $next \leftarrow next + 1$
        Volte enquanto
    end if
    $\Delta \leftarrow c_{prev,i} + c_{i,next} - c_{prev,next}$
    if $\Delta < \Delta^*$ **then**
        $S' \leftarrow$ Insira i depois de $prev$ em S
        if S' é factível **then**
            $pos^* \leftarrow prev$
            $\Delta^* \leftarrow \Delta$
        end if
    end if
    $prev \leftarrow next$
    $next \leftarrow next + 1$
end while
Retorne pos^*

serção de um novo nó na rota de todos veículos em tempo computacional $O(N)$, onde N representa o número total de nós presentes em uma solução S. A partir dos Algoritmos 5 e 6, as heurísticas de inserção utilizam as informações de custo das rotas para determinar qual será a próxima requisição e em qual rota ela será inserida. A seguir serão apresentadas as duas principais heurísticas de inserção do ALNS utilizadas em problemas de roteamento de veículos com coleta e entrega.

2.4.3.1 Inserção gulosa

O operador de inserção gulosa é uma heurística de inserção simples. Ela utiliza o critério de inserir a requisição com menor acréscimo de custo na solução. Em Pisinger e Ropke (2007) essa heurística de inserção é apresentada da seguinte forma.

Defina $\Delta_{i,v}$ como o acréscimo no custo da solução S se a requisição i for inserida no veículo v, defina U como o conjunto de y requisições a serem inseridas na solução S e defina $c_i = \min_{v \in V}\{\Delta_{i,v}\}$ como o custo em inserir a requisição i no melhor veículo v da solução S. Com base nisso, a heurística de inserção gulosa seleciona como próxima requisição para inserção aquela com menor custo definido por $\min_{i \in U} c_i$.

Esse operador itera até que todas as requisições sejam inseridas, uma a uma. Entretanto, cada vez que uma requisição é adicionada a solução, um veículo é modificado e, portanto, as requisições restantes precisam atualizar o custo de inserção relacionado a esse veículo. A atualização somente dos veículos modificados reduz o tempo de execução desse operador (PISINGER; ROPKE, 2007).

2.4.3.2 Inserção com critério de arrependimento

Esse operador visa complementar um problema relacionado ao operador guloso. O fato do operador guloso sempre selecionar a melhor inserção do momento faz com que ele ignore o impacto futuro daquela inserção nas demais requisições. O operador de inserção com critério de arrependimento resolve esse problema. Basicamente, ele considera a informação futura para cada requisição, onde ele analisa o custo da melhor inserção e considera o quão pior as demais opções de inserção serão se a melhor não for feita (PISINGER; ROPKE, 2007).

O operador com critério de arrependimento é formalmente definido da seguinte forma (PSARAFTIS, 1995). Considere $x_{iv} \in \{1, 2, ..., |V|\}$ como o v-ésimo melhor veículo para qual a requisição i pode ser inserida, dado que, $\Delta_{i,x_{iv}} \leq \Delta_{i,x_{iv'}}, \forall v \leq v'$. Além disso, considere o *valor de arrependimento* c_i^* como $c_i^* = \Delta_{i,x_{i2}} - \Delta_{i,x_{i1}}$, nessa demonstração o valor de arrependimento é representado pela diferença de custo entre a melhor e a segunda melhor inserção. Com base nessas definições o operador funciona da seguinte forma, em cada iteração é selecionada para inserção a requisição com maior valor de arrependimento, dado por $\max_{i \in U} c_i^*$ onde U representa o conjunto de requisições a serem inseridas. O operador de arrependimento foca em selecionar a requisição que se não for selecionada agora, pode gerar um maior arrependimento no futuro.

Até o momento esse operador foi definido em termos da diferença de custo dentre a primeira e a segunda inserção de uma requisição. Porém, ele pode ser estendido para suportar um intervalo maior de comparações. Nesse caso, regret-k é o operador onde k requisições são consideradas no cálculo do valor de arrependimento. A Equação 2.19 apresenta como o *valor de arrependimento* (c_i^*) pode ser calculado. Se no conjunto de requisições U existirem requisições que não possuem k rotas para calcular c_i^*, então a requisição com menor número de k será escolhida. Se ainda assim existirem casos onde mais de uma requisição possuir o mesmo número de veículos,

porém menor que k, então a opção com menor custo será selecionada (PISINGER; ROPKE, 2007).

$$\max_{i \in U} \{ \sum_{j=1}^{k} (\Delta_{i,x_{ij}} - \Delta_{i,x_{i1}}) \} \tag{2.19}$$

Similar ao operador guloso, a requisição escolhida para inserção irá modificar o veículo. Dessa forma, as demais requisições calculadas sobre esse veículo devem ser recalculadas para atualizar os custos associados. Fazer o cálculo somente do veículo modificado acelera o processo de otimização.

2.4.4 Geração de ruído

Os operadores utilizados pelo ALNS geralmente possuem algum fator aleatório. Esse fator é necessário porque ele possibilita uma maior exploração do espaço de busca, fazendo com que o algoritmo consiga fugir de ótimos locais.

Nos operadores de remoção a geração de ruído geralmente é feita randomizando a seleção das requisições. No exemplo do Algoritmo 3, o vetor $L = [r_0, r_1, ..., r_n]$ contém as requisições para remoção ordenadas conforme $gain(r_i) > gain(r_j), i < j, \forall r \in L$, ou seja, $L[0]$ representa a requisição com maior ganho associado. Para reduzir a chance de ficar preso em um mínimo local por sempre selecionar $L[0]$, embora seja a melhor opção, utiliza-se um fator de aleatorização na seleção. Nesse caso, seleciona-se a requisição $L[y^p|L|]$ onde y é um número aleatório gerado entre 0 e 1 e p é um parâmetro que controla a probabilidade de favorecimento das melhores requisições do vetor L. Dessa forma, quanto mais próximo de 1 o valor de p, mais uniforme será a chance de escolha de qualquer requisição em L, enquanto que quanto menor for o valor de p, maior será a chance de seleção dos primeiros elementos de L (PISINGER; ROPKE, 2007; LUTZ, 2014).

Nos operadores de inserção, a randomização é gerada da seguinte forma. Dado uma requisição r e um veículo v, depois de calculado o custo, $C(r, v)$, de inserir r em v por meio do Algoritmo 5, adiciona-se um fator aleatório $noise$ ao custo $C'(r, v) = max\{0, C(r, v) + noise\}$. O valor de $noise$ é definido conforme Equação 2.20. O ruído é aplicado pelo seguinte critério, a cada iteração a decisão de aplicar ou não o ruído é tomada usando o mesmo sistema de pesos/pontuações dos operadores de inserção/remoção (PISINGER; ROPKE, 2007; NACCACHE; CÔTÉ; COELHO, 2018).

$$noise = (rand() - 0.5) \times (\max_{i,j \in P} t_{ij}) \times 2 \tag{2.20}$$

2.4.5 Função de aceitação

Ambos LNS e ALNS possuem uma função de aceitação. Essa função define a probabilidade de uma determinada solução, de pior custo em relação a solução atual, ser aceita. Existem diversas formas de calcular essa probabilidade (LUTZ, 2014). Entretanto, o ALNS geralmente utiliza a função *Simulated Annealing*, ou SA (PISINGER; ROPKE, 2007).

Esse método tem suas origens nos conceitos da mecânica estatística (KIRKPATRICK; GELATT; VECCHI, 1983). Basicamente, o SA é baseado no processo físico de controle dos graus de liberdade das partículas atômicas de um dado material em função da temperatura. Nesse processo, quando o material é exposto a altas temperaturas, as partículas ganham um maior grau de liberdade, o que torna o material líquido e flexível. Após isso, durante um resfriamento controlado, o grau de liberdade das partículas vai reduzindo até retornar o material a um estado sólido, considerado ótimo (sem defeitos).

O processo de SA é usado para controlar a exploração do espaço de busca. Inicialmente, a temperatura é calculada e armazenada em uma variável τ, que irá decrescer conforme a taxa de resfriamento γ. Durante esse processo, novas soluções de pior custo serão aceitas se elas passarem pela função probabilística de aceitação apresentada na Equação 2.21, que considera a temperatura atual τ e o custo da solução $cost(S')$ (KIRKPATRICK; GELATT; VECCHI, 1983).

$$sa(S, S') = e^{\frac{cost(S) - cost(S')}{\tau}} \tag{2.21}$$

No caso da função SA, é necessário definir uma temperatura inicial. A forma mais básica usada pelo ALNS é apresentada na Equação 2.22 (PISINGER; ROPKE, 2007). O método apresentado define a temperatura com base no custo de uma solução, onde $cost(S)$ é o custo da solução base e ϵ é uma tolerância para ajuste da temperatura inicial. Essa equação é derivada da função de recozimento do SA e basicamente ela indica que a chance de aceitar uma solução pior que a solução inicial é de 50%.

$$\tau_{init}(S) = \frac{(cost(S) \times \epsilon)}{\log 2} \tag{2.22}$$

Durante o resfriamento de τ, a probabilidade de aceitar soluções piores decresce, ou seja, o algoritmo inicia a busca com maior intensidade de exploração, reduzindo-a com o passar do tempo. Entretanto, quando o algoritmo chega a um ponto de estagnação com $\tau < \tau_{min}$, o processo pode ser reinicializado aplicando a Equação

2.22 (NACCACHE; CÔTÉ; COELHO, 2018). O resfriamento é aplicado pelo fator de resfriamento γ, conforme Equação 2.23 (PISINGER; ROPKE, 2007).

$$\tau = \tau \times \gamma \tag{2.23}$$

2.5 HEURÍSTICAS DE BUSCA LOCAL

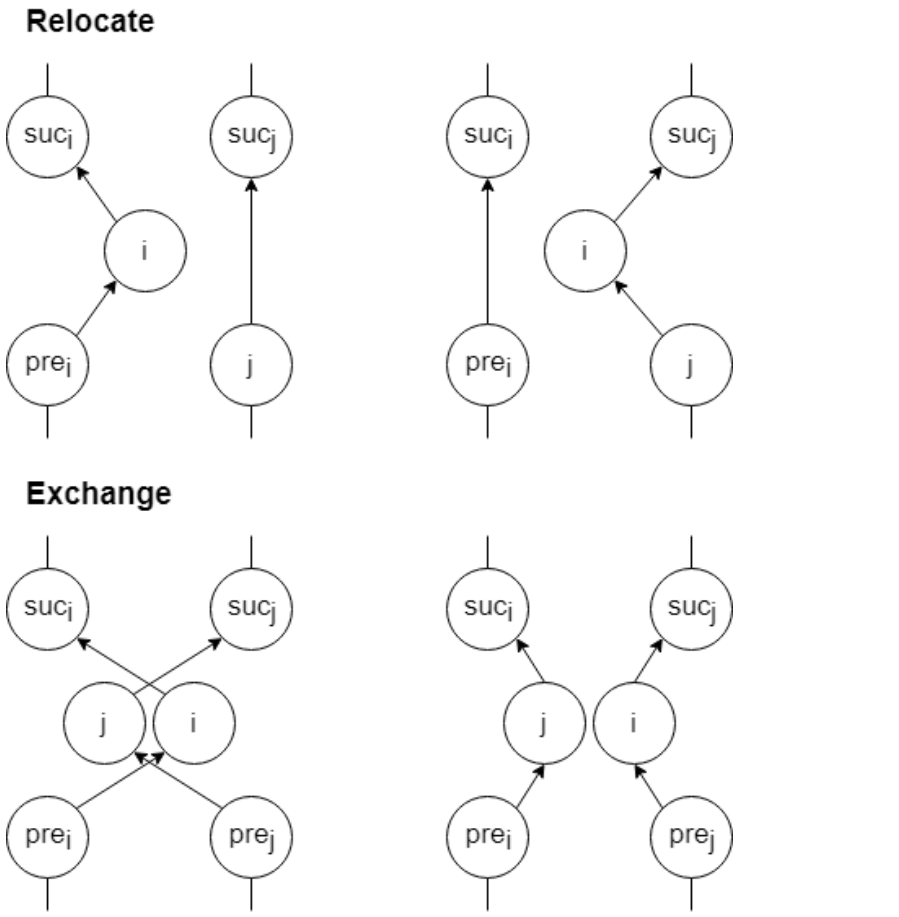
As heurísticas de buscas locais são usadas para intensificar a busca em torno de uma solução. Elas basicamente executam até não encontrar mais melhorias na solução. De forma geral, a busca local executa de forma determinística, incorporando à solução todas as melhorias encontradas. No caso do VRPTW, Savelsbergh (1992) propõem-se as seguintes buscas:

- Relocate: $(pre_i, i), (i, suc_i), (j, suc_j) \rightarrow (pre_i, suc_i), (j, i), (i, suc_j)$. Esse operador busca remover um nó de uma rota r_1 para tentar inserir em outra r_2, conforme Figura 7. Assim, essa heurística realoca cada requisição em cada rota, mantendo as soluções com menor custo. Ele executa até não encontrar mais melhorias.
- Exchange: $(pre_i, i), (i, suc_i), (pre_j, j), (j, suc_j) \rightarrow (pre_i, j), (j, suc_i), (pre_j, i), (i, suc_j)$. Esse operador busca selecionar um nó n_1 de uma rota r_1, e um nó n_2 de uma rota r_2, e trocar os nós de rota, inserindo n_1 em r_2 e n_2 em r_1, conforme Figura 7. Assim, essa heurística avalia cada par de requisição de rotas diferentes, mantendo as soluções de menor custo. Ele executa iterativamente para cada par até não encontrar mais melhorias.

2.6 ALNS APLICADO A SOLUÇÃO DO PDPTW

Um aspecto precisa ser levado em consideração na utilização do ALNS na solução do PDPTW. O ALNS não é adequado para lidar com dois objetivos simultaneamente, como é o caso da função objetivo definida na Equação 2.1a. Outras heurísticas geralmente tratam a minimização dos veículos como um coeficiente de custo associado ao número de veículos. Entretanto, essa estratégia não é adequada para o uso do ALNS. Portanto, quando a minimização do número de veículos é prioritária sobre outros objetivos, Pisinger e Ropke (2007) propõem o uso do ALNS em dois estágios separados. No primeiro estágio uma versão do ALNS é aplicada para minimizar o número de veículos ($ALNS_{NV}$), enquanto no segundo estágio, outro ALNS trata da minimização do custo total da solução ($ALNS_{TC}$). Um fator importante é que a proposta de Pisinger e Ropke (2007) só considera frotas homogêneas.

Figura 7 – Operações executadas pelos operadores de busca local *relocate* e *exchange*. O operador *relocate* busca trocar uma requisição de rota, enquanto operador *exchange* busca trocar duas requisições de duas rotas diferentes.



Fonte: Imagem adaptada pelo autor de (SAVELSBERGH, 1992)

Segundo Pisinger e Ropke (2007), a execução da abordagem de dois estágio ocorre da seguinte forma. No primeiro momento, uma heurística de construção é aplicada para gerar a solução inicial. Então, o primeiro estágio se inicia. Nessa etapa, o $ALNS_{NV}$ remove um veículo arbitrário da solução e as requisições desse veículo vão para um banco de requisições. Nesse momento, o $ALNS_{NV}$ tentará otimizar a solução alocando as requisições nos veículos restantes. Quando todas as requisições forem atendidas, um novo veículo é removido da solução e o ciclo reinicia. Para controlar o tempo de execução desse estágio, o critério de parada analisa o número de iterações que um determinado grupo de requisições não é alocado. No término da execução, a solução factível com menor número de veículos serve de entrada para o segundo estágio, no caso o $ALNS_{TC}$.

2.7 REVISÃO DE TRABALHOS RELACIONADOS

Nesta seção são apresentados os trabalhos relacionados ao tema de pesquisa dessa dissertação. Cada trabalho apresentado aqui possuí algum aspecto relacionado

ao problema de coleta e entrega com janela de tempo e requisições dinâmicas. As buscas de trabalhos foram feitas consultando as bases da *IEEExplorer*, *Scopus* e *Web of Science* usando os termos *dynamic pickup and delivery problem with time windows* no mês de novembro de 2018. Os filtros foram aplicados considerando a análise do tema pelo título e *abstract*. No final, foram avaliados aproximadamente 40 trabalhos relacionados, onde os mais relevantes são apresentados a seguir.

O trabalho de Dumas, Desrosiers e Soumis (1991) apresenta a primeira formulação do problema de coleta e entrega com janela de tempo. Os autores apresentam o trabalho como uma derivação do problema de roteamento de veículos com janela de tempo, considerando que cada cliente possuí associada a ele uma demanda, um ponto de coleta e um de entrega. Além disso, os autores propõem um algoritmo exato, baseado em um esquema de geração de colunas com a restrição de caminho de menor custo como sub-problema.

O trabalho de Pankratz (2005) considera a aplicação de um algoritmo genético para o problema do DPDPTW. Nesse estudo, é utilizada uma variante do algoritmo genético conhecida por algoritmo genético orientado à agrupamento, ou GGA (*Grouping Genetic Algorithm*), essa variante tem como princípio a ideia de agrupar requisições em cromossomos por veículo. O método proposto é composto de dois processos, um para gerenciar a parte dinâmica e outro para otimizar a solução. Para condução dos experimentos, os autores propõem um método de adaptação das instâncias de Li e Lim (2001) para instâncias dinâmicas que considera dois tipos de dinamicidade, i) dinamicidade do grau de urgência e do ii) grau de requisições dinâmicas. Os autores analisam os resultados para entender o grau de dinamicidade influencia nos resultados e como ele se relaciona com outras características de janela de tempo e localização geográfica dos pontos de coleta. Por fim, o estudo ainda compara os resultados do GGA contra duas heurísticas de otimização, indicando que o GGA é o melhor entre os avaliados. Uma das limitações é que esse trabalho aplica a otimização somente no objetivo do custo total.

O trabalho de Pisinger e Ropke (2007) propõe o algoritmo de busca larga adaptativa (ALNS) para o problema de coleta e entrega com janela de tempo, ou PDPTW. Nesse trabalho são considerados os operadores de remoção relacional, aleatória e de pior custo, e são considerados os operadores de inserção gulosa e com critério de arrependimento. O critério de aceite do ALNS é modelado sobre a função do *simulated annealing* (SA). Na avaliação dos resultados, parte da experimentação utiliza instâncias de benchmark de Li e Lim (2001) e outra parte utiliza instâncias geradas pelos autores. Nas instâncias geradas foram consideradas as restrições de múltiplos depósitos e frota heterogênea. A avaliação buscou comparar o ALNS com o LNS, no caso, o ALNS apresentou-se mais eficiente e novas soluções melhores foram

encontradas para as instâncias do *benchmark* de Li e Lim (2001).

O trabalho de Pureza e Laporte (2008) apresenta um estudo acerca de estratégias de espera e *buffering* de requisições dinâmicas aplicadas ao DPDPTW. O trabalho avalia como a política de atendimento das requisições deve ser feita para maximizar os ganhos durante a otimização. A estratégia de espera é baseada em postergar a viagem do veículo para o próximo cliente de forma a aumentar o tempo de otimização do algoritmo. A estratégia de *buffer* busca acumular requisições dinâmicas não urgentes de forma que o algoritmo consiga priorizar o atendimento de requisições urgentes que venham a aparecer. Os experimentos são realizados em instâncias de testes geradas aleatoriamente usando uma heurística de construção e desconstrução baseada na proposta de Solomon (1987). Testes foram executados com instâncias de até 400 nós de (Li; Lim, 2001), convertidas em dinâmicas variando o horário de anunciação e o tempo de viagem. Os resultados indicam que estratégias de espera sem *buffer* são melhores para casos de dinamismo médio, enquanto com *buffer* são melhores para dinamismo alto.

O trabalho de Lutz (2014) apresenta uma visão detalhada sobre o algoritmo ALNS e o problema do RPDPTW, (*Rich Pickup and Delivery Problem with Time Windows*). Nesse estudo, o autor aborda de forma detalhada os conceitos de busca local e as diferenças entre o LNS e o ALNS. Além disso, ele descreve o funcionamento dos diversos tipos de operadores de remoção e inserção. Adicionalmente, o autor apresenta um estudo acerca dos parâmetros do algoritmo e, por fim, conduz uma análise experimental. Os experimentos usam instâncias de *benchmark* propostas por Pisinger e Ropke (2007). É feita uma análise do impacto de cada operador e uma comparação com os resultados da literatura. Os resultados permitiram uma análise detalhada do impacto causado por cada operador e também foram encontradas novas melhores soluções para as instâncias de *benchmark* de Pisinger e Ropke (2007).

O trabalho de Dridi, Alaia e Borne (2015) propõe um algoritmo genético NSGA-II aplicado ao problema do DPDPTW com duplo objetivo de minimização: custo da solução total e atraso total no atendimento das requisições. No primeiro momento, os autores apresentam um operador de recombinação e um operador de mutação. No segundo momento os autores apresentam as alterações do algoritmo para manter as invariantes de precedência e paridade do problema. Por último, os autores apresentam um método baseado em valores mínimos para lidar com a seleção bi-objetiva. A avaliação dos experimentos é feita utilizando instâncias da literatura propostas por Li e Lim (2001) sob condições estáticas e dinâmicas. Entretanto, pelo fato da formulação dos objetivos ser diferente da proposto por Li e Lim (2001), não é feita nenhuma comparação com os resultados da literatura.

O trabalho de Necula, Breaban e Raschip (2017) apresenta uma avaliação

do algoritmo colônia de formigas (ACO - *Ant Colony Optimization*) sobre instâncias do VRPTW com requisições dinâmicas, ou DVRPTW. No método proposto dois processos paralelos são executados, em um deles o ACO é responsável por otimizar as requisições e no outro processo um algoritmo é responsável por gerenciar a anunciação de requisições dinâmicas. Além disso, os autores propõem o modelo de fatias de tempo para gerenciar os ciclos de otimização e anunciação de novas requisições. Os experimentos são conduzidos sobre as instâncias de *benchmark* propostas por Solomon (1987), porém adaptadas para a versão dinâmica. A comparação é feita entre o ACO proposto e outra variante da literatura. Por fim, o ACO proposto apresenta melhores resultados na maior parte das instâncias.

O trabalho de Tchoupo et al. (2017) propõem uma meta-heurística para o problema do PDPTW. Os autores utilizam uma abordagem ACO acoplada com algoritmos de busca local. Nesse trabalho, os autores identificam a dificuldade do uso de heurísticas construtivas (ACO) para problemas com restrições de paridade e precedência (PDPTW). A dificuldade reside em inserir uma coleta na etapa construtiva, e garantir que será possível adicionar a entrega no futuro de forma factível em um tempo de computação factível. Para superar essa dificuldade, os autores propõem uma verificação para testar a factibilidade da rota durante a etapa construtiva. A experimentação é feita por meio da aplicação do algoritmo em instâncias com 100 vértices do conjunto de *benchmark* de Li e Lim (2001). Os testes apresentam bons resultados e inclusive novas soluções ótimas.

O trabalho de Naccache, Côté e Coelho (2018) apresenta um algoritmo ALNS aplicado ao problema de múltiplas coletas e uma entrega com janela de tempo, ou MPDPTW (*Multi Pickup and Delivery Problem with Time Windows*. O MPDPTW é uma variante do PDPTW que permite múltiplas coletas por requisição. Os autores propõem um conjunto de instâncias de *bencharmk* com variados tamanhos das janelas de tempo e números de coletas por requisição. Antes da execução dos experimentos, os autores executam um *tunning* dos parâmetros do algoritmo. O resultado final compreende um novo *benchmark* com melhores soluções da literatura. Para comparar a qualidade dos resultados, um algoritmo exato também é aplicado para resolver as instâncias mais simples, em que os mesmos resultados foram encontrados pelo ALNS.

O trabalho de Chen et al. (2018) apresenta um estudo sobre o ALNS aplicado ao problema de roteamento de veículos dinâmicos. Nesse estudo eles exploram a capacidade do algoritmo em responder rapidamente aos eventos dinâmicos do problema. O cenário de estudo avaliado é o DVRPTW com frota heterogênea e janelas de tempo restritas. O gerenciamento da dinamicidade é feito quebrando o problema em sub-problemas conforme as fatias de tempo determinadas. São propostas adaptações

dos operadores de inserção/remoção e ainda aplica-se o uso de uma busca local 2-opt. Diferente do método padrão de pontuação individual por operador de inserção e remoção, os autores propuseram a pontuação para o par de operadores aplicados na iteração. Os experimentos são conduzidos sob instâncias do VRPTW dinâmicas e os resultados apresentam reduções de 4.93% no número de veículos e 4.65% no custo total.

O trabalho de Pereira (2019) apresenta um estudo acerca do problema de roteamento de veículos dinâmicos com as restrições de coleta e entrega simultâneas, janela de tempo e frota heterogênea (HDVRPSPDTW). O método proposto baseia-se na aplicação da heurística T-ONN como heurística de inserção, na heurística MMAS para otimização das rotas e em uma estratégia de busca local combinada das heurísticas *Relocate*, *Exchange* e *Cross* para refinamento das soluções encontradas. Os resultados apresentaram melhores resultados para instâncias clusterizadas em relação a literatura atual. As contribuições são uma nova modelagem do problema e o desenvolvimento de um novo método robusto para esse problema.

2.7.1 Considerações sobre os Trabalhos Relacionados

De forma geral, os trabalhos citados demonstram que diversas abordagens já foram propostas como solução, contemplando variadas características do VRP tais quais janela de tempo, coleta e entrega, capacidade de carga, tempo de serviço, demanda, frota heterogênea e dinamismo. Além disso, a literatura acerca da dinamicidade também apresenta material relevante a cerca de métricas e metodologias de controle aplicadas a caracterização desse tipo de problema. Para sintetizar a comparação em relação a literatura, a Tabela 2 apresenta uma comparação entre os trabalhos mais relevantes analisados nesta seção e o trabalho em tela (destacado na tabela em negrito).

Em relação aos tipos de problema apresentados na Tabela 2, coluna “Problema”, a literatura apresenta mais trabalhos relacionados a versão estáticos do PDPTW (considerando variações como RPDPTW e MPDPTW), enquanto que poucos trabalhos estudam a versão dinâmica (DPDPTW). Além disso, durante a pesquisa bibliográfica não foram encontrado métodos exatos aplicados a solução de DPDPTW, a falta de trabalhos que utilizam esse tipo de solução se justifica pelo elevado tempo de processamento desses métodos. Assim, abordagens heurísticas e meta-heurísticas, tais como: ALNS, ACO, GA e construção-desconstrução, são mais comumente aplicadas neste tipo de problema.

Ainda, analisando a Tabela 2, é possível perceber que somente trabalhos aplicados a versão estática consideram instâncias com até 1000 nós, a grande maioria dos trabalhos aplicados a problemas dinâmicos analisam instâncias com 100 nós e

Tabela 2 – Variáveis da formulação do PDPTW

Trabalho	**Problema**	**Método**	**Função objetivo**	**Tamanho máx. instâncias**
Dumas, Desrosiers e Soumis (1991)	PDPTW	*Column-generation*	TC	-
Pankratz (2005)	DPDPTW	GA	TC	100 nós
Pisinger e Ropke (2007)	PDPTW	ALNS	NV + TC	1000 nós
Pureza e Laporte (2008)	DPDPTW	const.-deconst.	NV + TC	400 nós
Dridi, Alaia e Borne (2015)	DPDPTW	GA	TC + WT	100 nós
Lutz (2014)	RPDPTW	ALNS	TC	1000 nós
Necula, Breaban e Raschip (2017)	DVRPTW	ACO	NV + TC	100 nós
Tchoupo et al. (2017)	PDPTW	ACO	NV + TC	100 nós
Naccache, Côté e Coelho (2018)	MPDPTW	ALNS	TC	400 nós
Chen et al. (2018)	DVRPTW	ALNS	NV + TC	100 nós
Pereira (2019)	HDVRPSPDTW	ACO	NV + TC	100 nós
Trabalho em tela	**DPDPTW**	**ALNS**	**NV + TC**	**1000 nós**

um caso avaliou instâncias com até 400 nós (PUREZA; LAPORTE, 2008). Instâncias de 100 nós podem ser consideradas fáceis e rápidas de serem resolvidas. Ainda em relação as instâncias dinâmicas, não foi encontrado nenhum trabalho que aplicou a otimização de problemas dinâmicos com até 1000 nós.

Outro aspectos importante é que não foi encontrado nenhuma base de *benchmark* do DPDPTW bi-objetivo (considerando otimização do número de veículos e do custo total) com as instâncias de teste e resultados finais para comparação de novos métodos. Os trabalhos mais próximos são de Pankratz (2005) e Pureza e Laporte (2008) que apresentam uma metodologia parar converter a base das instâncias estáticas de Li e Lim (2001) para instâncias dinâmicas. Além disso, os resultados obtidos pelos autores apresentam uma comparação em porcentagem de desvio das soluções, seria importante apresentar os valores reais obtidos para comparar o desempenho de novos métodos de forma mais precisa.

Foi observado também que a aplicação do ALNS para problemas de coleta e entrega é bem difundida na literatura, conforme pode ser observado na Tabela 2 o ALNS foi aplicada com sucesso a instâncias do PDPTW com até 1000 nós. Um dos seus fatores de sucesso do ALNS é o fato dessa ser uma heurística de melhoramento, isso implica que ela pode computar o custo da inserção de uma requisição de coleta e entrega de forma mais eficiente. Na Tabela 2 também são apresentados trabalhos com

algoritmos construtivos, como ACO, porém estes são menos eficientes porque que durante a construção sequencial de uma rota, é necessário computar uma garantia de que se uma coleta for inserida a entrega também poderá ser adicionada nas fases seguintes da construção.

Para o problema do DPDPTW, foi observado a falta de uma análise aprofundada da dinamicidade em relação as soluções finais e ao comportamento do algoritmo. Em (PANKRATZ, 2005) é feito uma análise de como o problema tende a se comportar sob variados graus de dinamicidade e características do problema (como horizonte de tempo e tipo de distribuição geográfica dos nós). Entretanto, os autores não é analisam internamente o comportamento do algoritmo para tentar encontrar gargalos. Outro problema é que os autores avaliam os resultados somente em função da otimização do custo total, enquanto a proposta original do DPDPTW prevê a otimização bi-objetivo do número de veículo e do custo total. No trabalho de (PUREZA; LAPORTE, 2008) os resultados são analisados com variados graus de dinamicidade, porém não é analisado como a otimização se comporta em função das diferentes características do problema.

Com base nos trabalhos revisados, percebe-se que o ALNS é uma das heurísticas mais aplicadas ao problema PDPTW, dado a sua capacidade de encontrar soluções otimizadas e seu bom desempenho computacional. Além disso, o estudo do DPDPTW apresenta-se como relevante devido às seguintes lacunas de pequisa identificadas: 1) a necessidade de algoritmos mais eficientes para lidar com instâncias maiores do problema (maior número de nós), 2) aos poucos trabalhos que atacam a otimização do DPDPTW e a 3) falta de uma analise aprofundada de como a dinamicidade impacta na otimização das soluções analisando os resultados sob diferentes perspectivas, como:

- Variação no grau de urgência das requisições dinâmicas;
- Variação do grau do número de requisições dinâmicas;
- Variação do tamanho do horizonte de tempo do dia de trabalho;
- Variação dos tipos de distribuições geográficas dos nós;
- Impacto na otimização do número de veículos;
- Impacto na otimização do custo total;
- Impacto na porcentagem de requisições atendidas;
- Impacto no comportamento interno do algoritmo;

3 SOLUÇÃO PROPOSTA

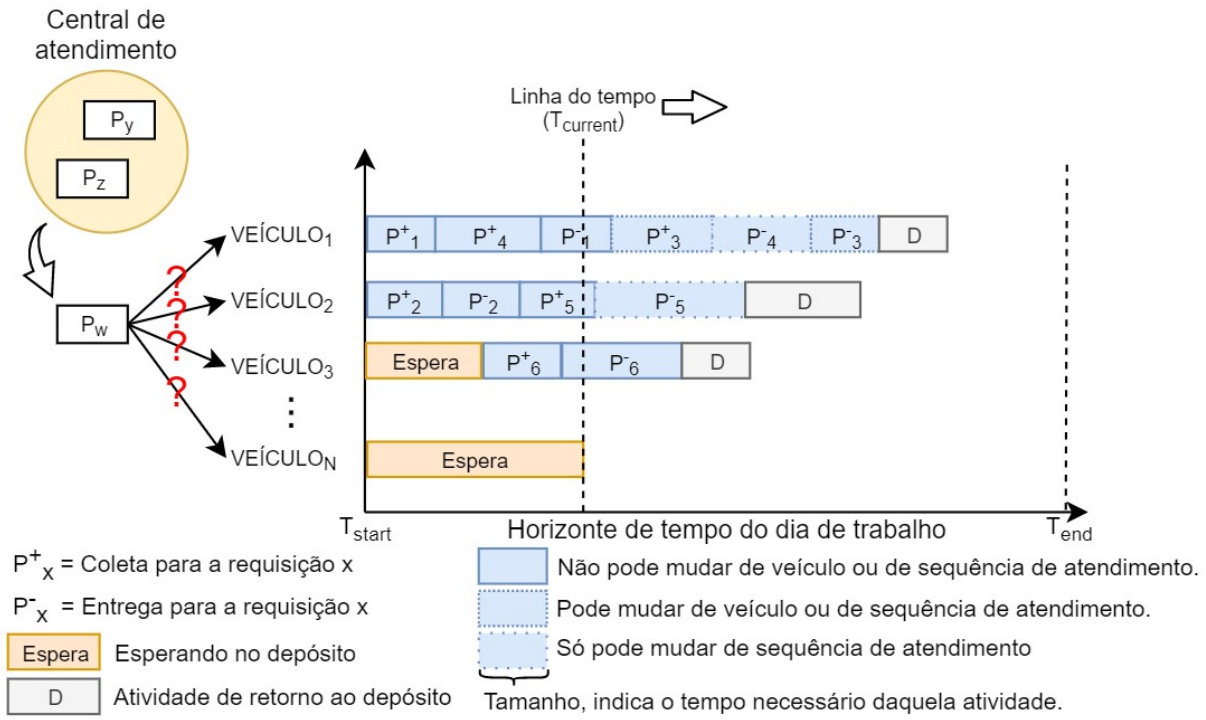
O objetivo desse trabalho é propor um método de otimização para o problema de roteamento de veículos com coleta e entrega, janela de tempo e requisições dinâmicas, ou DPDPTW. Para atacar esse problema, é proposto o desenvolvimento de um *solver* baseado no uso do algoritmo ALNS. Antes de entrar nos detalhes da solução, inicialmente será dada ênfase aos aspectos que caracterizam o problema DPDPTW, e que devem ser considerados para o desenvolvimento da solução. A Figura 8 e a listagem a seguir apresentam um resumo dos principais aspectos:

- Existe uma central de atendimento que recebe e insere as requisições dinâmicas no problema;
- Todas as requisições possuem um nó de coleta e entrega, denotado por P_i^+ e P_i^- para $i \in R$, respectivamente;
- O horizonte de tempo do dia de trabalho é dado pela janela de tempo do depósito;
- A variável $T_{current}$ é atualizada com o tempo corrente durante o dia de trabalho e é usada para definir o estado de atendimento das requisições (Seção 3.1);
- Veículos aguardam no depósito (em espera) até possuírem uma rota para iniciar o atendimento;
- Somente nós de clientes que ainda não foram atendidos podem ser re-arranjados na solução;
- Demais restrições do PDPTW (Seção 3.2.3).

O DPDPTW compreende o atendimento de requisições estáticas e dinâmicas, ambos os tipos geram implicações diferentes na modelagem da solução (Seção 2.3), conforme apresentado a seguir:

- *Requisições estáticas*: representam requisições de clientes anunciadas fora do horizonte de tempo do dia de trabalho, por exemplo: compras feitas durante a noite em um *e-commerce* que precisam ser agendas para entrega no dia seguinte. Para essas requisições, o tempo de anunciação A_i é definido como zero.
- *Requisições dinâmicas*: aparecem durante a operação dos veículos para serem atendidas no mesmo dia, por exemplo: compras de clientes para entrega no

Figura 8 – Visão geral do problema DPDPTW. Apresenta as limitações dos arranjos possíveis nas requisições impostas com base nas visitas dos veículos. Apresenta também a central de atendimento que recebe requisições dinâmicas.



Fonte: Imagem gerada pelo autor.

mesmo dia. Para essas requisições, o tempo de anunciação A_i representa o horário da ligação do cliente.

No mundo real, as empresas possuem um tempo de operação pré-determinado, chamado de horizonte de tempo. Na Figura 8, o tempo T_{start} e T_{end} definem o início e término do horizonte de tempo do dia de trabalho. No DPDPTW, uma solução inicial composta pelas requisições estáticas deve ser gerada antes do tempo T_{start} para que os veículos possam iniciar a operação. Em seguida, durante o decorrer do dia, o *solver* deve continuar a melhorar a solução buscando a otimização do custo a medida que a central de atendimento vai recebendo novas requisições dinâmicas. No final do dia de trabalho (tempo T_{end}), o *solver* deve prever o retorno de todos os veículos ao depósito com todos os clientes atendidos.

Quando uma requisição dinâmica é anunciada via ligação do cliente, o otimizador executa as seguintes ações: 1) tenta alocar a requisição em um dos veículos já em operação, caso não consiga, 2) tenta alocar um veículo ocioso no depósito ou, caso não haja, 3) aloca um veículo extra (terceirizar para outra empresa). Do ponto de vista operacional, o objetivo é tentar reutilizar ao máximo os veículos em operação antes de alocar veículos ociosos ou extras.

Uma requisição dinâmica requer a existência de um tempo de alocação de

veículo (*setup time*). Esse tempo provê uma certa folga para que a empresa possa determinar como a requisição será atendida, seja por um novo veículo ou por um veículo em operação.

No DPDPTW, todas as requisições possuem um ponto de coleta e um de entrega, isso implica em duas restrições: 1) a coleta deve sempre preceder a entrega e 2) o veículo que executa a coleta deve ser o mesmo a fazer a entrega, ou seja, quando uma requisição é coletada por um veículo ela se torna exclusiva daquele veículo. O tempo corrente $T_{current}$ é usado no cálculo da realocação, quando o horário da coletado é menor que $T_{current}$, isso significa que a requisição se torna exclusiva do veículo, porque o veículo que coleta a carga deve deve ser o mesmo a entrega-lá (ex: requisição 1 da Figura 8). Entretanto, a realocação é possível enquanto a coleta ainda não foi realizada, isso significa que a requisição pode trocar de veículo se todas as restrições do problema (tempo, demanda, etc.) continuarem sendo atendidas. Na Figura 8 a requisição 3 é um exemplo de requisição que pode ser alocada em outro veículo.

No problema DPDPTW, os veículos podem iniciar o dia de trabalho no momento mais adequado, como no caso do veículo 3 da Figura 8, que adia o máximo possível sua saída do depósito para atendimento do seu primeiro cliente, esperando a chegada da primeira requisição dinâmica para dar inicio a sua operação. A alocação de novos veículos só acontece para as requisições dinâmicas. Além disso, todos os veículos que partem do depósito devem retornar a ele no final do dia de trabalho, respeitando o tempo limite T_{end}. Esse retorno é calculado considerando o tempo de retorno dos veículos entre a última entrega e o depósito no planejamento da rota. Essa restrição é necessária para atender leis trabalhistas de jornada de trabalho, onde infringir essas leis pode resultar em multas para as empresas de transportes.

Para o trabalho em questão, o problema DPDPTW possuí dois objetivos priorizados de forma hierárquica. O primeiro é a minimização de veículos e o segundo é a minimização do custo total (tempo de viagem) (ver Seção 2.2). Isso significa que uma solução é considerada melhor que outra quando ela possuir menor número de veículos ou número igual de veículos com menor custo total. Nesse sentido, o método proposto permite que a solução possa diminuir o número de veículos mesmo que isso cause o aumento do custo total do roteamento. Essa priorização foi adotada conforme trabalhos de Li e Lim (2001), Pisinger e Ropke (2007).

3.1 RESTRIÇÕES TEMPORAIS E DINÂMICAS

As restrições temporais e dinâmicas são necessárias para controlar quais requisições podem ser re-arranjadas durante o roteamento (Figura 8) e se as requisições dinâmicas são válidas para serem aceitas pela central de atendimento. Conforme já

discutido na Seção 2.3.2, essas restrições devem ser tratadas pelo *solver* devido às características de veículo em movimento e anunciação de requisições dinâmicas. As restrições temporais e dinâmicas são aplicadas durante a execução do algoritmo.

No DPDPTW, o tempo de execução do algoritmo é determinado pelo horizonte de tempo do dia de trabalho ($T_{wd} = T_{end} - T_{start}$) definido com base na janela de tempo do depósito ($T_{start} = e_D$ e $T_{end} = l_D$). Em um cenário real, se $e_D = 8AM$ e $l_D = 5PM$ isso resulta em um dia de trabalho de $9H$ ($T_{wd} = 9H$), ou seja, o algoritmo inicia a execução as 8AM e executa durante 9 horas. Com base no valor de T_{wd}, as restrições temporais e dinâmicas são sempre aplicadas ao longo desse tempo. Para isso, a variável $T_{current}$ (momento atual na linha do tempo, $T_{start} < T_{current} < T_{end}$) serve como referência para o *solver* e o ALNS ao longo da execução. A variável $T_{current}$ é atualizada com o tempo atual a cada iteração do *solver*, essa informação ajuda o algoritmo a decidir quais arranjos de nós são possíveis durante o processo combinatório e validar se as requisições dinâmicas podem ser aceitas. Além disso, a variável $T_{current}$ é usada na decisão de término do algoritmo, conforme Definição 3.1.1.

Definição 3.1.1. *Decisão de término:* Se $T_{current} > T_{end}$, dado que $T_{current}$ representa o tempo corrente e T_{end} o término do dia de trabalho, então o dia de trabalho terminou e o algoritmo é finalizado.

3.1.1 Restrição temporal

O roteamento online limita as otimizações que podem ser feitas ao longo da evolução das visitas dos veículos. Na Seção 2.3.2 é introduzido o conceito de fixação dos nós visitados, ele serve para indicar quais nós das rotas já foram visitados e não podem mais ser otimizados. Entretanto, naquela seção não é apresentada uma definição explícita de como essas propriedades são definidas. Portanto, esta seção visa definir essas propriedades com base no método proposto.

As restrições temporais do problema determinam como os nós de coleta e entrega das requisições podem ser arranjados pelo *solver* em relação a seus tempos de atendimento. Basicamente, as restrições são validadas com base nos tempos de chegada e partida calculados para as visitas em relação a variável $T_{current}$. Isso permite determinar quais nós podem ou não ser permutados. Para facilitar o entendimento, a Figura 8 mostra alguns exemplos práticos limitados pelas restrições temporais durante a otimização online:

- Os nós $P_1^+, P_4^+, P_2^+, P_2^-$ e P_6^+ estão fixos na solução, ou seja, esses nós não podem ser arranjados de forma diferente. Isso acontece porque o horário da visita é menor em relação ao tempo corrente;

- Os nós P_1^-, P_5^+ e P_6^-, assim como seus antecessores, também estão fixos e não podem ser rearranjados na solução. Nesse caso, o tempo corrente indica que os veículos estão viajando (ou transitando) em direção a aquele nó;
- Os nós P_3^+ e P_3^- podem ser otimizados arranjando posições diferentes na rota do mesmo veículo ou de outro veículo. Porém, para que arranjos diferentes possam ser aplicadas, todas as restrições da Equação 3.3a devem ser respeitadas e os nós fixados não podem ser alterados;
- O nó P_5^- pode ser arranjado em posições diferentes na rota do mesmo veículo porém não em outros veículos. Esse nó não pode trocar de veículo porque sua coleta já foi feita pelo veículo atual. Como no item anterior, esse tipo de arranjo também deve respeitar as restrições impostas pela Equação 3.3a e não pode alterar o arranjo dos nós fixados;
- No final das rotas, todos veículos devem retornar ao depósito antes do término do dia de trabalho.

Para que o *solver* saiba quais nós podem ser arranjados em função de $T_{current}$, foi definido um conjunto de restrições compostas pelos seguintes estados: Em espera (*Idle*), Transição (*Transition*) ou Atendido (*Committed*). Formalmente, o conjunto de estados para cada nó do problema é definido usando as seguintes restrições:

Definição 3.1.2. *Estado de espera (Idle):* Dado um nó qualquer de coleta ou entrega $P_i \in P$ e $i \in N$, esse nó é definido como *Idle* se $T_{current} \leq T_{prev(i)}^{depart}$, onde $T_{current}$ representa o tempo corrente e $T_{prev(i)}^{depart}$ representa o tempo de partida do nó predecessor de P_i na rota do veículo de atendimento de P_i.

Definição 3.1.3. *Estado de transição (Transition):* Dado um nó qualquer de coleta ou entrega $P_i \in P$ e $i \in N$, esse nó é definido como *Transition* se $T_{current} > T_{prev(i)}^{depart} \wedge T_{current} \leq T_i$, onde $T_{current}$ representa o tempo corrente, T_i representa o tempo de atendimento do nó P_i e $T_{prev(i)}^{depart}$ representa o tempo de partida do nó predecessor de P_i na rota do veículo de atendimento de P_i.

Definição 3.1.4. *Estado de atendido (Committed):* Dado um nó qualquer de coleta ou entrega $P_i \in P$ e $i \in N$, esse nó é definido como *Committed* se $T_{current} > T_i$, onde $T_{current}$ representa o tempo corrente e T_i representa o tempo de atendimento do nó P_i.

Sem a definição desses estados o processo de otimização poderia falhar ao arranjar nós já atendidos para posições a frente da linha do tempo, fazendo com que o veículo visitasse o mesmo nó mais de uma vez, ou ainda arranjar um nó que ainda não foi atendido para trás da linha do tempo, fazendo com que este nó nunca fosse atendido.

Para que o controle dos estados seja preciso, a variável $T_{current}$ é comparada com as medidas de agendamento do tempo dos nós (ver Seção 2.4.1). Na solução proposta, essas medidas de agendamento do tempo são calculadas durante a avaliação da Equação 3.3a. Dessa forma, os estados podem facilmente ser calculados a cada iteração do *solver*.

3.1.2 Restrição de dinamicidade

O objetivo da restrição de dinamicidade é garantir que uma requisição anunciada dinamicamente pode ser atendida (ver Seção 2.3). A invalidação acontece quando não há tempo disponível para atender uma nova requisição em função do horário que a mesma foi anunciada. No método proposto a garantia é dada da seguinte forma: quando uma requisição dinâmica é anunciada, verifica-se a possibilidade de um novo veículo fazer o atendimento partindo do depósito no momento atual ($T_{current}$) considerando também o tempo de preparação (T_{setup}). Essa validação garante que caso o otimizador não consiga reaproveitar um dos veículos em operação, um novo veículo alocado exclusivamente para essa requisição consegue. Se o otimizador não for eficiente no reaproveitamento da frota, então muitos novos veículos podem ser alocados e isso pode levar a solução a realizar uma super alocação da frota. Inclusive, a super alocação pode extrapolar a capacidade de frota da empresa. No mundo real, esse cenário nas empresas transportadoras implicaria em terceirizar veículos para atender as novas requisições.

Formalmente, a restrição de aceitação de uma requisição dinâmica é feita da seguinte forma. Defina $R_{new} = \{P^+_{new}, P^-_{new}\}$ com uma requisição dinâmica, onde P^+_{new} representa o nó de coleta e P^-_{new} representa o nó de entrega, defina T^{depart}_x como o tempo de partida de um determinado nó x, defina T_{setup} como o tempo de setup do novo veículo e defina $T_{current}$ como o tempo corrente do dia de trabalho. Considerando que um novo veículo pode ser alocado no depósito, a nova requisição R_{new} é aceita se a Equação 3.1 for satisfeita.

$$\begin{aligned} valid(P_{new}) \Rightarrow\ & T_{current} + T_{setup} + t_{D,P^+_{new}} < l_{P^+_{new}} \\ & \wedge\, T^{depart}_{P^+_{new}} + t_{P^+_{new},P^-_{new}} < l_{P^-_{new}} \\ & \wedge\, T^{depart}_{P^-_{new}} + t_{P^-_{new},D} < l_{Depot} \end{aligned} \tag{3.1}$$

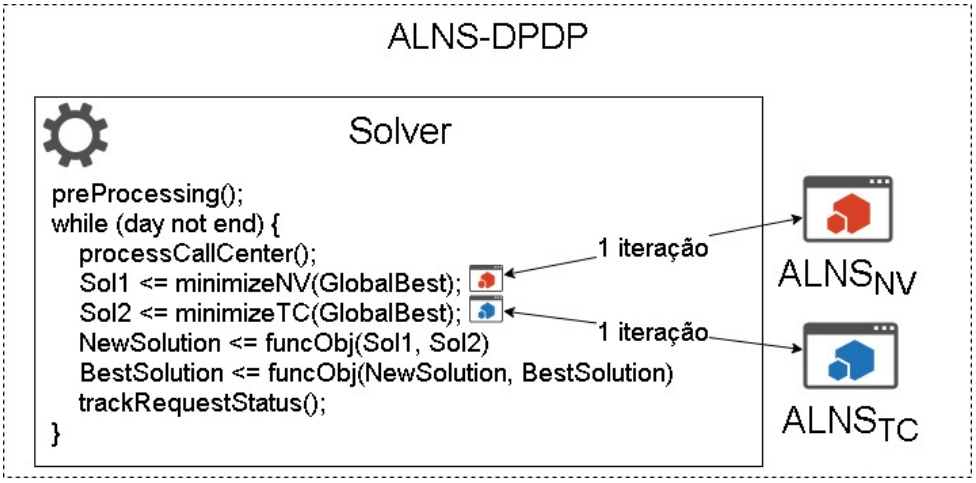
3.2 MÉTODO PROPOSTO

Esta seção visa apresentar o *solver* proposto aplicado ao DPDPTW. Como o problema possuí dois objetivos, o método proposto deve otimizar ambos respeitando

a hierarquia entre eles. O nome escolhido para o método foi **Adaptive Large Neighborhood Search for Dynamic Pickup and Delivery Problem** - **ALNS-DPDP**. Esse nome é derivado da meta-heurística utilizada como base para sua construção (Seção 2.4). A Figura 9 apresenta de forma simplificada a arquitetura proposta, que é composta de um *solver* e duas instâncias do ALNS. Cada instância do ALNS foca em um objetivo do problema enquanto o *solver* resolve o gerenciamento da parte dinâmica do problema, recebendo e inserindo na solução novas requisições a medida que elas chegam na central de atendimento. As responsabilidades do *solver* apresentadas na Figura 9 são executadas na seguinte ordem:

- Gerar uma solução inicial com requisições conhecidas *a priori* do início da operação do dia de trabalho;
- Acompanhar o tempo corrente ($T_{current}$) conforme o período do dia de trabalho;
- Receber requisições dinâmicas na central de atendimento e inseri-las na solução;
- Invocar uma iteração do $ALNS_{NV}$ para tentar reduzir o número de veículos;
- Invocar uma iteração do $ALNS_{TC}$ para tentar reduzir o custo total;
- Escolher a melhor solução respeitando a hierarquia dos objetivos;
- Atualizar o estado das requisições com base nas visitas realizadas até o momento ($T_{current}$).

Figura 9 – Visão simplificada da solução proposta do ALNS-DPDP, onde o *solver* é responsável por gerenciar a otimização online e a execução dos algoritmos ALNS. A cada iteração, um iteração do $ALNS_{NV}$ e do $ALNS_{TC}$ é executada.



Fonte: Imagem gerada pelo autor.

A estratégia de ter uma instância do ALNS para cada objetivo é baseada na proposta por Pisinger e Ropke (2007), apresentada na Seção 2.6. A diferença reside na forma como os ALNSs executam ao longo do processo de otimização. Pisinger e Ropke (2007) propõem que o $ALNS_{NV}$ finalize seu ciclo de iterações para a minimização do número de veículos antes que o $ALNS_{TC}$ inicie seu ciclo de iterações para a minimização do custo total, fazendo com que a otimização seja dividida em dois estágios. Essa abordagem é proposta para otimização de problemas estáticos.

Diferentemente da estratégia apresentada por Pisinger e Ropke (2007), o ALNS-DPDP proposto neste trabalho realiza a execução de uma iteração do $ALNS_{NV}$ e em seguida uma do $ALNS_{TC}$ dentro de cada iteração do *solver*. A execução sequencial dos dois ALNSs dentro da mesma iteração do *solver*, diferente de Pisinger e Ropke (2007), é necessária para dar respostas rápidas à otimização das requisições dinâmicas recebidas ao longo do dia de trabalho. Por exemplo, o $ALNS_{NV}$ precisar ser capaz de evitar a adição de um novo veículo quando uma requisição dinâmica é anunciada, e em contrapartida, o $ALNS_{TC}$ precisa ser capaz de encontrar rotas de menor custo para os veículos em operação. Com base nisso, as seções seguintes irão detalhar o funcionamento do *solver* e dos dois algoritmos ALNS.

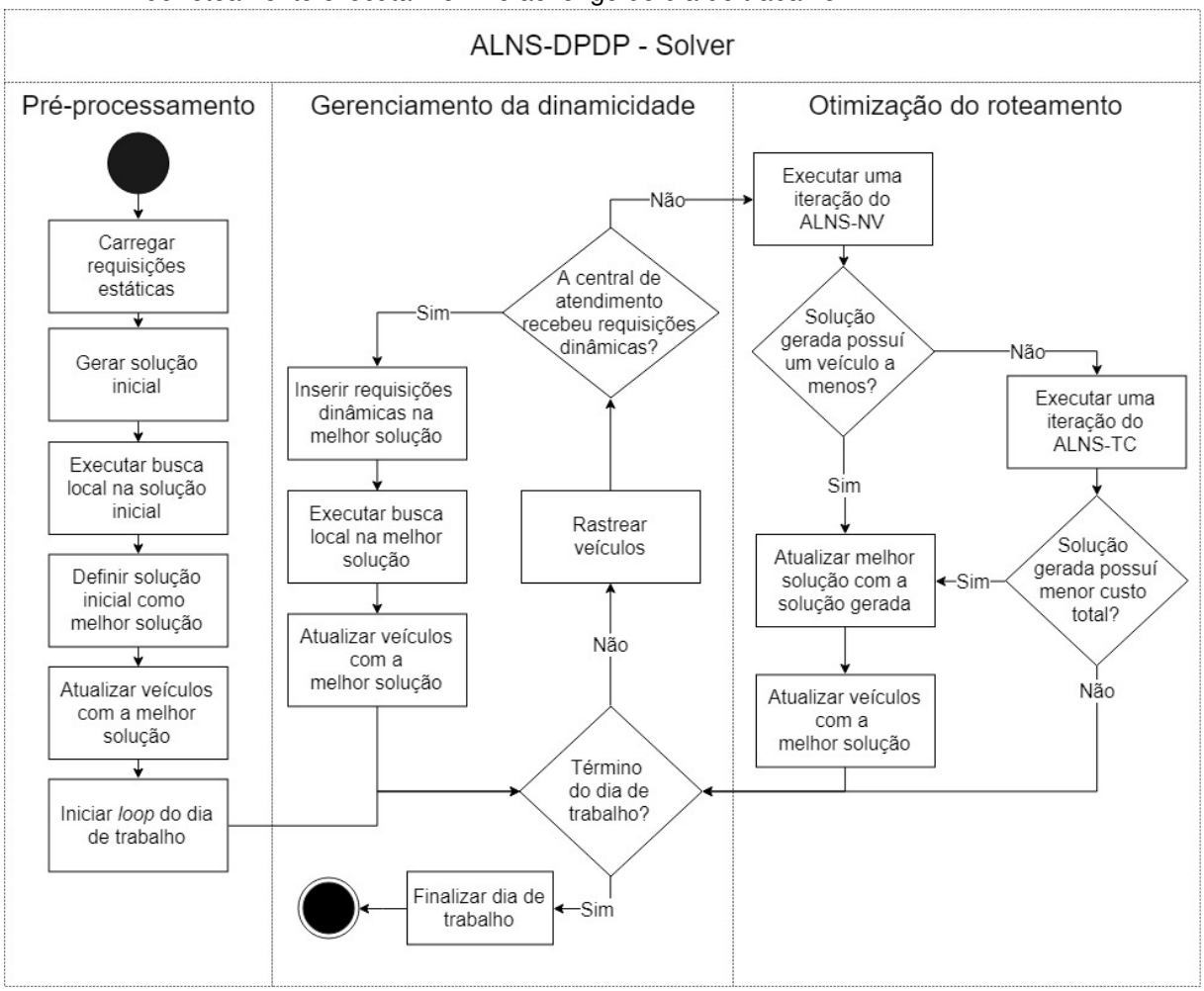
3.2.1 Método do *solver*

Esta seção detalha o funcionamento do *solver* para o ALNS-DPDP. A Figura 10 apresenta as tarefas e o fluxo de execução completo da solução proposta. O *solver* é composto por três módulos de atividades, o módulo de **Pré-processamento**, o módulo de **Gerenciamento da dinamicidade** e o módulo de **Otimização do roteamento**. A separação visa agrupar as funcionalidades para aumentar a coesão das partes relacionadas com propósitos similares. A coesão é importante para reduzir o acoplamento entre os módulos, com isso o *solver* se torna mais simples de entender e implementar. Por exemplo, os módulos de **Pré-processamento** e **Otimização do roteamento** somente se acoplam com o módulo de **Gerenciamento da dinamicidade**, dessa forma qualquer alteração no módulo de **Pré-processamento** só impactará no módulo de **Gerenciamento da dinamicidade**, sem interferir diretamente no funcionamento do módulo de **Otimização do roteamento**.

O módulo de **Pré-processamento** é responsável pela inicialização do ALNS-DPDP. Esse módulo gera a solução de roteamento inicial para atender todas as requisições estáticas recebidas antes do início do dia de trabalho. O fluxo de execução compreende a execução sequencial das seguintes tarefas (ver Figura 10):

- **Carregar requisições estáticas**: Nessa tarefa o algoritmo carrega todas as requisições estáticas anunciadas antes do horário de início do dia de trabalho (va-

Figura 10 – Solução proposta do *solver*. O módulo de pré-processamento é executado antes do início do dia de trabalho, enquanto os módulos de gerenciamento da dinamicidade e otimização do roteamento executam online ao longo do dia de trabalho.



Fonte: Imagem gerada pelo autor.

riável T_{start}). Em sistemas produtivos reais, o algoritmo pode acessar um arquivo de texto ou uma base de dados para obter essas informações;

- **Gerar solução inicial**: Nessa tarefa o algoritmo gera uma solução inicial para atender todas as requisições estáticas. É importante ressaltar que geralmente não são usados todos os veículos da frota porque o número de requisições estáticas não é superdimensionado. Nessa tarefa uma heurística de inserção baseada no critério de inserção guloso é aplicada. Maiores detalhes são apresentados na Seção 3.3.5;

- **Executar busca local na solução inicial**: Nessa tarefa o *solver* tenta melhorar a solução inicial aplicando algoritmos de busca local. Maiores detalhes sobre os algoritmos utilizados são apresentados na Seção 3.3.6;

- **Definir solução inicial como melhor solução**: Nessa tarefa a solução inicial é definida como a melhor solução encontrada até o momento pelo *solver*;

- **Atualizar veículos com a melhor solução**: Nessa tarefa os veículos são notificados sobre as suas rotas e podem iniciar seu período de trabalho. O horário de saída dos veículos é sincronizado com o horário de início da janela de tempo do depósito. Neste trabalho, como a janela de início do depósito começa no tempo T_{start}, todos os veículos que iniciam a operação também partem no tempo T_{start};
- **Iniciar *loop* do dia de trabalho**: Nesse momento acontece a passagem de bastão para o módulo de **Gerenciamento da dinamicidade**, ou seja, o algoritmo inicia o *loop* do dia de trabalho (Figura 9). Em paralelo, a linha do tempo ($T_{current}$) começa a avançar de forma contínua até o término do dia (janela de término do depósito). Maiores detalhes sobre o uso do ($T_{current}$) são apresentados na Seção 3.1.

O módulo de **Gerenciamento da dinamicidade** apresentado na Figura 10 é responsável por:

- **Gerenciar as requisições dinâmicas**: Quando novas requisições são anunciadas na central de atendimento, esse módulo insere elas na solução e notifica a frota de veículos acerca das atualizações. Nesse processo, novos veículos podem ser alocados caso a frota em operação não tenha disponibilidade para atender as novas requisições.
- **Monitorar a operação dos veículos**: Controla as requisições atendidas e a posição de cada veículo durante todo o dia de trabalho. Conforme os veículos vão atendendo os clientes, seguindo a solução de roteamento, as requisição mudam de estado e isso restringe as otimizações que podem ser feitas (ver Figura 8 e Seção 3.1.1).
- **Comunicar com o módulo de otimização do roteamento**: Faz as chamadas para o módulo de **Otimização do roteamento**, responsável por melhorar a solução de roteamento.

A execução do módulo **Gerenciamento da dinamicidade** é feita por um processo iterativo que executa durante todo o dia de trabalho (desde T_{start} até T_{end}). A execução termina quando o dia chega ao fim, nesse ponto os devidos cuidados devem ser tomados para garantir que todos os veículos retornem para o depósito no tempo certo. A cada iteração do módulo compreende-se a execução sequencial das seguintes tarefas (ver Figura 10):

- **Término do dia de trabalho?**: Nessa decisão, o algoritmo verifica se o tempo corrente de execução ($T_{current}$) é maior ou igual ao tempo de término do dia de

trabalho (T_{end}). Caso verdadeiro, a execução termina invocando a tarefa **Finalizar dia de trabalho**. Caso falso, o fluxo segue para a próxima tarefa de **Rastrear veículos** (ver Definição 3.1.1);

- **Rastrear veículos**: Nessa tarefa o estado de atendimento das requisições é atualizado. O objetivo é definir quais nós de coleta/entrega já foram visitados, quais estão em transição e quais estão em espera. Isso é feito comparando o tempo corrente ($T_{current}$) versus os tempos calculados das visitas (ver Seção 3.1.1). Em uma situação do mundo real, essa comparação poderia ser enriquecida com dados coletados pelo GPS do celular dos caminhoneiros;

- **A central de atendimento recebeu requisições dinâmicas?**: Nessa decisão o algoritmo consulta a central de atendimento por novas requisições dinâmicas. Se novas requisições foram anunciadas, então o fluxo é desviado para a tarefa de **Inserir requisições na melhor solução**. Caso contrário, o módulo de **Otimização do roteamento** é invocado;

- **Inserir requisições dinâmicas na melhor solução**: Nessa tarefa as requisições dinâmicas são inseridas na melhor solução utilizando a heurística de inserção apresentada na Seção 3.3.5. Novos veículos podem ser alocados caso os veículos em operação não consigam atender a nova demanda;

- **Executar busca local na melhor solução**: Nessa tarefa uma busca local é aplicada para melhorar a melhor solução. Mais detalhes sobre a busca local são apresentados na Seção 3.3.6;

- **Atualizar veículos com a melhor solução**: Essa tarefa é responsável por atualizar a frota de veículos acerca da melhor solução de roteamento até o momento. Na prática, os motoristas dos veículos seriam atualizados por meio de aplicativo de celular. Essa tarefa é o ponto de ligação do *loop* do algoritmo. A partir daqui se invoca a decisão **Término do dia de trabalho?**;

O último módulo, o módulo de **Otimização do roteamento** apresentado na Figura 10, é responsável por buscar uma nova melhor solução aplicando os algoritmos ALNSs. Esse módulo gerencia a prioridade dos objetivos intercalando a execução dos algoritmos $ALNS_{NV}$ e $ALNS_{TC}$. Basicamente, a cada iteração do *solver* é feita uma iteração do $ALNS_{NV}$ para tentar minimizar o número de veículos, caso a solução gerada não seja melhor que a melhor solução conhecida pelo *solver*, então uma interação do $ALNS_{TC}$ é executada para tentar otimizar o custo total. No final desse módulo, o fluxo é direcionado novamente para o módulo do **Gerenciamento da dinamicidade**. Quando uma solução gerada por um dos ALNS é otimizada, seja por reduzir NV ou por reduzir TC, então a melhor solução conhecida pelo *solver* é atualizada. A execução

desse módulo compreende a execução sequencial das seguintes tarefas (ver Figura 10):

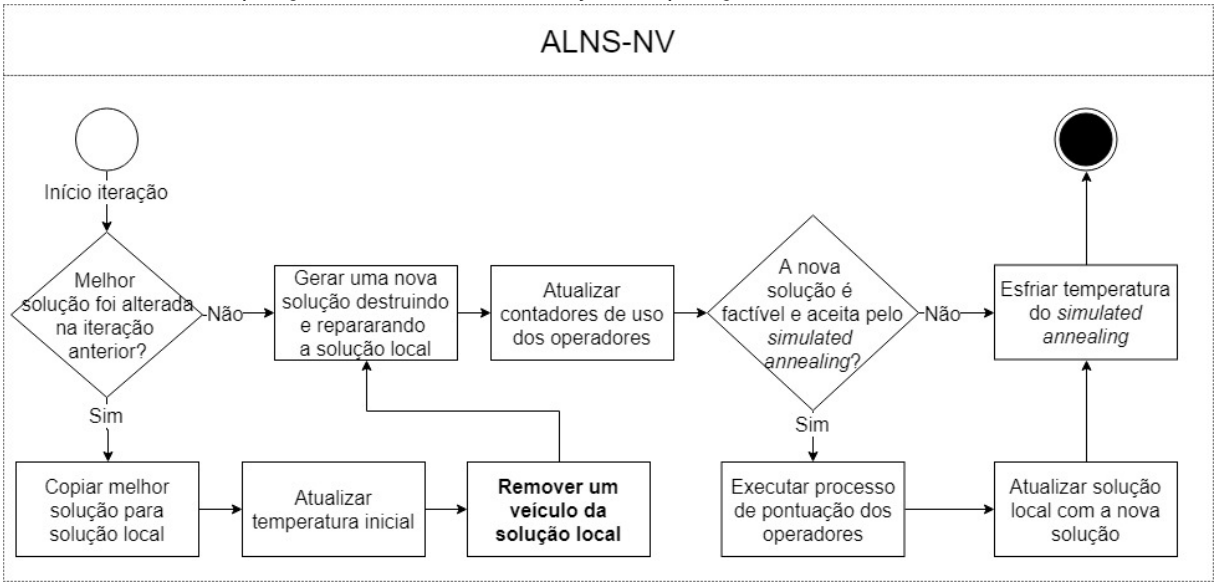
- **Executar uma iteração do ALNS-NV**: Essa tarefa realiza uma execução do algoritmo $ALNS_{NV}$. A chamada é feita passando a melhor solução do *solver* como parâmetro. Maiores detalhes são discutidos na Seção 3.2.2;
- **Solução gerada possuí um veículo a menos?**: Nessa decisão, caso o algoritmo $ALNS_{NV}$ encontre uma solução factível com menos veículos, então o fluxo é desviado para a tarefa de **Atualizar melhor solução com a solução gerada** e **Atualizar veículos com a melhor solução**. Caso contrário, o fluxo é desviado para a tarefa de **Executar uma iteração do ALNS-TC**. Maiores detalhes a cerca da factibilidade e restrições de uma solução serão discutidos na Seção 3.2.3;
- **Executar uma iteração do ALNS-TC**: Essa tarefa, faz a execução de uma iteração do algoritmo $ALNS_{TC}$. A chamada é feita passando a melhor solução do *solver* como parâmetro. Maiores detalhes são discutidos na Seção 3.2.2;
- **Solução gerada possuí menor custo total?**: Nessa decisão, se a solução retornada pelo $ALNS_{TC}$ for válida e tiver custo menor em relação à melhor solução, então as tarefas de **Atualizar melhor solução com a solução gerada** e **Atualizar veículos com a melhor solução** são invocadas. Caso contrário, o fluxo é desviado para o módulo de **Gerenciamento da dinamicidade**, mais especificamente para a decisão **Término do dia de trabalho?**.
- **Atualizar melhor solução com a solução gerada**: A melhor solução do solver é atualizada para a nova solução gerada.
- **Atualizar veículos com a melhor solução**: Essa tarefa é responsável por atualizar a frota de veículos acerca da nova melhor solução de roteamento.

Essa seção apresentou detalhadamente o fluxo de execução das tarefas realizadas pelo *solver* ALNS-DPDP. A seção seguinte irá apresentar de forma detalhada o fluxo de execução dos algoritmos $ALNS_{NV}$ e $ALNS_{TC}$.

3.2.2 Método proposto do ALNS-NV e ALNS-TC

Esta seção visa apresentar de forma detalhada o fluxo das tarefas executadas pelos algoritmos $ALNS_{NV}$ e $ALNS_{TC}$. Eles são apresentados nas Figuras 11 e 12, respectivamente. Os algoritmos são similares na maior parte, a diferença entre eles está demarcada com o texto em negrito. No caso do algoritmo $ALNS_{NV}$, ele executa a

Figura 11 – Solução algorítmica proposta do $ALNS_{NV}$ utilizado pelo *solver*. O algoritmo tem foco em otimizar o número de veículos. A estratégia é remover veículos continuamente e realocar suas requisições em outros veículos já em operação.



Fonte: Imagem gerada pelo autor.

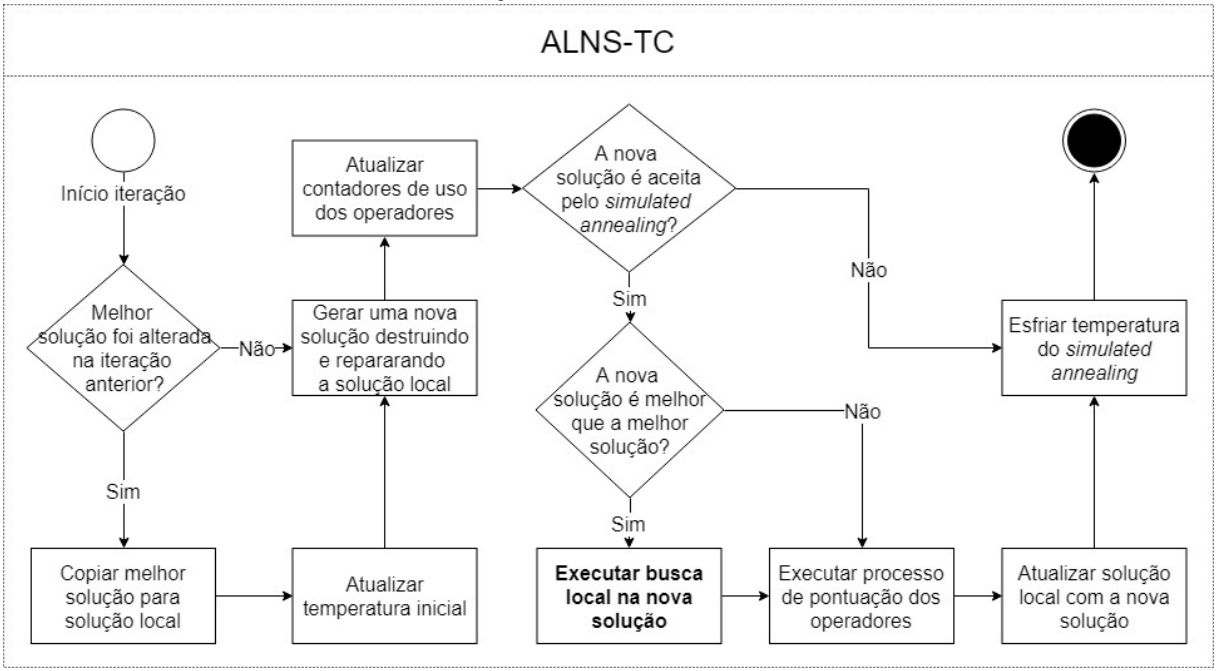
tarefa adicional de **Remover um veículo da solução**, enquanto o algoritmo $ALNS_{TC}$ executa a tarefa adicional de **Executar busca local na nova solução**.

Ambos os algoritmos $ALNS_{NV}$ e $ALNS_{TC}$ são baseados no algoritmo apresentado na Seção 2.4. Entretanto, para que cada algoritmo seja capaz de lidar com seu objetivo, o $ALNS_{NV}$ remove um veículo da solução sempre que uma solução factível é encontrada enquanto o $ALNS_{TC}$ otimiza a solução com busca local para reduzir o custo total do roteamento.

O $ALNS_{NV}$ requer mais exploração para buscar por soluções com menos veículos. Por outro lado, o $ALNS_{TC}$ requer mais intensificação para minimizar o custo total. No $ALNS_{NV}$ a exploração é importante visto que a realocação das requisições de um veículo removido modifica o espaço da otimização. Geralmente, essa realocação implica em aceitar soluções de maior custo total para fugir de ótimos locais. Essa característica exploratória tende a ser reduzida quando se aplicam buscas locais. Por esse motivo é que a busca local não é aplicada ao $ALNS_{NV}$. A listagem a seguir detalha o fluxo de execução das tarefas executadas nas Figuras 11 e 12:

- **Início da iteração**: Ponto de entrada de ambos os algoritmos. Basicamente, é quando o módulo de **Otimização do roteamento** (apresentado na Figura 10) invoca uma iteração do ALNS;
- **Melhor solução foi alterada na iteração anterior?**: Nessa decisão o algoritmo verifica se a melhor solução do *solver* foi alterada em relação a solução local mantida pelo algoritmo;

Figura 12 – Solução algorítmica proposta do $ALNS_{TC}$ utilizado pelo *solver*. Algoritmo focado em otimizar o custo total da solução. A execução é combinada com operadores de busca local para aumentar a ênfase na otimização do custo total.



Fonte: Imagem gerada pelo autor.

- **Copiar melhor solução para solução local**: Visa manter a consistência entre a solução do *solver* e dos algoritmos ALNS atualizando a referência da solução local do ALNS. Isso acontece quando, por exemplo, o *solver* insere requisições dinâmicas na melhor solução;

- **Atualizar temperatura inicial**: Recalcula a temperatura inicial aplicando a Equação 2.22. Isso é feito para manter a exploração do espaço de busca sincronizada com a escala de custo da melhor solução. Por exemplo, se o módulo de **Gerenciamento da dinamicidade** (Figura 10) inserir 10 requisições dinâmicas na melhor solução e o ALNS não atualizar a temperatura, isso implica em uma exploração mais difícil porque a temperatura não considera o custo adicional das 10 requisições;

- **Remover um veículo da solução local**: Essa tarefa só existe no $ALNS_{NV}$ (Figura 11). Para otimizar o número de veículos, ela remove um veículo arbitrário da solução e executa o ALNS até que uma nova solução factível seja encontrada sem aquele veículo;

- **Gerar uma nova solução destruindo e reparando a solução local**: Executa procedimentos de otimização padrões do ALNS (ver Seção 2.4);

- **Atualizar contadores de uso dos operadores**: Aplica o procedimento de atualização dos contadores de uso dos operadores do ALNS (ver Seção 2.4);

- **A solução é factível e aceita pelo *simulated annealing*?** Verifica se a nova solução é válida (factível, atende todas restrições) e foi aceita pela função de aceitação, então ela pode seguir no fluxo de atualização da solução local do ALNS (ver Seção 2.4.5 e Seção 3.2.3);
- **Executar processo de pontuação dos operadores**: Atualiza os pontos dos operadores usados na geração da nova solução conforme a qualidade da solução (ver Seção 2.4);
- **Atualizar solução local com a nova solução**: Atualiza a solução local com a nova solução gerada;
- **Esfriar temperatura do *simulated annealing***: Diminui a temperatura do algoritmo conforme o fator de resfriamento linear (ver Equação 2.23);
- **A nova solução é melhor que a melhor solução?**: Essa decisão é padrão do ALNS, porém ela só é demonstrada na figura no $ALNS_{TC}$ (Figura 12) porque ela serve como critério para execução da busca local. Nesse momento a Equação 3.2 é aplicada;
- **Executar busca local na nova solução**: Essa tarefa só existe para o $ALNS_{TC}$ (Figura 12). Nessa tarefa são aplicados os algoritmos de busca local sob a solução para tentar otimizar ainda mais o seu custo (ver Seção 3.3.6);
- **Término iteração**: Retorna o controle para o módulo de **Otimização do roteamento** do *solver* junto com a nova solução gerada (Figura 10). Se a iteração finalizou um segmento, então os pesos dos operadores são recalculados nessa fase (ver Seção 2.4 e Equação 2.10).

A seguir serão detalhados outros aspectos da solução, tais como: função objetivo da otimização (Seção 3.2.3), operadores de remoção e inserção (Seções 3.3.3 e 3.3.4, respectivamente), heurística de inserção de novas requisições (Seção 3.3.5) e heurísticas de busca local (Seção 3.3.6)

3.2.3 Função objetivo

A Equação 3.2 apresenta a função de escolha definida para o problema em questão. Essa função seleciona a melhor entre duas soluções factíveis, respeitando a hierarquia entre os dois objetivos NV e TC. Diferente da função proposta por Dumas, Desrosiers e Soumis (1991), essa função não considera a soma dos termos NV e TC como critério de escolha da melhor solução. No caso, ela se baseia na proposta apresentada de Pisinger e Ropke (2007) (ver Seção 2.6) onde os objetivos são tratados separadamente, visto que o ALNS não é preparado para lidar com os dois

objetivos conjuntos. Entretanto, diferente de Pisinger e Ropke (2007) onde o algoritmo $ALNS_{NV}$ executa antes do $ALNS_{TC}$ e possuí um critério de otimização para cada algoritmo, nesse trabalho o critério é definido pela Eq. 3.2 porque os algoritmos são executados de forma intercalada. Assim, é possível garantir que a cada iteração do *solver* a melhor solução seja sempre escolhida.

$$best(S', S) = \begin{cases} \text{se } NV(S') < NV(S), & \text{retorne } S' \\ \text{se } NV(S') = NV(S) \wedge TC(S') < TC(S), & \text{retorne } S' \\ \text{caso contrário}, & \text{retorne } S \end{cases} \tag{3.2}$$

Além da Equação 3.2, o funcionamento do *solver* depende também da função de custo total usada para: 1) calcular a probabilidade de aceitação do SA (*simulated annealing*) no ALNS, 2) no caso do $ALNS_{TC}$, otimizar a solução e 3) validar se todas restrições do problema estão sendo atendidas. As restrições indicam a factibilidade (ou validade) de uma solução, ou seja, uma solução é considerada factível se e somente se todas as restrições são atendidas. Portanto, a Equação 3.3a é responsável pela minimização do custo total sujeita as restrições definidas pelo problema. Essa equação é baseada na proposta original de Dumas, Desrosiers e Soumis (1991) (Equação 2.1a) para o problema do PDPTW, porém a minimização do número de veículos. Um ponto importante é que, como esse trabalho trata da versão dinâmica do problema (DPDPTW), a formulação foi estendida para adicionar uma restrição que valida se o início do atendimento de uma requisição é condizente com o horário que ela foi anunciada (Equação 3.3n).

$$\min_{cost(S)} \quad \vartheta \sum_{i \in P^+} (1 - \sum_{v \in V} \sum_{j \in N} X_{ij}^v) + \sum_{v \in V} \sum_{i \in N} \sum_{j \in N} t_{ij} X_{ij}^v \tag{3.3a}$$

s.t.

$$\sum_{v \in V} \sum_{j \in N} X_{ij}^v = 1, \ i \in P^+, \tag{3.3b}$$

$$\sum_{j \in N} X_{ij}^v - \sum_{j \in N} X_{ji}^v = 0, \ i \in P, v \in V, \tag{3.3c}$$

$$\sum_{i \in P^+} X_{0i}^v \leq 1, \ v \in V, \tag{3.3d}$$

$$\sum_{i \in P^-} X_{i,2M+1}^v \leq 1, \ v \in V, \tag{3.3e}$$

$$\sum_{j \in N} X_{ij}^v - \sum_{j \in N} X_{j,i+M}^v = 0, \ i \in P^+, v \in V, \tag{3.3f}$$

$$T_i + s_i + t_{i,M+i} \leq T_{M+i}, \ i \in P^+, \tag{3.3g}$$

$$X_{ij}^v = 1 \Rightarrow T_i + s_i + t_{ij} \leq T_j, \ i, j \in P, v \in V, \tag{3.3h}$$

$$X_{0j}^v = 1 \Rightarrow T_0^v + t_{0j} \leq T_j, \ j \in P^+, v \in V, \tag{3.3i}$$

$$X_{i,2M+1}^v = 1 \Rightarrow T_i + s_i + t_{i,2M+1} \leq T_{2M+1}^v, \ i \in P^-, v \in V, \tag{3.3j}$$

$$e_i \leq T_i \leq l_i, \ i \in P, \tag{3.3k}$$

$$e_0 \leq T_0^v \leq l_0, \ v \in V, \tag{3.3l}$$

$$e_{2M+1} \leq T_{2M+1}^v \leq l_{2M+1}, \ v \in V, \tag{3.3m}$$

$$T_{prev(i)} + s_{prev(i)} \geq A_i, \ i \in P, \tag{3.3n}$$

$$X_{ij}^v = 1 \Rightarrow Y_i + q_j = Y_j, \ i \in P, j \in P^+, v \in V, \tag{3.3o}$$

$$X_{ij}^v = 1 \Rightarrow Y_i - q_j = Y_j, \ i \in P, j \in P^-, v \in V, \tag{3.3p}$$

$$X_{0j}^v = 1 \Rightarrow Y_0 + q_j = Y_j, \ j \in P^+, v \in V, \tag{3.3q}$$

$$Y_0 = 0, \ q_i \leq Y_i \leq C, \ i \in P^+, \tag{3.3r}$$

$$X_{ij}^v \in \{0, 1\}, \ i, j \in N, v \in V \tag{3.3s}$$

Diferente da Equação 2.1a de Dumas, Desrosiers e Soumis (1991), a Equação 3.3a usa como primeiro termo a penalização por requisições não atendidas (PISINGER; ROPKE, 2007). Esse termo permite que o ALNS remova requisições da solução, deixando a solução temporariamente com custo elevado e infactível, porém permitindo explorar melhor o espaço de busca. O termo de penalização aumenta o custo da solução em um fator fixo (fator ϑ) para cada requisição não atendida (variável binária X_{ij}^v). Para isso é requerido que o fator ϑ seja suficientemente alto, caso contrário o algoritmo terá dificuldade em identificar o ganho entre uma solução que atende todas as requisições e uma que não atende.

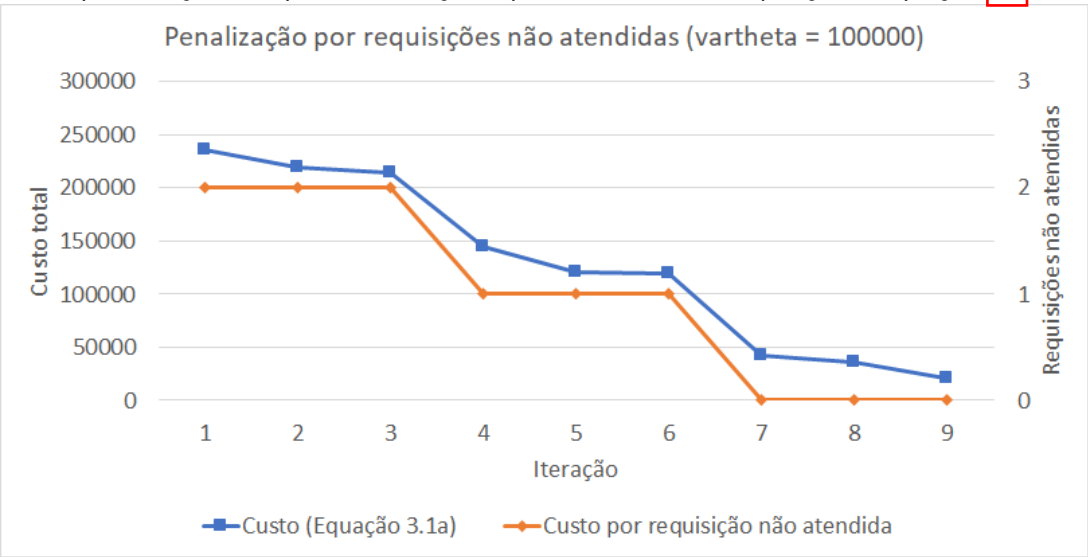
O segundo termo da função de minimização da Equação 3.3a calcula o custo total (TC) da viagem feita pelos veículos. Esse custo pode ser calculado em termos de distância percorrida, tempo de viagem, consumo de diesel, entre outros. Para o problema em questão, o custo está associado ao tempo de viagem. Aqui é importante ressaltar que o tempo de viagem considera apenas o tempo que o veículo está em movimento, ou seja, se o veículo está atendendo um cliente ou esperando iniciar a janela de tempo então esse tempo não é contabilizado. A ideia desse termo é medir o custo operacional dos veículos associado ao desgaste do equipamento, gasto com combustível, emissão de gás carbônico, entre outros.

Os termos de penalização e custo total, quando combinados, resultam em platôs no gráfico da função de custo total. Os platôs indicam ao algoritmo o ganho relacionado com atender o maior número possível de requisições, mantendo a otimização do custo total. Para demonstrar o impacto da penalização, a Figura 13 apresenta uma análise de um cenário hipotético simulando a otimização de uma solução ao longo de 9 iterações, nesse cenário $\vartheta = 100000$. Nas iterações 1 à 3 a solução começa com duas requisições não atendidas, ou seja, é acrescido um custo de $2\vartheta = 200000$ no custo da viagem. Em seguida, o algoritmo consegue alocar uma requisição em um veículo durante as iterações 4 à 6, onde agora é acrescido um custo de $1\vartheta = 100000$ na solução. E finalmente, na iteração 7 o algoritmo consegue alocar todas as requisições, nesse ponto o custo da solução está somente em função do custo da viagem. A importância de ϑ ser consideravelmente alto serve para priorizar o atendimento das requisições em relação ao custo total. Por exemplo, sem o fator ϑ o algoritmo não aceitaria a solução da iteração 4 (que ignorando ϑ é de aprox. 45000) porque o custo da iteração 3 (que ignorando ϑ é de aprox. 14000) é menor, embora atenda menos requisições.

Na função objetivo proposta na Equação 3.3a, a minimização dos veículos é guiada pelo custo das requisições não atendidas. Portanto, quando um veículo é removido pelo $ALNS_{NV}$, as requisições dele se tornam não atendidas, forçando que o custo total da solução aumente pela penalização do fator ϑ. Dessa forma, o ALNS tentará realocar essas requisições em outros veículos, até a penalização ser igual a zero. Quando todas as requisições forem alocadas, então o $ALNS_{NV}$ remove mais um veículo para continuar a otimização.

Durante a execução do ALNS a Equação 3.3a é calculada em diversos momentos diferentes para avaliar o custo das soluções. Em relação a Figura 10 a avaliação ocorre em todas tarefas que alteram o arranjo dos nós na solução, por exemplo: ao gerar uma solução inicial, ao inserir requisições dinâmicas ou otimizar a solução atual. A avaliação da Equação 3.3a pode ocorrer de forma completa ou parcial. Na avaliação completa a Equação 3.3a é aplicada para todos os veículos da solução, isso

Figura 13 – Exemplo de curva de convergência da Equação 3.3a. Demonstra o calculo em função do custo total dos veículos com rotas e da penalização total por requisições não atendidas. A penalização visa priorizar soluções que atendam mais requisições, Equação 3.2.



Fonte: Imagem gerada pelo autor.

implica em um tempo de computação maior. Já na avaliação parcial é avaliado o custo e restrições para um veículo único, isso implica em um tempo computacional menor. A avaliação parcial serve para reduzir as computações dos operadores de inserção e remoção. Quando um operador de inserção verifica se uma requisição pode ser inserida em um veículo, não é necessário executar a Equação 3.3a para todos veículos, ao invés disso pode-se aplicar a equação somente naquele veículo.

O método proposto nesse trabalho deve atender a todas às restrições da Equação 3.3a. Portanto, deve atender as Restrições 3.3b e 3.3c, que indicam que todos os nós devem ser visitados uma única vez, possuindo uma aresta de entrada e uma de saída. As Restrições 3.3d e 3.3e, que representam a forma construtiva das rotas, ou seja, todo veículo deve partir do depósito para uma coleta e terminar a rota partindo de uma entrega e chegando no depósito. As Restrições 3.3f e 3.3g, que definem paridade e precedência, ou seja, toda requisição deve ser coletada e entregue pelo mesmo veículo e a coleta deve preceder a entrega. As Restrições 3.3h, 3.3i e 3.3j, que definem a ordem com que as visitas devem ser feitas em função do agendamento. As Restrições 3.3k, 3.3l e 3.3m, que definem o cumprimento das janelas de tempo. A restrição 3.3n que garante que um requisição tenha sido anunciada antes do horário de partida do nó anterior na rota do veículo. As Restrições 3.3o, 3.3p, 3.3q e 3.3r, que definem o cumprimento da capacidade de carga dos veículos. E, por fim, o domínio da variável X que é definido pela Restrição 3.3s.

Por fim, vale ressaltar que no problema em questão as restrições das janelas de tempo são atendidas de forma restrita. Embora seja possível relaxar essas res-

trições penalizando a solução por atrasos no atendimento dos clientes, o problema não considera essa flexibilidade. Esse fator é importante porque ele afeta o tamanho do espaço de busca das soluções, visto que com janelas de tempo mais relaxadas, mais soluções podem ser geradas e logo a capacidade de convergência do algoritmo é reduzida.

3.3 MODELAGEM DOS ALGORITMOS

Essa seção visa apresentar em maiores detalhes os algoritmos do ALNS-DPDP. Muitos desses algoritmos já foram apresentados na Seção 2. Entretanto, várias modificações tiveram de ser feitas para atender as restrições de dinamicidade apresentadas na Seção 3.1. O ALNS-DPDP proposto é composto de um algoritmo *solver* e dois algoritmos ALNS.

A seguir será apresentado o detalhamento desses algoritmos, na Seção 3.2.1 será apresentado o *solver* e na Seção 3.2.2 será apresentado o ALNS. Além disso, serão apresentadas as sub-rotinas necessárias para o funcionamento dos algoritmos. Portanto, nas Seções 3.3.3, 3.3.4, 3.3.5 e 3.3.6 serão discutidas as adaptações realizadas nessas sub-rotinas. Por fim, a Seção 3.4 detalha a forma de implementação dos algoritmos.

3.3.1 Algoritmo *Solver*

O algoritmo *solver* é o principal algoritmo do método proposto. Ele é responsável por gerenciar a dinamicidade das requisições e coordenar a execução dos algoritmos $ALNS_{NV}$ e $ALNS_{TC}$. O pseudocódigo do *solver* é apresentado no Algoritmo 7. Esse pseudocódigo basicamente contempla os módulos de **Pré-processamento**, **Gerenciamento da dinamicidade** e **Otimização do roteamento** apresentados na Figura 10.

O Algoritmo 7 inicia a execução pelo módulo de **Pré-processamento**. A primeira tarefa desse módulo é executar o carregamento das requisições estáticas (linha 1). Se não existir nenhuma requisição, as tarefas da solução inicial são ignoradas. Entretanto, se existirem requisições, então o algoritmo gera a solução inicial. Isso é feito chamando-se a sub-rotina da heurística de inserção (linha 2), seguido da sub-rotina de busca local (linha 3), que visa melhorar a qualidade da solução inicial. Com a solução inicial o algoritmo notifica os veículos para eles iniciarem suas operações (linha 4). A última etapa desse módulo é atualizar o tempo de referência ($T_{current}$) com o tempo de início da operação (linha 5).

Realizado o **Pré-processamento** (linhas 1-5), o Algoritmo 7 inicia o módulo de **Gerenciamento da dinamicidade** (linhas 6-18). Nesse momento o *solver* começa

Algoritmo 7 Algoritmo do solver, responsável por gerenciar a dinamicidade do problema e coordenar a execução dos algoritmos $ALNS_{NV}$ e $ALNS_{TC}$.

Require: T_{start}: Horário de início, T_{end} Horário de término
1: $R_{static} \leftarrow$ Carregar requisições estáticas ▷ Pré-processamento
2: $S_{best} \leftarrow$ Gerar solução inicial com R_{static} usando Algoritmo 11
3: Executar busca local em S_{best} usando Algoritmos 12
4: Atualizar veículos com S_{best}
5: $T_{current} \leftarrow T_{start}$
6: **while** $T_{current} \leq T_{end}$ **do** ▷ Gerenciamento da dinamicidade
7: $T_{current} \leftarrow$ Tempo atual
8: Aplicar restrições temporais (Seção 3.1.1).
9: **if** Alguma requisição mudou de estado **then**
10: Atualizar solução local do $ALNS_{NV}$
11: **end if**
12: $R_{dynamic} \leftarrow$ Obter requisições dinâmicas da central de atendimento
13: **if** $R_{dynamic} \neq \emptyset$ **then**
14: Inserir $R_{dynamic}$ em S_{best} usando Algoritmo 11
15: Executar busca local em S_{best} usando Algoritmo 12
16: Atualizar solução local do $ALNS_{NV}$ e $ALNS_{TC}$
17: Atualizar veículos com S_{best}
18: **end if**
19: $S' \leftarrow$ Executar uma iteração do $ALNS_{NV}$ ▷ Otimização do roteamento
20: **if** $best(S', S_{best}) = S'$ **then**
21: $S_{best} \leftarrow S'$
22: Atualizar solução local do $ALNS_{TC}$
23: Atualizar veículos com S_{best}
24: **else**
25: $S'' \leftarrow$ Executar uma iteração do $ALNS_{TC}$
26: **if** $best(S'', S_{best}) = S''$ **then**
27: $S_{best} \leftarrow S''$
28: Atualizar veículos com S_{best}
29: **end if**
30: **end if**
31: **end while**

a acompanhar o dia de trabalho, ou seja, enquanto o fim do dia não chegar o método continua executando o laço de repetição (linha 6). A primeira função dentro do laço é atualizar o valor do $T_{current}$ com o tempo atual (linha 7).

Com base no valor de $T_{current}$ o Algoritmo 7 executa os cálculos das restrições temporais (linhas 8-11). Nesse momento, aplicam-se as regras de restrição em função do tempo e das visitas realizadas pelos veículos para atualizar o estado de atendimento das requisições (Seção 3.1.1). Quando uma requisição troca de estado (linha 9), a solução local do $ALNS_{NV}$ é atualizada (linha 10), com isso o *solver* garante que a solução local do $ALNS_{NV}$ não seja divergente da melhor solução sendo usada pelos veículos. Essa atualização somente ocorre para o $ALNS_{NV}$ por causa do seu procedimento de remoção de veículo (ver Seção 3.3.2).

No Algoritmo 7, o gerenciamento das requisições dinâmicas é feito nas linhas 12-18. Nesse momento, o algoritmo verifica com a central de atendimento se novas requisições foram anunciadas (linha 12). Se for esse o caso (linha 13), a inserção é feita usando a heurística de inserção e a busca local (linhas 14-15), seguido da atualização da melhor solução nos algoritmos ALNS e da atualização das rotas dos veículos (linhas 16-17). Quando novas requisições são anunciadas, os algoritmos ALNS devem ter suas soluções locais atualizadas para recalcularem suas temperaturas e sincronizarem as rotas (linha 16). Caso contrário, a função de aceitação do *Simulated Annealing* de cada ALNS terá dificuldade em realizar a exploração do espaço de busca.

A execução dos algoritmos ALNS do módulo de **Otimização do roteamento** é feito nas linhas 19-30 do Algoritmo 7. A primeira etapa do algoritmo executa uma iteração do $ALNS_{NV}$ (linha 19), se a solução for melhor (linha 20), então ocorre a atualização dos seguintes itens: melhor solução, $ALNS_{TC}$ e frota de veículos (linhas 21-23). A comparação das duas soluções é feita com base na Equação 3.2, ou seja, se o número de veículos (NV) da solução S' for menor que da solução S_{best}, mesmo que o custo total (TC) seja maior, a solução S_{best} será atualizada. Um ponto importante em relação a condição da linha 20 é que o $ALNS_{NV}$ só vai conseguir remover veículos que ainda não iniciaram o atendimento, caso contrário haveria uma quebra de factibilidade em relação as restrições apresentadas na Equação 3.3a (ver Seção 3.3.2). Nesse algoritmo, se o $ALNS_{NV}$ não reduziu o número de veículos então o $ALNS_{TC}$ tenta reduzir o custo total (linha 25). Nesse caso, se a nova solução do $ALNS_{TC}$ tiver menor custo total (linha 26), segundo a Equação 3.2, então a melhor solução e os veículos serão atualizados (linhas 27-28).

3.3.2 Algoritmo ALNS

Essa seção visa descrever em detalhe os algoritmos $ALNS_{NV}$ e $ALNS_{TC}$ invocados pelo módulo de **Otimização do roteamento** (Figura 10). Embora ambos os algoritmos tenham comportamentos diferentes, a implementação é muito similar. As diferenças estão nas mudanças de fluxos apresentados nas tarefas em negritos das Figuras 11 e 12 e nos valores dos parâmetros de execução. Portanto, para simplificar o detalhamento, os algoritmos $ALNS_{NV}$ e $ALNS_{TC}$ foram unificados no Algoritmo 8.

Algoritmo 8 Pseudocódigo do algoritmo ALNS proposto para o DPDPTW. A execução depende da melhor solução do solver (S_{solver}) e do objetivo de otimização, ($F = NV$ ou $F = TC$).

Require: S_{solver}: Melhor solução do solver, F: Objetivo de otimização
if $S_{best} \neq S_{solver}$ **then**
    $S_{best} \leftarrow$ Copiar S_{solver}
    $S_{local} \leftarrow$ Copiar S_{best}
    $\tau \leftarrow$ Calcular temperatura inicial usando Eq. 2.22
    if $F = NV$ **then**
        Remover veículo arbitrário de S_{best}
    end if
end if
$q \leftarrow$ Eq. 2.15, Gerar número para remoção
$r \leftarrow$ Eq. 2.9, Selecionar operador em O^{rem}
$i \leftarrow$ Eq. 2.9, Selecionar operador em O^{ins}
$n \leftarrow$ Eq. 2.9, Selecionar ruído em O^{noise}
$S' \leftarrow$ Executar $O_r^{rem}(S_{local}, q)$
$S' \leftarrow$ Executar $O_i^{ins}(S', O_n^{noise})$
Incrementar contadores $u_{O_r^{rem}}$, $u_{O_i^{ins}}$ e $u_{O_n^{noise}}$
if $accept(S', S_{local}, \tau)$ **then**
    if $best(S', S_{best}) = S'$ **then**
        if $F = TC$ **then**
            $S_{best} \leftarrow$ Executar busca local em S' (Algoritmo 7)
        end if
        $S_{best} \leftarrow S'$
        Atualizar $\phi_{O_r^{rem}}$, $\phi_{O_i^{ins}}$ e $\phi_{O_n^{noise}}$ com θ_1
    else if $best(S', S_{local}) = S'$ **then**
        Atualizar $\phi_{O_r^{rem}}$, $\phi_{O_i^{ins}}$ e $\phi_{O_n^{noise}}$ com θ_2
    else
        Atualizar $\phi_{O_r^{rem}}$, $\phi_{O_i^{ins}}$ e $\phi_{O_n^{noise}}$ com θ_3
    end if
    $S_{local} \leftarrow S'$
end if
$\tau \leftarrow \tau \times \gamma$
if Terminou segmento k **then**
    Atualizar pesos $\omega_{O^{rem}}$, $\omega_{O^{ins}}$ e $\omega_{O^{noise}}$ usando Eq. 2.10
end if

O Algoritmo 8 é gerenciado pelo módulo de **Otimização do roteamento** do Algoritmo 7. Para cada objetivo do problema, duas instâncias do Algoritmo 8 são criadas, uma para o $ALNS_{NV}$ e outra para o $ALNS_{TC}$. A variável F define qual objetivo será otimizado, se $F = NV$ então o algoritmo irá minimizar o número de veículos, caso contrário, se $F = TC$, então o algoritmo irá minimizar o custo total. Além disso, a definição dos parâmetros de execução (ex: γ e ε) dependem do objetivo definido por F. Na Seção 4.1.3 será discutido acerca dos parâmetros para cada objetivo.

A execução do Algoritmo 8 inicia verificando se a melhor solução do *solver*, S_{solver}, foi alterada na última iteração (linha 1). Se uma alteração ocorreu, então o processo de atualização da solução base do algoritmo é executado (linhas 2-7). Nesse processo atualiza-se as referência das soluções (linhas 2-3) e a temperatura inicial (linha 4). Caso o objetivo do algoritmo seja $F = NV$, (linha 5), então remove-se um veículo arbitrário da solução (linha 6). No geral, esse procedimento de atualização é importante pelos seguintes motivos:

- A solução S_{solver} foi alterada: a alteração ocorre pela inserção de novas requisições, otimização da solução ou alteração do estado das requisições (Ver Seção 3.3.1). Nesses casos, quando as soluções são diferentes é importante atualizar os algoritmos para que o ALNS não otimize uma solução desatualizada.

- A alteração de S_{solver} mudou a escala de temperatura: Nesse caso, quando as soluções forem diferentes, o ALNS precisa recalcular a temperatura inicial. Isso serve para manter a sincronização da temperatura entre as soluções S_{solver} e S_{best}. Se a re-parametrização não for aplicada, a função de aceitação poderá ter dificuldades em explorar o espaço de busca.

- No caso do $ALNS_{NV}$, um veículo deve ser removido: Como a solução de S_{solver} é sempre factível,o algoritmo remove um veículo para reiniciar a busca em torno da otimização dos veículos. Caso nenhum veículo possa ser removido (se todos já partiram do depósito) então o $ALNS_{NV}$ não conseguirá otimizar a solução.

No Algoritmo 8, a partir da linha 9 a única diferença em relação ao algoritmo da literatura (Seção 2.4) está na execução da busca local (linhas 18-20). O objetivo dessa busca é otimizar o custo total da solução (Equação 3.3a). A busca local é um processo computacional caro, por esse motivo a mesma só é executada ao encontrar uma nova melhor solução. Além disso, o $ALNS_{NV}$ não executa busca local porque para minimizar o número de veículos o algoritmo precisa ter facilidade em aumentar o custo total, ou seja, aumentar a exploração da busca. Quando um veículo é removido, a exploração é necessária para redistribuir essa carga em outros veículos, ou seja, aumentar a rota dos outros veículos em operação.

A escolha dos operadores de inserção e remoção é determinante para a qualidade das soluções geradas pelo ALNS. É importante que sejam escolhidos operadores que foquem em diferentes características do problema. Uma boa escolha de operadores permite ao algoritmo cobrir uma área maior do espaço de busca, visto que operadores diferentes consideram características diferentes do problema. Isso se converte em resultados de melhor qualidade. Os operadores selecionados estão apresentados na Tabela 3. Esses operadores são basicamente os mesmos do algoritmo padrão (ver Seção 2.4). A diferença é que para atender a problemas dinâmicos (DPDPTW) eles foram modificados em alguns pontos. As seções seguintes irão discutir os detalhes das modificações realizadas.

Tabela 3 – Operadores de inserção e remoção utilizados pelos algoritmos $ALNS_{NV}$ e $ALNS_{TC}$ nas etapas de remoção e inserção de requisições.

Tipo	Nome do operador	Descrição
Remoção	Random	Remove requisições baseado no critério aleatório (Seção 2.4.2.1).
Remoção	WorstRemoval	Remove requisições baseado no critério de custo agregado por requisição (Seção 2.4.2.2).
Remoção	Shaw	Remove requisições baseado no critério de similaridade (Seção 2.4.2.3).
Inserção	Greedy	Insere requisições pelo critério de melhor custo (Seção 2.4.3.1).
Inserção	Regret2	Insere requisições pelo critério de arrependimento, nível 2 (Seção 2.4.3.2).
Inserção	Regret3	Insere requisições pelo critério de arrependimento, nível 3 (Seção 2.4.3.2).
Inserção	Regret4	Insere requisições pelo critério de arrependimento, nível 4 (Seção 2.4.3.2).
Inserção	RegretM	Insere requisições pelo critério de arrependimento, nível M. Onde M é dado pelo número de veículos em operação (Seção 2.4.3.2).

3.3.3 Operadores de remoção

O objetivo de cada operador é explorar uma parte diferente do espaço de busca. Neste trabalho, os operadores utilizados para remoção são os mesmos dos Algoritmos 2, 3 e 4, revisados na Seção 2.4.2. Porém, dadas as restrições temporais e dinâmicas apresentadas na Seção 3.1, todos os algoritmos foram adaptados para considerar os estados dos nós. As adaptações tiveram como objetivo impedir que requisições com estado *Committed* ou *Transition* possam ser removidas de uma rota.

Embora os três operadores de remoção sejam algoritmos diferentes, todos eles chamam a mesma rotina de remoção de uma requisição. Dessa forma, foi necessário alterar somente essa rotina para adaptar todos os operadores de remoção. A versão adaptada é apresentada no Algoritmo 9. Nesse algoritmo, H_v é a rota em que a requisição R_i será removida. Se a requisição R_i possuí P_i^+ em espera (estado *Idle*), então a requisição pode ser removida. Caso contrário, a remoção é ignorada e o algoritmo segue o fluxo. Nessa adaptação, mesmo que a requisição R_i não seja removida, ela é considerada na lista de y requisições removidas. Isso permite os operadores de inserção tentar re-arranjar o nó de entrega em uma posição melhor (se o mesmo estiver em espera).

Algoritmo 9 Algoritmo de remoção de uma requisição associada a requisição R_i de um veículo H_v.

Require: H_v: Rota do veículo v, R_i: Requisição i para remoção
1: **if** $isIdle(P_i^+)$ **then**
2: $H_v \leftarrow H_v \setminus \{P_i^+\}$
3: $H_v \leftarrow H_v \setminus \{P_{M+i}^-\}$
4: **end if**

3.3.4 Operadores de inserção

Essa seção visa discutir sobre os operadores de inserção. Para esse trabalho foram usados os mesmos operadores da implementação padrão apresentada na Seção 2.4.3. Os operadores são a inserção gulosa e a inserção baseada no critério de arrependimento. A inserção gulosa foca na requisição de maior ganho no momento, enquanto a inserção baseada em critério de arrependimento verifica a perda potencial caso a requisição não seja inserida no momento. A combinação desses dois tipos de operadores de inserção é interessante porque eles se contra-balanceiam.

Assim como nos operadores de remoção, os operadores de inserção também possuem um processo de geração de aleatoriedade. Basicamente, aplica-se um fator de aleatoriedade sobre o custo calculado de uma inserção, conforme apresentado na Equação 2.20 da Seção 2.4.4. Esse processo é importante para aumentar a exploração do espaço de busca.

Assim como nos operadores de remoção, a adaptação dos operadores de inserção visa atender as restrições temporais (Seção 3.1). Para isso, a alteração foi proposta no Algoritmo 5 que é responsável por calcular o custo de inserção de uma requisição em um veículo. A alteração foi feita nesse algoritmo porque ele avalia o arranjo das rotas em todas as inserções. O Algoritmo 10 é a versão adaptada do Algoritmo 5. Nesse algoritmo, dado H_v como a rota de um veículo v e dado R_i como a requisição a ser inserida, o Algoritmo 10 busca determinar qual é a melhor posição

em H_v para inserir R_i, respeitando as restrições temporais. Se para a requisição R_i, P_i^+ não está em "Em espera" (foi coletada) e se P_{i+M}^- está "Em espera" (ainda não foi entregue), então existe somente um veículo $H_v, v \in V$ no qual o algoritmo 10 pode tentar uma inserção melhor para o nó de entrega (os outros veículos serão ignorados). Agora, se o nó de coleta não foi atendido (P_i^+ está "Em espera"), então a inserção dos nós de entrega e coleta da requisição R_i serão calculada em H_v, tal qual também serão avaliados nos outros veículos $H_{v'}, \forall v' \in V \setminus v$.

O funcionamento detalhado do Algoritmo 10 ocorre da seguinte forma. Primeiro, o algoritmo determina qual é a primeira posição em H_v com status *Em espera* (linha 1). Isso é importante porque todas as posições maiores pos_{start} podem ser rearranjas na solução. Em seguida, o algoritmo verifica se P_i^+ está *Em espera*, isso quer dizer que a requisição pode ser alocada no veículo v (linha 2). Então, o algoritmo tenta determinar a posição $pos_{P_i^+}$ para inserir P_i^+ em H_v (linha 3), caso essa posição não exista (linha 4) o algoritmo retorna (linha 5) porque ele não pode alocar essa requisição nesse veículo dadas as restrições. Caso contrário, se $pos_{P_i^+}$ existe, então pos_{start} é atualizada (linha 7) para manter a restrição de precedência com P_i^-.

Algoritmo 10 Algoritmo para avaliar a inserção de uma determinada requisição i em um veículo v

Require: Requisição i para inserção, rota H_v de um veículo v para avaliação

1: $pos_{start} \leftarrow$ determina primeira posição em H_v onde o nó possui o status *Idle*
2: **if** isIdle(P_i^+) **then**
3: $\quad pos_{P_i^+} \leftarrow$ determina posição de P_i^+ em H_v a partir de pos_{start} com Algoritmo 6
4: $\quad$ **if** $pos_{P_i^+} = null$ **then**
5: $\quad\quad$ Retorne solução não factível
6: $\quad$ **else**
7: $\quad\quad pos_{start} \leftarrow pos_{P_i^+}$
8: $\quad$ **end if**
9: **else if** $P_i^+ \notin H_v$ **then**
10: $\quad$ Retorne solução não factível
11: **end if**
12: **if** isIdle(P_i^-) **then**
13: $\quad pos_{P_i^-} \leftarrow$ determina posição de P_i^- em H_v a partir de pos_{start} com Algoritmo 6
14: $\quad$ **if** $pos_{P_i^-} = null$ **then**
15: $\quad\quad$ Retorne solução não factível
16: $\quad$ **end if**
17: **else**
18: $\quad$ Retorne solução não factível
19: **end if**
20: $cost \leftarrow$ custo de S com i nas posições $pos_{P_i^+}$ e $pos_{P_i^-}$ em R_v
21: Retorne $\{pos_{P_i^+}, pos_{P_i^-}, cost\}$

No Algoritmo 10, se P_i^+ estiver no estado *Atendido* ou em *Transição*, o al-

goritmo verifica se a requisição P_i^+ não pertence à H_v (linha 9), caso verdadeiro, o algoritmo retorna inserção não factível porque H_v não é o veículo responsável pela requisição i (linha 10). Entretanto, se a requisição pertencer a H_v, quer dizer que a entrega já é prevista nesse veículo ($pos_{P_i^-} \in H_v$) e, nesse caso, o algoritmo pode tentar otimizar essa posição. Caso uma posição melhor não seja encontrada, então a posição prevista será mantida. Só é possível ter uma posição prevista de P_i^- dado a invariante mantida pelo Algoritmo 9, que não remove efetivamente P_i^- de H_v se P_i^+ esta no estado *Atendido* ou *Em Transição*. Assim, no pior caso, a posição de $pos_{P_i^-}$ somente não será otimizada.

Continuando o Algoritmo 10, se P_i^- está "Em espera" (linha 12), então $pos_{P_i^-}$ é determinada considerando pos_{start} (linha 13). Se essa inserção não for possível então o algoritmo retorna solução não factível (linha 15). Nesse ponto, se a P_i^+ está no estado *Atendido* ou *Em transição*, então a condição da linha 15 nunca será infactível, isso porque, nesse caso, a invariante mantida pelo Algoritmo 9 mantém um posição de entrega prevista. Entretanto, se P_i^- está no estado "Atendido" ou "Em transição", então significa que a requisição já foi atendida e o algoritmo retorna solução não factível (linha 18). Por fim, o algoritmo calcula o custo adicional com a requisição nas posições calculadas (linha 20) e retorna (linha 21).

Em relação a variável pos_{start}, ela garante que a invariante de precedência seja mantida entre P_i^+ e P_i^-. Caso P_i^+ esteja com status *Atendido* ou *Em Transição*, então pos_{start} será o índice em v onde começam os nós com status *Em Espera*. Ou seja, como $pos_{start} > pos_{P_i^+}$. Entretanto, se P_i^+ está com o status de *Em Espera*, então, após determinado $pos_{P_i^+}$, o valor de pos_{start} será atualizado para esse valor. Assim, a invariante de precedência é mantida pelo algoritmo.

3.3.5 Heurística de inserção

A heurística de inserção difere dos operadores de inserção pela sua simplicidade e velocidade. Enquanto os operadores de inserção buscam avaliar o custo de cada inserção para decidir qual deve ser feita primeira. A heurística de inserção é um procedimento sequencial usado para inserir as requisições na solução de forma rápida. O Algoritmo 11 apresenta a heurística usada no método proposto. Basicamente, ela recebe uma solução S e um conjunto de requisições para serem inseridas R. Para cada requisição $r \in R$ a heurística de inserção irá tentar inserir r no próximo veículo que o Algoritmo 10 retornar inserção factível. Se a heurística de inserção não conseguir inserir r na solução, então um novo veículo é alocado para atender r.

A heurística de inserção é utilizada para gerar a solução inicial e para adicionar as novas requisições que são anunciadas durante a execução (requisições dinâmicas). O custo dessa solução serve como limite superior do processo de otimização, ou seja,

Algoritmo 11 Heurística de inserção.

Require: Solução S, Conjunto R de requisições para inserir
1: **for all** $r \in R$ **do**
2: $v' \leftarrow null$
3: **for all** $v \in S$ **do**
4: **if** É factível inserir r em v executando Algoritmo 10 **then**
5: Atualizar v em S com nova rota gerada pelo Algoritmo 10
6: $v' \leftarrow v$
7: **break**
8: **end if**
9: **end for**
10: **if** $v' = null$ **then**
11: Alocar novo veículo para r
12: **end if**
13: **end for**

a solução gerada por essa heurística pode ser vista como um pior caso. Basicamente, o custo será usado para calcular a temperatura inicial dos algoritmos ALNS (Eq. 2.22).

A heurística de inserção respeita as restrições temporais e dinâmicas do problema apresentadas na Seção 3.1. As propriedades são mantidas pela sub-rotina invocada na linha 3 do Algoritmo 10. Como já apresentado na Seção 3.3.4, o Algoritmo 10 é responsável por respeitar as restrições temporais do problema.

3.3.6 Heurísticas de busca local

Essa seção visa apresentar o uso da busca local no método proposto. O objetivo da busca local é intensificar a otimização em alguns pontos específicos da execução. No método proposto, a busca local é aplicada em três momentos: 1) após a criação da solução inicial, Algoritmo 11; 2) após a inserção de requisições dinâmicas, Algoritmo 11 e 3) sempre que uma nova melhor solução é encontrada pelo $ALNS_{TC}$, Algoritmo 8. Essas execuções podem ser visualizadas na Figura 10 e na Figura 12. No método proposto, são utilizadas as duas heurísticas *relocate* e *exchange* apresentadas na Seção 2.5. Entretanto, essas heurísticas são adaptadas para atender as restrições temporais (Seção 3.1).

O Algoritmo 12 invoca as heurísticas de busca local. Ele controla a ordem e a forma como elas são executadas. O primeiro passo do algoritmo é copiar a solução inicial S em S' e S_{ls} (linhas 2 e 3). Isso serve para manter uma referência da melhor solução encontrada até o momento. Em seguida, o algoritmo executa as heurísticas *relocate* e *exchange* na solução S' enquanto houver melhorias (linhas 4-12). Sempre que uma solução melhor é encontrada (linha 8), a referência de S_{ls} é atualizado com S' para retornar no final.

Algoritmo 12 Algoritmo de execução geral da busca local

Require: Solução S
1: $\delta \leftarrow 1$
2: $S' \leftarrow$ copiar solução S
3: $S_{ls} \leftarrow$ copiar solução S
4: **while** $\delta = 1$ **do**
5: $\quad \delta \leftarrow 0$
6: $\quad S' \leftarrow relocate(S')$
7: $\quad S' \leftarrow exchange(S')$
8: $\quad$ **if** $best(S', S_{ls}) = S'$ **then**
9: $\quad\quad \delta \leftarrow 1$
10: $\quad\quad S_{ls} \leftarrow$ copiar solução S'
11: $\quad$ **end if**
12: **end while**
13: **return** S_{ls}

A heurística *relocate* é apresentada no Algoritmo 13. Como já citado, essa heurística foi adaptada para atender as restrições temporais. No primeiro momento, ela executa enquanto houver melhorias. Esse controle é feito pela variável δ, que mantém a execução enquanto o custo estiver reduzindo ($\delta = 1$). O laço iterativo da linha 5 busca testar cada realocação das requisições de coleta. Na versão proposta, p só é processado quando seu estado está em *Em Espera* (linha 6). Isso garante que a requisição não seja realocada se a coleta já foi feita pelo veículo. Em seguida, as seguintes operações acontecem na seguinte sequência: 1) a solução S_{ls} é copiada para S' (linha 7); 2) é determinado o veículo da requisição p (linha 8); 3) é feita a remoção de p da rota desse veículo (linha 9). Feito isso, tenta-se inserir a requisição p em cada rota da solução S (linha 11-19). Se o custo de uma relocação resultar em uma solução otimizada (linha 13), S_{ls} é atualizado (linha 15). Na sequência, p é removido de H_j. Isso é feito para dar continuidade nos demais testes (linha 17). Um ponto importante desse algoritmo é que o veículo original da requisição p também é testado para otimização. Isso permite tentar uma realocação mais eficiente no próprio veículo.

A heurística *exchange* é apresentada no Algoritmo 14. Similar a heurística *relocate*, essa heurística também considera as restrições temporais (ver Seção 3.1) e executa enquanto houverem melhorias a serem feitas. A ideia da heurística *exchange* é selecionar duas requisições, cada uma de uma rota diferente, e tentar trocar as mesmas de rota (ver Seção 2.5). Nesse algoritmo, as variáveis i, $H_{v'}$, j e $H_{v''}$ representam as requisições e os veículos das requisições, respectivamente. Para melhorar o desempenho, o teste da inserção de i em $H_{v''}$ (linha 15) só é feito se j puder ser inserido em $H_{v'}$ (linha 13). Toda vez que uma troca resulta em uma solução melhor (linha 16), o algoritmo faz a cópia dela (linha 18). Assim como na heurística *relocate*, somente são otimizadas as requisições em que a coleta ainda não foi feita (linhas 7 e 11). Nesse al-

Algoritmo 13 Algoritmo de busca local *relocate*

Require: Solução S
1: $\delta \leftarrow 1$
2: $S_{ls} \leftarrow$ copia de S
3: **while** $\delta = 1$ **do**
4: $\quad \delta \leftarrow 0$
5: $\quad$ **for all** $p \in P^+$ **do**
6: $\quad\quad$ **if** $isIdle(p)$ **then**
7: $\quad\quad\quad S' \leftarrow$ copia de S_{ls}
8: $\quad\quad\quad H_v \leftarrow$ encontrar o veículo v em S' associado a requisição p
9: $\quad\quad\quad H_v \leftarrow H_v \setminus \{P_p^+, P_{p+M}^-\}$
10: $\quad\quad\quad H \leftarrow$ todas rotas de S'
11: $\quad\quad\quad$ **for all** $j \in H$ **do**
12: $\quad\quad\quad\quad$ **if** Factível $p \in H_j$ pelo Algoritmo 10 **then**
13: $\quad\quad\quad\quad\quad$ **if** $best(S', S_{ls}) = S'$ **then**
14: $\quad\quad\quad\quad\quad\quad \delta \leftarrow 1$
15: $\quad\quad\quad\quad\quad\quad S_{ls} \leftarrow$ copia de S'
16: $\quad\quad\quad\quad\quad$ **end if**
17: $\quad\quad\quad\quad\quad H_j \leftarrow H_j \setminus \{P_p^+, P_{p+M}^-\}$
18: $\quad\quad\quad\quad$ **end if**
19: $\quad\quad\quad$ **end for**
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: **end while**
23: **return** S_{ls}

goritmo, uma etapa de recuperação da rota anterior a modificação é aplicada $H_{v'}$ e $H_{v''}$ (linhas 21 e 26), isso visa manter as mesmas rotas para os próximos testes. Devido a troca de requisições entre duas rotas, essa heurística consome mais processamento do que a heurística *relocate*.

3.4 IMPLEMENTAÇÃO

O método proposto foi desenvolvido em duas etapas. A primeira etapa focou em desenvolver uma versão do software para PDPTW, ou seja, a versão estática. O objetivo dessa versão foi validar a solução perante a literatura. A segunda etapa focou em adaptar as funções do software para lidar com o DPDPTW, adicionando as restrições temporais e dinâmicas e o gerenciamento do tempo. A Figura 14 apresenta uma visão geral da arquitetura de componentes do software.

O componente ***Solver*** é a parte central da arquitetura do software proposto. Esse componente processa as novas requisições recebidas pela **Central de Atendimento**, inserindo-as na solução usando a **Heurística de Inserção** e a **Busca Local**. O componente ***Solver*** também se comunica com o componente **Roteador de Veículos**,

Algoritmo 14 Algoritmo de busca local *exchange*

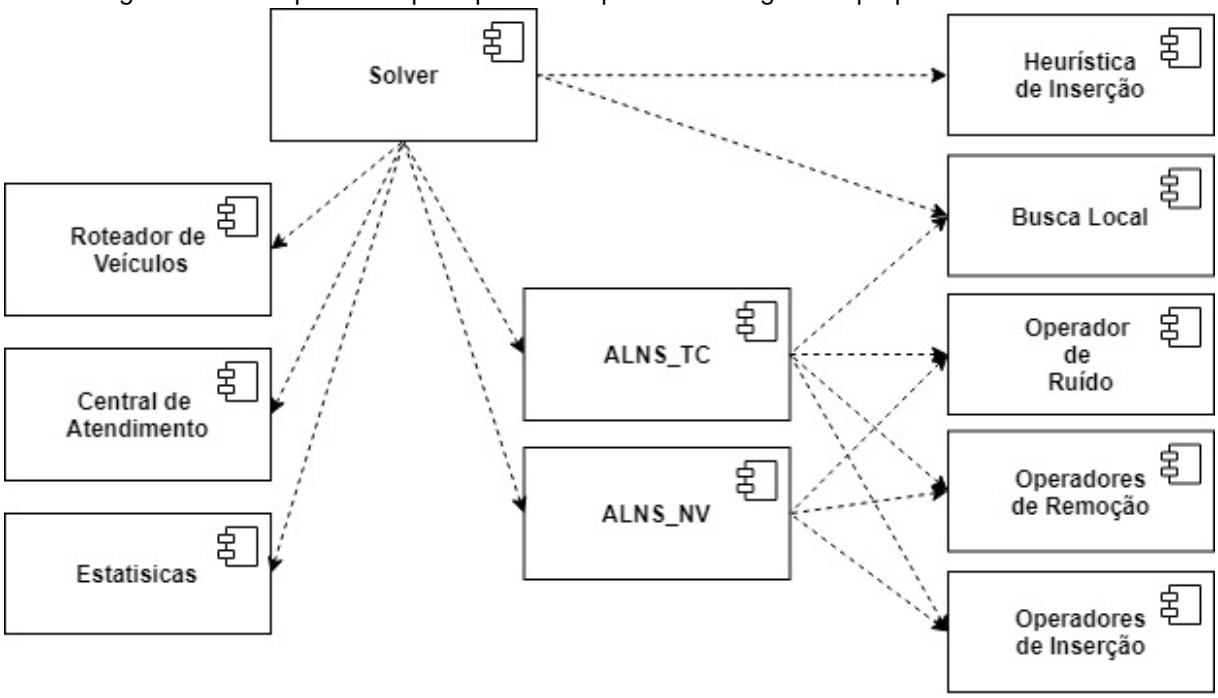
Require: Solução S
1: $\delta \leftarrow 1$
2: $S_{ls} \leftarrow$ copia de S
3: **while** $\delta = 1$ **do**
4: $\quad \delta \leftarrow 0$
5: $\quad S' \leftarrow$ copia de S_{ls}
6: $\quad$ **for all** $i \in P^+$ **do**
7: $\quad\quad$ **if** $isIdle(i)$ **then**
8: $\quad\quad\quad H_{v'} \leftarrow$ encontrar veículo v' em S' associado a requisição i
9: $\quad\quad\quad H_{v'} \leftarrow H_{v'} \setminus \{P_i^+, P_{i+M}^-\}$
10: $\quad\quad\quad$ **for all** $j \in P^+$ **do**
11: $\quad\quad\quad\quad$ **if** $isIdle(j)$ AND $i <> j$ **then**
12: $\quad\quad\quad\quad\quad H_{v''} \leftarrow$ encontrar veículo v'' em S' associado a requisição j
13: $\quad\quad\quad\quad\quad$ **if** $H_{v'} <> H_{v''}$ AND Factível $j \in H_{v'}$ pelo Algoritmo 10 **then**
14: $\quad\quad\quad\quad\quad\quad H_{v''} \leftarrow H_{v''} \setminus \{P_j^+, P_{j+M}^-\}$
15: $\quad\quad\quad\quad\quad\quad$ **if** Factível $i \in H_{v''}$ pelo Algoritmo 10 **then**
16: $\quad\quad\quad\quad\quad\quad\quad$ **if** $best(S', S_{ls}) = S'$ **then**
17: $\quad\quad\quad\quad\quad\quad\quad\quad \delta \leftarrow 1$
18: $\quad\quad\quad\quad\quad\quad\quad\quad S_{ls} \leftarrow$ copia de S'
19: $\quad\quad\quad\quad\quad\quad\quad$ **end if**
20: $\quad\quad\quad\quad\quad\quad$ **end if**
21: $\quad\quad\quad\quad\quad\quad$ Recuperar rota anterior a modificação de $H_{v''}$
22: $\quad\quad\quad\quad\quad\quad H_{v'} \leftarrow H_{v'} \setminus \{P_j^+, P_{j+M}^-\}$
23: $\quad\quad\quad\quad\quad$ **end if**
24: $\quad\quad\quad\quad$ **end if**
25: $\quad\quad\quad$ **end for**
26: $\quad\quad\quad$ Recuperar rota anterior a modificação de $H_{v'}$
27: $\quad\quad$ **end if**
28: $\quad$ **end for**
29: **end while**
30: **return** S_{ls}

essa comunicação serve para manter a execução e a otimização sincronizadas. O **Roteador de Veículos** monitora os veículos nas rotas definidas pelo ***Solver*** e altera os *status* das requisições conforme as visitas realizadas (status *Em Espera*, *Em Transição* e *Atendido*). Quando o ***Solver*** encontra uma rota mais otimizada, o **Roteador de Veículos** é notificado da mesma para atualizar os veículos com os novos caminhos.

O componente **Estatísticas** é utilizado pelo ***Solver*** como repositório de informações coletadas durante a execução. Esse componente armazena informações de custo das soluções geradas, número de iterações, tempo de execução, etc. No final da execução, esse componente sumariza as informações para facilitar as análises estatísticas.

Os componentes **ALNS_TC** e **ALNS_NV**, apresentados na Figura 14, são de-

Figura 14 – Componentes principais da arquitetura do algoritmo proposto ALNS-DPDP.



Fonte: Imagem gerada pelo autor.

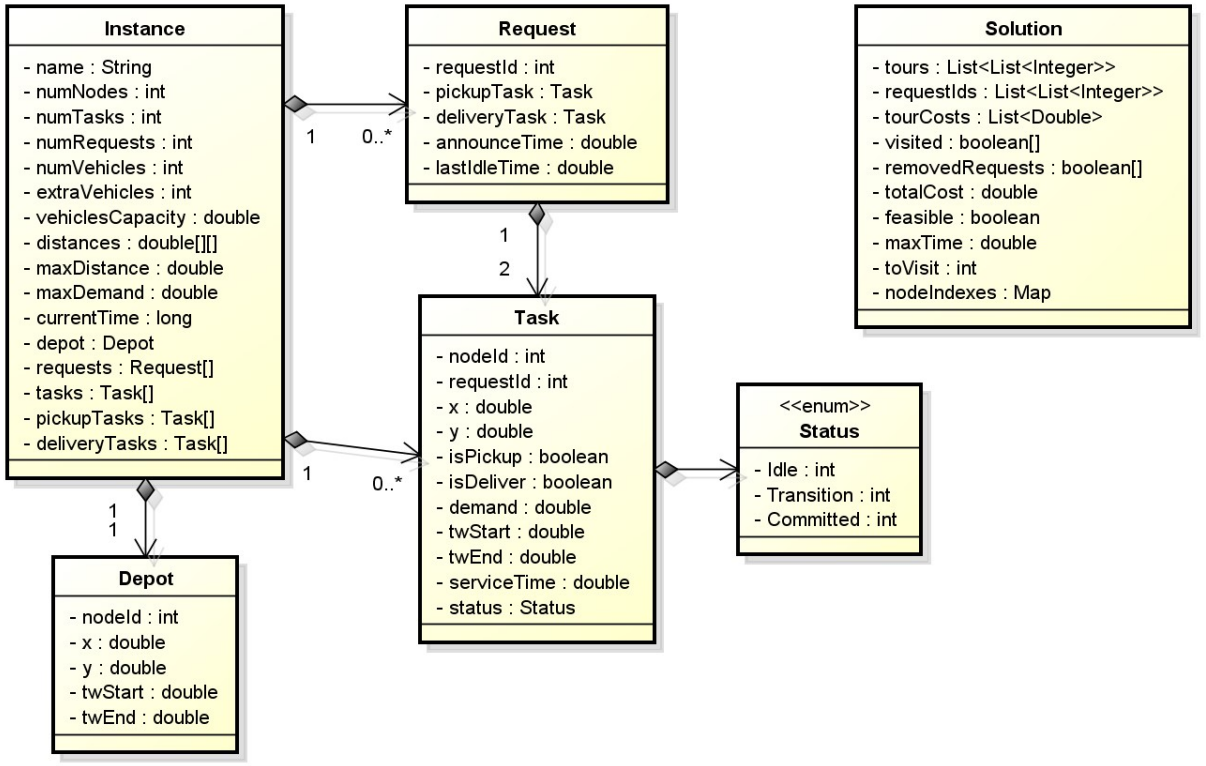
pendências do ***Solver*** e implementam os algoritmos $ALNS_{TC}$ e $ALNS_{NV}$, respectivamente. Na Seção 3.3.2 foi apresentado o Algoritmo 8 como uma implementação única dos dois algoritmos. Entretanto, devido à parametrização de cada objetivo, os componentes **ALNS_TC** e **ALNS_NV** foram executados em instâncias separadas, assim cada componente possuí sua própria parametrização e suas próprias instâncias do **Operador de Ruído**, dos **Operadores de Remoção** e dos **Operadores de Inserção**. Conforme já discutido na Seção 3.3.2, a busca local só é aplicada na otimização do custo total. Dessa forma, somente o **ALNS_TC** possuí uma instância do componente **Busca Local**.

3.4.1 Entidades do domínio

Esta seção apresenta as entidades que modelam as definições do problema e da solução. Essas entidades são usadas por todos os componentes apresentados na Figura 14. A Figura 15 apresenta todas as entidades do ALNS-DPDP. Basicamente, as entidades **Instance**, **Depot**, **Request**, **Task** e **Status** modelam o problema do DPDPTW em termos das restrições, enquanto a entidade **Solution** modela a solução de roteamento.

A entidade **Instance** armazena os dados das requisições e das restrições dos clientes. O atributo *numNodes* armazena o número total de nós. Esse atributo compreende todos os locais da solução (nós de coletas + nós de entregas + nó do depósito). O atributo *numTasks* armazena o número de nós de coletas e entregas, menos o de-

Figura 15 – Principais entidades de domínio do ALNS-DPDP. A entidade *Instance* define uma instância do DPDPTW e a entidade *Solution* representa a solução de roteamento do DPDPTW.



Fonte: Imagem gerada pelo autor.

pósito ($numTasks = numNodes - 1$). Uma **Task** é definida como uma tarefa de coleta ou de entrega que um veículo deve executar. No modelo proposto, cada requisição, entidade **Request**, representa uma solicitação de cliente com uma **Task** de coleta e uma **Task** de entrega. O atributo *numRequests* armazena o número de requisições, como uma requisição é composta por duas tarefas, logo $numRequests = numTasks/2$. Além disso, os atributos: *numVehicles* e *vehiclesCapacity* armazenam as restrições de frota e capacidade dos veículos, o atributo *distances* armazena a matriz de custo do tempo de viagem entre todos os pares de nós do problema, o atributo *currentTime* armazena o valor de $T_{current}$ e o atributo *depot* armazena a referência para a entidade **Depot**. As requisições são armazenadas no atributo *requests*. Os demais atributos *tasks*, *pickupTasks* e *deliveryTasks* são indexadores das informações das requisições armazenadas no atributo *requests*.

A relação entre as entidades **Request** e **Task** é modelada da seguinte forma. Toda requisição feita por um cliente gera um ID de requisição, e para cada ID de requisição são criadas duas instâncias da entidade **Task**, uma responsável pela coleta, onde o atributo *isPickup* será verdadeiro, e outra responsável pela entrega, onde o atributo *isDelivery* será verdadeiro. Cada **Task** também recebe um ID único. As instâncias de **Request** e **Task** são armazenadas na entidade **Instance** via vetores para permitir o acesso em tempo constante via índice do ID. Os atributos *requests*, *tasks*,

pickupTasks e *deliveryTasks* contém as mesmas informações porém organizadas de forma estruturada, isso serve para reduzir a complexidade computacional do software.

Além do atributo *nodeId* e do atributo *requestId*, a entidade **Task** possui as seguintes informações do domínio do problema: coordenadas geográficas, atributos *x* e *y*; identificação se uma requisição é de coleta ou entrega, atributos *isPickup* e *isDelivery*; a informação de demanda do produto, atributo *demand*, que será positivo para coletas e negativo para entregas; os intervalos da janela de tempo, atributos *twStart* e *twEnd*; a informação do tempo de serviço, atributo *serviceTime*, e; por fim, o estado de atentimento da **Task**, atributo *status* representado por um *enum* que contém os valores *Idle*, *Transition* e *Committed*.

No método proposto, toda vez que uma nova requisição dinâmica é anunciada pela central de atendimento, o software cria duas instancias da entidade **Task** e uma instância da entidade **Request**. Em seguida, atualiza-se a entidade **Instance** para acomodar essa nova requisição. Quando uma nova requisição é criada, uma invariante garante que os IDs das entidades tenham continuidade em relação as requisições existentes. Dessa forma, o acesso das informações via IDs continua válido. A entidade **Request** armazena o horário de anunciação na variável *annouceTime* e o último momento válido que a coleta estava com status *Idle* no atributo *lastIdleTime*.

A última entidade de domínio é a entidade **Solution**. Essa entidade armazena os dados da solução usada pelo componente **Roteador de Veículos** (Figura 14). As instâncias da entidade **Solution** são geradas e gerenciadas pelo *Solver* e pelos ALNS. Quando uma solução é criada, ela é composta por uma lista de rotas (atributo *tours*) que são atribuídas aos veículos do depósito. Além disso, as informações de requisição e custo são pareadas em relação as rotas, ou seja, a rota 1 da lista de *tours* está associada às requisições 1 da lista de *requestIds*, que está associada ao custo 1 da lista *toursCosts*. O atributo *totalCost* é calculado com base no somatório dos custos da lista *toursCosts*, conforme a Equação 3.3a. O vetor *visited* armazena as informações relacionadas aos nós visitados e o atributo *removedRequests* indica quais requisições não estão sendo atendidas pela solução. Por fim, o atributo *feasible* indica se a solução é factível, conforme Equação 3.3a.

3.4.2 Tecnologia

O software proposto foi desenvolvido utilizando a linguagem de programação Java, mais especificamente Java 10. Para garantir a qualidade dos software e que os experimentos se mantivessem consistentes ao adicionar mais funcionalidades, foi utilizado o método de desenvolvimento *Test-driven development* (TDD). Isso permitiu avançar mais rápido no desenvolvimento, garantindo o funcionamento das funções que já haviam sido testadas. Durante o desenvolvimento, tomou-se cuidado em usar

tipos de dados primitivos ao invés de seus correspondentes objetos, por exemplo, usou-se vetores de $double[]$ ao invés de $Double[]$, pelo ganho de desempenho obtido por não ter conversões de tipo.

4 EXPERIMENTOS E RESULTADOS

Neste capítulo são apresentados os experimentos executados e os resultados obtidos pelo método proposto denominado de ***Adaptive Large Neighborhood Search for the Dynamic Pickup and Delivery Problem***, ou **ALNS-DPDP**. Na seção 4.1 é apresentado o protocolo de experimentação, a base de dados e os parâmetros de configuração dos experimentos. Em seguida, na seção 4.2 são apresentados os resultados do *benchmark* estático. Na seção 4.3 são apresentados os resultados do *benchmark* dinâmico. Na seção 4.4 é apresentada uma análise do tempo de execução. E por fim, na seção 4.5 são apresentadas as conclusões dos resultados.

4.1 PROTOCOLO DE EXPERIMENTAÇÃO

O protocolo de experimentação focou em analisar a eficiência do ALNS-DPDP aplicado ao problema DPDPTW, porém como não foi encontrado na literatura nenhum *benchmark* dinâmico, adotou-se a seguinte estratégia. No primeiro momento, realizou-se uma análise da eficiência do ALNS-DPDP com um *benchmark* estático da literatura (PDPTW). Essa análise serviu para comparar o ALNS-DPDP em relação aos melhores resultados conhecidos. No segundo momento, realizou-se a análise do ALNS-DPDP com instâncias dinâmicas, adaptadas do *benchmark* estático (DPDPTW). O intuito dessa análise foi avaliar o impacto causado pela dinamicidade no processo de otimização. Por fim, para avaliar a eficiência final do ALNS-DPDP, foram comparados os resultados do *benchmark* dinâmico com os resultados do *benchmark* estático. O objetivo foi entender o impacto causado pela dinamicidade usando os resultados estáticos como base de referência.

Para transformar a base de dados do *benchmark* estático em dinâmico, uma das formas é a aplicação do método proposto por Pankratz (2005). A aplicação desse método tem por objetivo converter instâncias estáticas para dinâmicas considerando dois fatores que impactam na anunciação de requisições dinâmicas, que são: 1) o grau de urgência e 2) o grau de requisições dinâmicas. Para gerar as instâncias do benchmark, diversos graus de variação da dinamicidade foram introduzidos ao *benchmark* estático. Essa variação permitiu avaliar o algoritmo sob diferentes condições de dinamicidade. Além disso, a medida Φ_{edodtw} proposta por Larsen (2000) foi utilizada para analisar o aumento do custo na solução em função do grau da dinamicidade efetiva (ver Seção 2.3.1.3).

O *benchmark* estático utilizado nesses experimentos é o proposto por Li e Lim (2001). Esse *benchmark* tem como função objetivo a minimização de dois objetivos:

1) o número de veículos (NV) e 2) o custo total (TC). O objetivo NV representa o número de veículos com rotas atribuídas e o objetivo TC representa a soma dos custos de tempo das rotas feitas por cada veículo utilizado na solução. Essa função objetivo é aplicada em um modelo com prioridade hierárquica, onde o objetivo NV tem prioridade sob o objetivo TC (ver Equação 3.2). Esse objetivo está em concordância com o método proposto ALNS-DPDP.

O *benchmark* de Li e Lim (2001) compreende variações de diversos fatores, como: tipo de distribuição geográfica dos nós de coleta e entrega, número de requisições para otimização e tamanho do horizonte de tempo do dia de trabalho. Além disso, ele possuí uma base com as melhores soluções encontradas pela literatura, disponível em Sintef (2019). Acerca das melhores soluções, são providas as seguintes informações de cada as instância: a solução com o roteamento de cada veículo, o número de veículos utilizados (NV), o custo total (TC) da solução e uma referência para o trabalho que encontrou a solução (SINTEF, 2019; BENT; HENTENRYCK, 2006; Li; Lim, 2001; ROPKE, 2005; HASLE; KLOSTER, 2007; BALDACCI; BARTOLINI; MINGOZZI, 2011).

Nessa análise, além das variáveis NV e TC, uma variável de factibilidade (FC) foi adicionada para indicar se a solução encontrada não usa mais veículos do que a frota disponível. Quando isso ocorre, a solução é considerada não factível e o valor de $FC = 0$. Essa quebra de factibilidade serve para analisar como o algoritmo se comporta quando os graus de dinamicidade são muito elevados. Além disso, a comparação dos resultados foi feita analisando como essas três métricas são impactadas sob diferentes características do problema.

Ainda, foram avaliadas a temperatura da função de aceitação e o custo da solução local do $ALNS_{NV}$ e do $ALNS_{TC}$ para cada iteração ao longo de toda a execução dos experimentos. Essas medidas foram correlacionadas para verificar a influência delas na qualidade dos resultados da melhor solução, que é mantida pelo *solver*.

Para cada instância de *benchmark* foi feito um experimento com 30 execuções do ALNS-DPDP. O estudo original de (ROPKE; PISINGER, 2006) considerou a experimentação com 10 execuções, assim o uso de 30 execuções aumenta a confiança acerca dos resultados.

Em relação a configuração do ALNS-DPDP, não foi usada nenhuma estratégia de *tunning* para definir melhores parâmetros do $ALNS_{NV}$ e $ALNS_{TC}$. Portanto, os parâmetros utilizados são os mesmos utilizados na experimentação do ALNS aplicado ao DPDPTW em Pisinger e Ropke (2007). Maiores detalhes a cerca dos valores serão apresentados na seção 4.1.3.

Em relação ao hardware utilizado, as execuções foram distribuídas em 3 computadores diferentes. Isso foi necessário dado o massivo número de instâncias de

teste (melhor descrito na Seção 4.1.1). A Tabela 4 apresenta os hardwares utilizados. Pelo fato do software ser desenvolvido em Java para executar sob uma JVM, o código é consistente em diferentes ambientes. Dessa forma, os resultados não foram influenciados pelas diferentes arquiteturas de hardware utilizadas, com exceção do tempo de execução que teve de ser re-avaliado usando um mesmo hardware. No caso do tempo de execução, como são muitas instâncias de teste (ver Seção 4.1.1), foi necessário selecionar apenas um subconjunto de instâncias para realizar as análises.

Tabela 4 – Hardwares utilizados para execução dos experimentos.

ID	**Processador**	**Num. Núcleos**	**Memória**	**Sistema Operacional**	**Java**
PC1	Inter(R) Core(TM) i7-4790 CPU @ 3.60 GHz	8	16.0 GB	Windows 10 Professional Edition x64	Oracle JDK 1.8.0_201
PC2	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz	8	8.0 GB	Windows 10 Home Basic x64	Oracle JDK 10.0.1
PC3	Intel(R) Xeon(R) CPU E7-8860 @ 2.27GHz	80	1.3 TB	Ubuntu 18.04.3 LTS	OpenJDK 11.0.4

4.1.1 Base de dados

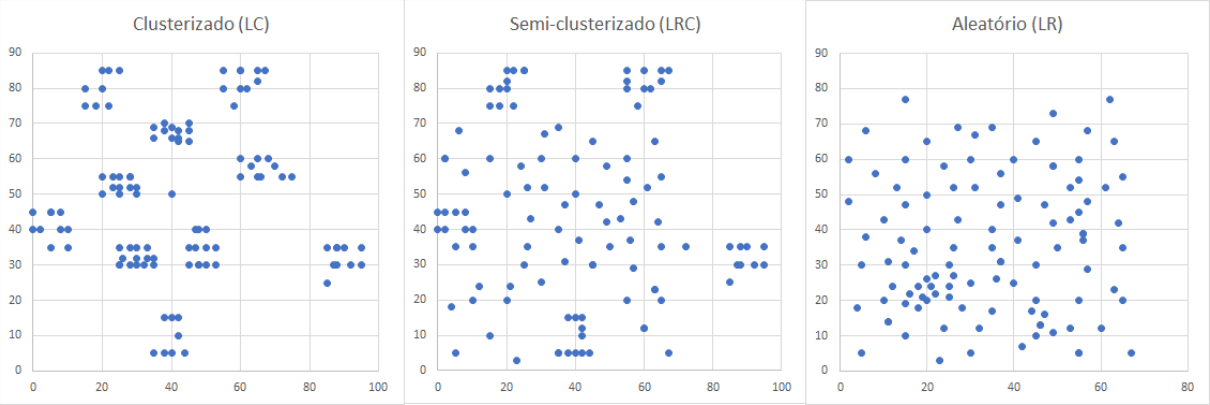
Como já citado anteriormente, os experimentos utilizaram as instâncias de teste do *dataset* do PDPTW proposto por Li e Lim (2001), disponível de forma pública em Sintef (2019). Esse *benchmark* é uma extensão do proposto por Solomon (1987) que incorpora a restrição de coleta e entrega.

O uso dessa base de dados é relevante porque as instâncias são construídas considerando: diferentes tipos de distribuição geográfica dos nós; diferentes números de requisições; variados horizontes de tempo do dia de trabalho e variados tamanhos das janelas de tempo (SOLOMON, 1987). Um detalhe importante é que esse *benchmark* considera frota homogênea, ou seja, todos os veículos da frota possuem a mesma capacidade de carga.

Em Li e Lim (2001) são disponibilizados *benchmarks* com aproximadamente 100, 200, 400, 600, 800 e 1000 nós, que representam 50, 100, 200, 300, 400 e 500 requisições de coleta e entrega, respectivamente. Como pode ser percebido, cada requisição é representada no *benchmark* como um par de nós, um para representar o ponto de coleta e o outro o ponto de entrega da requisição. Ainda, existe um nó adicional que representa o depósito.

Ainda, no *benchmark* de Li e Lim (2001) as instâncias são organizadas em 6 grupos distintos, baseados na distribuição geográfica e no horizonte de tempo do dia de trabalho. Em relação a distribuição geográfica, os prefixos apresentados a seguir determinam o tipo de distribuição das coordenadas dos nós de coleta e entrega. O primeiro conjunto, denominado como *LC*, contém nós com coordenadas agrupadas de forma clusterizada. O segundo conjunto, denominado como *LR*, contém nós com coordenadas distribuídas de forma aleatória e uniformemente variada. Por fim, o terceiro conjunto, denominado como *LRC*, contém nós com coordenadas organizadas de forma *semi-clusterizada* (uma mistura de coordenadas clusterizadas e distribuídas aleatoriamente) (SOLOMON, 1987). A Figura 16 apresenta um exemplo dos diferentes tipos de distribuição geográfica.

Figura 16 – Distribuição geográfica das instâncias lc101, lr101 e lrc101 do *benchmark* PDPTW com 100 nós.



Fonte: Imagem gerada pelo autor

Em relação ao horizonte de tempo do dia de trabalho, essa característica representa o tempo total que os veículos dispõem para atender os clientes em um dia de trabalho (ou também a janela de tempo do depósito). Por exemplo, se os veículos iniciam o dia as 9h e terminam as 18h, então o horizonte de tempo do dia de trabalho é de 9h. No *benchmark* de Li e Lim (2001), os grupos com prefixos $LR1$, $LC1$ e $LRC1$ apresentam um horizonte menor em relação aos grupos com prefixo $LR2$, $LC2$ e $LRC2$. Segundo Solomon (1987), o tamanho do horizonte de tempo e a capacidade de carga dos veículos influenciam diretamente na quantidade de veículos necessários para atender todos os clientes. Portanto, o aumento do horizonte de tempo e da capacidade de carga resultam na diminuição dos veículos necessários para atender a todos os clientes, visto que eles podem fazer rotas maiores.

A Tabela 5 apresenta o número de instâncias agrupadas pelo prefixo do grupo de teste e pelo número de nós. Cada grupo de teste está armazenado em um arquivo *.zip* que contém as instâncias e pode ser baixado em (SINTEF, 2019).

O arquivo de cada instância é formatado conforme as Tabelas 6 e 7. Na pri-

Tabela 5 – Número de instâncias do *benchmark* PDPTW organizadas pelo tipo (tipo de distribuição geográfica e tamanho do horizonte de tempo) e número de nós.

Grupo	**100**	**200**	**400**	**600**	**800**	**1000**	**Total**
LC1	9	10	10	10	10	10	59
LC2	8	10	10	10	10	10	58
LR1	12	10	10	10	10	10	62
LR2	11	10	10	10	10	10	61
LRC1	8	10	10	10	10	10	58
LRC2	8	10	10	10	10	8	56
Total	56	60	60	60	60	58	354

meira linha dos arquivos são definidas as informações da frota de veículos, representado pelas colunas “Número de veículos” e “Capacidade de carga” (Tabela 6). A partir da segunda linha o arquivo contém as informações dos nós, formatadas conforme Tabela 7. Para cada nó é definido o ID do nó (coluna “ID Nó”), as coordenadas geográficas (colunas “Lat. X” e “Long. Y”), as variáveis da janela de tempo (colunas “Inicio Janela Tempo” e “Termino Janela Tempo”), a demanda (Coluna “Demanda”), o tempo de serviço (coluna “Tempo de serviço”) e a relação entre os nós de coleta e entrega (colunas “Índice nó coleta” e “Índice nó entrega”). Especificamente na segunda linha, são definidas as informações do nó de depósito. Nesse caso, o ID do depósito sempre será 0 (por ser o nó de origem) e as janelas de tempo irão definir o horizonte de tempo do dia de trabalho. As demais colunas do depósito são definidas com valor 0 porque não são relevantes. A partir da terceira linha, inicia-se a definição dos nós das requisições e nesse caso todas as colunas contêm informação. Para cada requisição, diferente do depósito, a demanda é definida de forma que o valor do nó de coleta seja positivo e do nó de entrega seja negativo, onde a soma resulte em zero. Para as colunas “Índice nó coleta” e “Índice nó entrega”, a informação é representada de forma a criar uma ligação entre os dois nós de uma mesma requisição. Por exemplo, o nó de coleta de uma dada requisição referencia o nó de entrega da mesma requisição pelo “Índice nó entrega”, e vice-versa.

Tabela 6 – Primeira linha do arquivo das instâncias de teste do *benchmark* PDPTW. K representa o número de veículos disponível na frota e Q a capacidade de carga dos veículos.

Número de veículos	Capacidade de carga
K	Q

Durante o carregamento do arquivo a distância entre cada par de nós precisa ser calculada. Isso é feito por meio da distância Euclidiana, definida na Equação 4.1, onde i e j representam os ID's dos nós, e x e y representam suas coordenadas.

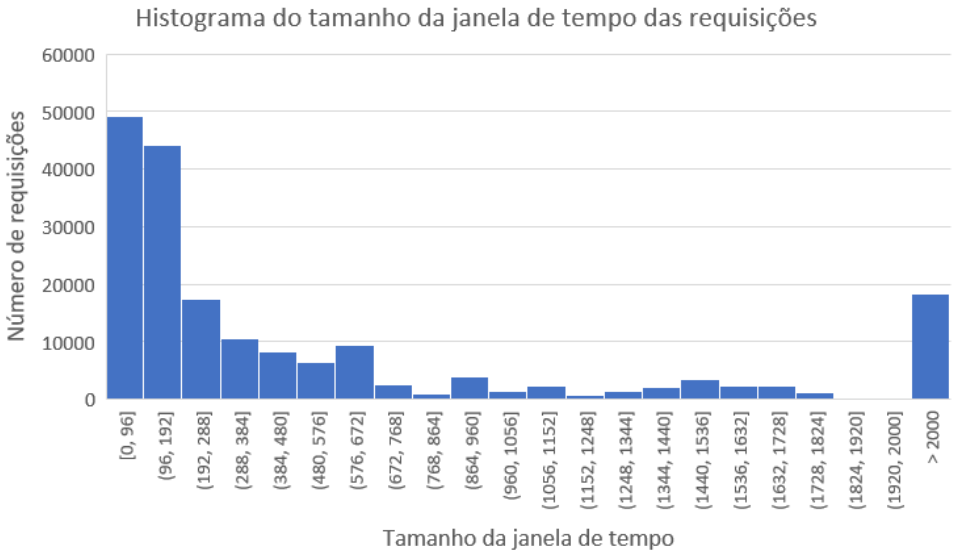
$$dist(i,j) = \sqrt{(G_i^x - G_j^x)^2 + (G_i^y - G_j^y)^2} \tag{4.1}$$

Tabela 7 – Restante das linhas do arquivo das instâncias de teste do *benchmark* PDPTW.

ID Nó	Lat. X	Long. Y	Demanda	Inicio Janela Tempo	Término Janela Tempo	Tempo de serviço	Índice nó coleta	Índice nó entrega
0	G_0^x	G_0^y	q_0	e_0	l_0	s_0	i_0	j_0
1	G_1^x	G_1^y	q_1	e_1	l_1	s_1	i_1	j_1
...	...	...	...	...	...	...	...	...

Uma análise sobre as janelas de tempo das instâncias do *benchmark* permitiu entender melhor como as tarefas de coleta e entrega estão organizadas. A Figura 17 apresenta um histograma da diferença entre o tempo de início e fim das janelas de atendimento de todas as coletas e entregas das instâncias do *benchmark*. Pode-se observar que a grande maioria dos nós possui janela entre 5 e 300 unidades de tempo, ou seja, janelas de tempo curtas. A importância da janela de tempo está relacionada à flexibilidade no re-arranjo das rotas, porque quanto menor a janela de tempo menor é o número de combinações possíveis. Em problemas com requisições dinâmicas, as janelas de tempo curtas aumentam a complexidade da otimização já que as soluções são menos flexíveis a alterações.

Figura 17 – Histograma dos tamanhos das janelas de tempo de todas as instâncias do *benchmark* PDPTW



Fonte: Imagem gerada pelo autor

4.1.2 Detalhamento dos cenários dinâmicos

Como o *benchmark* de Li e Lim (2001) é composto somente por instâncias estáticas, foi necessário aplicar o método de Pankratz (2005), descrito na Seção 2.3.1.4, para gerar o *benchmark* dinâmico. Com base nesse método, inicialmente foi calculado o tempo limite de anunciação, chamado de T^{latest}, para cada requisição contida no

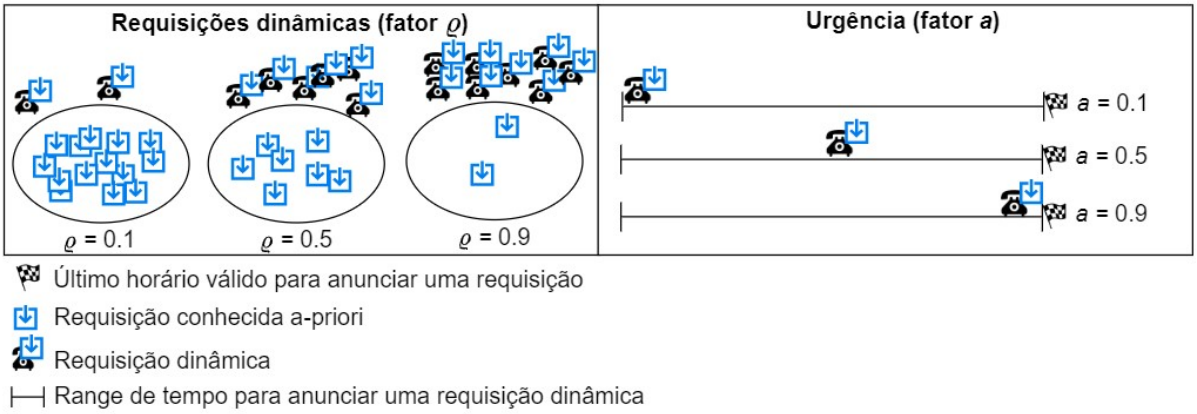
benchmark de Li e Lim (2001). Em seguida, o *benchmark* dinâmico teve suas instâncias organizadas levando em consideração os seguintes fatores: 1) grau de urgência (fator a) e 2) grau de requisições conhecidas *a priori* (fator φ). Para simplificar as análises, neste trabalho o grau de requisições conhecidas *a priori* será analisado como o grau de requisições conhecidas *a posteriori*, ou seja, requisições dinâmicas (definida pela fórmula $\varrho = 1 - \varphi$).

A aplicação do método de conversão de Pankratz (2005) exigiu adaptar o formato da Tabela 7 para comportar uma nova informação. Nesse novo formato foi adicionada a coluna "Tempo de anúncio" para armazenar a variável A_i. Essa variável define o tempo de anunciação de uma requisição dinâmica durante a execução do *benchmark*. No caso de uma requisição estática, o valor de A_i sempre é definido como zero. Já para uma requisição dinâmica, o valor de A_i é definido conforme o método de Pankratz (2005). Esse método garante que A_i não ultrapasse o tempo limite de anunciação (calculado pela a Equação 2.6), ou seja, $A_i \leq T_i^{latest}$. A definição do valor de A_i para cada *benchmark* varia conforme o grau de urgência desejado (Eq. 2.7). Quando o grau de urgência, fator a, for menor (mais próximo de zero) as requisições serão anunciadas mais próximas ao tempo de início da operação (T_{start}) e mais distantes do último horário válido (T_i^{latest}). Consequentemente o algoritmo de otimização terá mais flexibilidade para arranjar a requisição na solução visto que haverá um intervalo de tempo maior, ou seja, a otimização do número de veículos e do custo é mais fácil de ser feita.

A Figura 18 ilustra como os dois fatores de dinamicidade, fator a e fator ϱ, influenciam na geração das instâncias dinâmicas. Na esquerda pode-se observar a variação do fator ϱ (grau de requisições dinâmicas), de forma que o aumento de ϱ implica no aumento do número de requisições dinâmicas. Enquanto que na direita pode-se observar a variação do fator a (grau de urgência), de forma que o aumento do fator a implica no aumento da urgência da requisição. Uma consideração importante a ser destacada é que quando $a = 0$ todas as requisições são conhecidas no início do dia de trabalho ($A_i = 0$), por outro lado, quando $a = 1$ as novas requisição são anunciadas no último momento válido ($A_i = T_i^{latest}$). No caso do grau de urgência, quanto maior for o a menor será o tempo de reação, o que implica em menos tempo para otimizar a solução.

Um último fator aplicado às instâncias de *benchmark* é o tempo de preparação (*setup time*). Esse tempo visa definir um valor mínimo de tempo necessário para alocar uma requisição a um veículo. No *benchmark* dinâmico, esse tempo foi definido de forma empírica como 90 unidades de tempo. Dessa forma, durante a execução do *benchmark* para uma dada requisição R_i, quando a inequação $T_{current} \geq A_i - 90$ for atendida, então a requisição é anunciada para o *benchmark*. Lembrando que $T_{current}$

Figura 18 – Diagrama ilustrando diferentes características dos grau de dinamicidade. Na esquerda variação do grau de requisições dinâmicas e na direita variação no grau de urgência.



Fonte: Imagem criada pelo autor.

é a variável que acompanha o passar do tempo durante a execução do problema com o algoritmo.

De forma a organizar os fatores a e ϱ, o método proposto por Pankratz (2005) define dois conjuntos de testes, P1 e P2. O conjunto P1 foca no grau de urgência e o conjunto P2 foca no grau de requisições dinâmicas. No conjunto P1, assume-se que todas as requisições são dinâmicas, $\varrho = 1.0$, enquanto o fator a é variado no intervalo $[0, 1]$. Ou seja, o valor de A_i é calculado para todas as requisições. No conjunto P2, todas as requisições são anunciadas com urgência máxima, fator $a = 1.0$, enquanto varia-se o número de requisições dinâmicas, fator ϱ no intervalo $[0, 1]$. O valor de ϱ define a porcentagem de requisições que terá o valor de $A_i > 0$. O processo para selecionar quais requisições serão dinâmicas é feito por meio de uma seleção aleatória até que a porcentagem de ϱ seja atendida.

A partir do *benchmark* estático foram gerados cinco conjuntos de teste para P1 e cinco conjuntos de teste para P2. A geração de ambos foi feita da mesma forma usando o método de Pankratz (2005), a diferença reside nos graus dos fatores a e ϱ. Os conjuntos de P1 foram gerados variando-se a urgência em baixa, média-baixa, média, média-alta e alta, $a =$ [0.1, 0.25, 0.5, 0.75, 1.0], respectivamente, mantendo-se $\varrho = 1.0$. Enquanto os conjuntos de P2 foram gerados variando-se o grau de requisições dinâmicas em baixa, média-baixa, média, média-alta e alta, $\varrho =$ [0.1, 0.25, 0.5, 0.75, 1.0], respectivamente, mantendo-se $a = 1.0$.

Os problemas de requisições dinâmicas também consideram a condição do veículo em movimento, aqui nomeada como MV (*moving vehicle*). Essa condição ativa a visita dos clientes pelos veículos ao longo da execução do problema conforme definido na rota da melhor solução. Para atender essa situação, as requisições vão mudando de estado com o passar do tempo (ver Seção 3.1). Essa condição impõem

restrições a otimização *online* no que diz respeito as alterações na rota, pois quando um veículo visita um cliente a rota do depósito até aquele cliente não pode mais ser alterada para impedir a geração de inconsistências.

Para nomear as instâncias de testes, foram concatenados os nomes das respectivas instâncias estáticas com um sufixo que indica o tipo de dinamicidade. Para o grau de urgência foi usado o sufixo a_x e para o grau de requisições dinâmicas foi usado o sufixo ϱ_x, onde x representa a variação do respectivo fator. Na Tabela 8 é apresentado um exemplo de nomenclatura, juntamente com o grau de complexidade para resolver aquela instância. A coluna “Grau de complexidade” compreende o nível de dificuldade em relação ao veículo em movimento e a entrada de requisições dinâmicas.

Tabela 8 – Exemplo da nomenclatura para as instâncias dinâmicas de *lc101*. O grau de complexidade é baseado na aplicação do veículo em movimento (MV) e no grau de dinamicidade.

Nome da instância	Grau de urgência (fator a)	Grau de requisições dinâmicas (fator ϱ)	Nível de complexidade
`lc101`	0.0	0.0	sem MV baixo
`lc101_a_0.1`	0.1	1.0	com MV baixo
`lc101_a_0.25`	0.25	1.0	com MV baixo-médio
`lc101_a_0.5`	0.5	1.0	com MV médio
`lc101_a_0.75`	0.75	1.0	com MV médio-alto
`lc101_a_1.0`	1.0	1.0	com MV alto
`lc101_`ϱ`_0.9`	1.0	0.9	com MV médio-alto
`lc101_`ϱ`_0.75`	1.0	0.75	com MV médio
`lc101_`ϱ`_0.5`	1.0	0.5	com MV médio-baixo
`lc101_`ϱ`_0.25`	1.0	0.25	com MV baixo
`lc101_`ϱ`_0.0`	1.0	0.0	com MV baixo

Para cada instância do *benchmark* estático, foram realizadas experimentações para as respectivas 10 instâncias do *benchmark* dinâmico e para a instância do *benchmark* estático. A Tabela 9 apresenta a distribuição das instâncias por tipo e número de nós. No total, somando as instâncias dinâmicas e estáticas, foram usadas $354 + 10 \times 354 = 3.894$ instâncias de teste. Para a análise de cada instância foram feitas 30 execuções isoladas. Esse número elevado de execuções aumenta a confiabilidade dos resultados. Considerando a quantidade de instâncias e a quantidade de execuções realizadas, foram realizadas 106.200 execuções do algoritmo (3.894 instâncias $\times$ 30 execuções).

4.1.3 Definição de parâmetros

A definição dos parâmetros do ALNS foi feita com base na versão original do algoritmo proposto por Pisinger e Ropke (2007). A Tabela 10 apresenta os valores jun-

Tabela 9 – Distribuição do número de instâncias de testes do *benchmark* de Li e Lim (2001) geradas a partir do método de (PANKRATZ, 2005).

Tipo	100 nós	200 nós	400 nós	600 nós	800 nós	1000 nós	Total
LC1	90	100	100	100	100	100	590
LC2	80	100	100	100	100	100	580
LR1	120	100	100	100	100	100	620
LR2	110	100	100	100	100	100	610
LRC1	80	100	100	100	100	100	580
LRC2	80	100	100	100	100	80	560
Total	560	600	600	600	600	580	3.540

tamente com uma breve descrição de sua utilização. Como o método proposto utiliza dois algoritmos ALNS integrados ao *solver*, $ALNS_{NV}$ e $ALNS_{TC}$, alguns parâmetros foram duplicados.

4.1.4 Metodologia de execução dos experimentos

Dado ao massivo número de instâncias, inicialmente foi feito uma estimativa do tempo necessário para executar todos os experimentos. Para isso, foi executado o algoritmo com o *benchmark* estático e então extrapolado o tempo de execução para o *benchmark* dinâmico. A Tabela 11 apresenta os resultados de tempo obtidos em cada experimento. Nessa tabela, a extrapolação é feita multiplicando por 10, visto que são cinco varições do grau de urgência e cinco variações do grau de requisições dinâmicas para cada instância de teste.

Com base nos resultados da Tabela 11, foi necessário dividir a execução dos experimentos nos hardwares descritos na Tabela 4. Essa distribuição, na prática, permitiu reduzir o tempo total da experimentação para aproximadamente 3 semanas. A Tabela 12 apresenta de forma detalhada como a experimentação foi distribuída.

Para tirar proveito da distribuição, a implementação do algoritmo também considerou a execução paralela dos experimentos usando de forma eficiente as CPUs do computador. Por exemplo, no PC1, que dispõem de 8 cores, o algoritmo foi parametrizado para fazer 6 execuções paralelas, deixando disponível uma CPU para o sistema operacional e uma CPU para controlar as 6 execuções. Como cada experimento é composto de 30 execuções, no PC1 e PC2, que dispõem de 8 CPUs, a execução de um *benchmark* foi quebrada em 5 execuções sequenciais de 6 threads. Já no PC3, que dispõem de 80 cores, foi possível executar dois experimentos paralelos usando 60 threads do sistema (30 threads cada experimentos).

O algoritmo foi desenvolvido para permitir a parametrização via argumentos, que descrevem como o algoritmo irá executar os experimentos. A Tabela 13 apresenta os principais parâmetros. A inicialização do algoritmo ocorre da seguinte forma. A pri-

Tabela 10 – Descrição e valores dos parâmetros dos algoritmos $ALNS_{NV}$ e $ALNS_{TC}$

Parâmetro	$ALNS_{TC}$	$ALNS_{NV}$	Descrição
ε	0.05	0.25	Tolerância do custo da solução inicial. Usado pelo cálculo da temperatura inicial.
γ	0.99975	0.9999	Taxa de resfriamento da temperatura do SA.
k	100		Número de iterações (segmento) usado na atualização dos pesos.
χ	9		Operador de remoção *shaw*, peso da distância entre nós.
χ'	3		Operador de remoção *shaw*, peso dos tempos de chegada entre nós.
χ''	2		Operador de remoção *shaw*, peso das janelas de término entre nós.
p_{shaw}	6		Grau de randomização do operador de remoção *shaw*.
p_{worst}	3		Grau de randomização do operador de remoção *worst*.
θ_1	33		Pontuação dos operadores quando uma melhor solução é encontrada.
θ_2	9		Pontuação dos operadores quando uma solução local é encontrada.
θ_3	13		Pontuação dos operadores quando uma solução de pior custo é aceita.
ρ	0.1		Fator de reação, fator de controle do histórico no ajuste dos pesos.
ζ	0.025		Nível de ruído, determina o fator de variação usado pelos operadores de inserção.
ξ	0.4		Controle de remoção, define um número máximo de requisições que podem ser removidas.
i_{max}	25000		Número total de iterações do algoritmo *solver*.
ϑ	100000		Parâmetro de penalização por requisição não atendida.

meira ação é acessar o repositório de entrada (parâmetro *-inputDir*) e carregar as instâncias de teste. Em seguida, para cada instância de teste é criado um novo experimento. O experimento é configurado com base no número de execuções (parâmetro *-numTrials*) e no número de CPUs disponíveis (parâmetro *-numCpus*). Com base nisso, calcula-se a distribuição das execuções por meio da relação $numTrials/numCpus$, ou seja, se existirem mais execuções do que CPUs disponíveis, então serão criados grupos de execuções menores que serão executados sequencialmente. Então, o algoritmo inicia a execução do experimento configurando o Solver com os demais

Tabela 11 – Análise preliminar do algoritmo com as instâncias estáticas (PDPTW) para calcular a distribuição e o tempo necessário para executar todos experimentos. Resultados obtidos usando o PC3 da Tabela 4.

Instância	Tempo	Extrapolação instâncias dinâmicas ($\times 10$)
pdptw 100	2h	20h
pdptw 200	8h	80h
pdptw 400	27h	270h
pdptw 600	28h	280h
pdptw 800	49h	490h
pdptw 1000	52h	520h
Total	166h	1660h

Tabela 12 – Separação dos benchmarks por hardwares para execução dos experimentos. A separação foi feita com base no número de nós (ex: “pdptw 200”) e em alguns casos pelo tipo (ex: “dpdptw 1000 (LC1)”).

Hardware	Benchmark	Tipo
PC1	pdptw 200	Estático
	pdptw 400	Estático
	pdptw 1000	Estático
	dpdptw 1000 (LC1)	Dinâmico
	dpdptw 1000 (LRC2)	Dinâmico
	dpdptw 1000 (LR1)	Dinâmico
PC2	pdptw 600	Estático
	pdptw 800	Estático
	dpdptw 200	Dinâmico
PC3	pdptw 100	Estático
	dpdptw 100	Dinâmico
	dpdptw 400	Dinâmico
	dpdptw 600	Dinâmico
	dpdptw 800	Dinâmico
	dpdptw 1000 (LC2)	Dinâmico
	dpdptw 1000 (LRC1)	Dinâmico
	dpdptw 1000 (LR2)	Dinâmico

parâmetros (parâmetros *-localSearch*, *-maxIterations* e *-movingVehicle*). No final do experimento, o resultado de todas as execuções é agrupado e salvo no diretório de saída (parâmetro *-outputDir*).

Os experimentos para as instâncias estáticas e dinâmicas foram parametrizados de forma diferente para ativar o veículo em movimento (MV). Esse parâmetro só é usado nas instâncias dinâmicas porque estas representam um cenário de otimização *online* integrado aos veículos em movimento. No caso da anunciação das requisições dinâmicas, não foi necessário um parâmetro adicional para diferenciar um arquivo de instância estática de um arquivo de instância dinâmica, porque a diferença entre os ar-

Tabela 13 – Parâmetros de configuração para execução do algoritmo proposto ALNS-DPDP.

Parâmetro	Função
-numCpu	Define o número CPUs para paralelização
-numTrials	Define o número de execuções para cada instância
-maxIterations	Define o número de iterações do solver
-movingVehicle	Ativa o simulador do veículo em movimento
-localSearch	Ativa os algoritmos de busca local
-inputDir	Define o diretório das instâncias de *benchmark*
-outputDir	Define o diretório para geração dos resultados

quivos é a coluna do tempo de anunciação (A_i) que pode ser ignorada se não existir. A listagem abaixo apresenta as linhas de comando usadas em um *benchmark* estático e um *benchmark* dinâmico de 100 nós.

- *Benchmark* estático: *java -Xmx100g -Xms10g -jar alns-dpdptw.jar -numCpu 30 -numTrials 30 -maxIterations 25000 -localSearch -inputDir /static-input/pdp-100/ -outputDir /static-output/pdp-100/*
- *Benchmark* dinâmico: *java -Xmx100g -Xms10g -jar alns-dpdptw.jar -numCpu 30 -numTrials 30 -maxIterations 25000* ***-movingVehicle*** *-localSearch -inputDir /dynamic-input/pdp-100/ -outputDir /dynamic-output/pdp-100/*

Por fim, a última parte do algoritmo é a geração dos relatórios dos experimentos. Para cada experimento realizado o algoritmo gera três arquivos de resultados:

- Melhor solução final de um experimento (*<nome da instancia>_bsf.csv*): Contém o custo total, o número de veículos, a factibilidade da solução e o tempo de execução da melhor solução final das 30 execuções de um experimento.
- Média das soluções finais do experimento (*<nome da instancia>_summary.csv*): Contém a média e desvio padrão do custo total, do número de veículos, da factibilidade da solução e do tempo de execução das 30 execuções de um experimento.
- Variáveis de cada iteração do experimento (*<nome da instancia>_iteration.csv*): Contém a média e desvio padrão do custo total, do número de veículos e da factibilidade da melhor solução do Solver. Adicionalmente, contém a média e desvio padrão da solução local e da temperatura dos algoritmos $ALNS_{TC}$ e $ALNS_{NV}$. Onde essas variáveis são a média das 30 execuções coletadas a cada iteração do algoritmo.

A seguir será apresentada a análise dos resultados obtidos. Lembrando que essa análise foi conduzida em duas frentes, uma dedicada à avaliação dos resultados

do *benchmark* estático, e outra à avaliação dos resultados do *benchmark* dinâmico. Ambas as avaliações foram comparadas com os resultados da literatura. A análise do *benchmark* estático teve por intuito avaliar se o ALNS-DPDP alcançou resultados próximos em relação ao melhor encontrado na literatura. E a análise do *benchmark* dinâmico teve por intuito avaliar se o ALNS-DPDP trouxe contribuição no tratamento da dinamicidade no DPDPTW. Nessas análises, foram comparados os resultados absolutos e os relativos. Os resultados absolutos se referem aos valores reais obtidos pelo algoritmo, enquanto os resultados relativos são porcentagens que indicam o quão melhor ou pior os valores obtidos foram em relação a um valor de referência da literatura.

4.2 RESULTADOS DO ALNS-DPDP APLICADO AO *BENCHMARK* ESTÁTICO

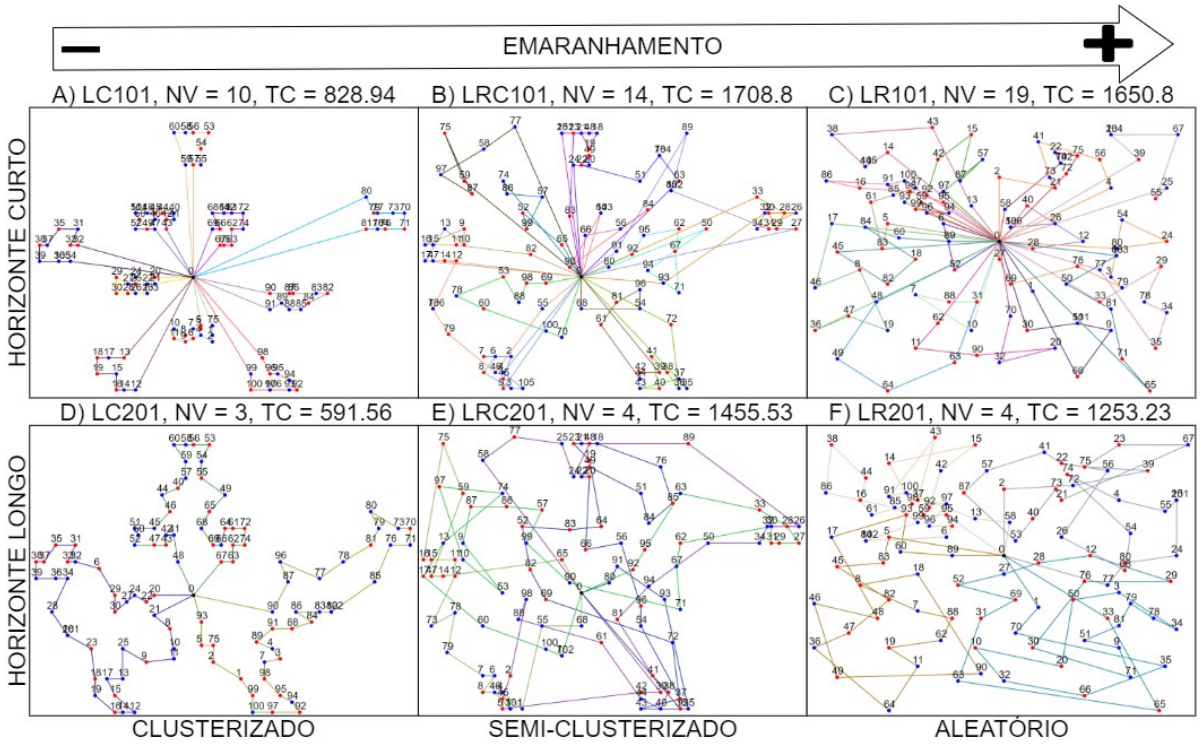
A análise dos resultados estáticos é necessária para comparar a eficiência do método proposto perante a literatura. Para isso, inicialmente serão comparados os resultados obtidos do problema em relação a literatura, e em seguida, para entender o comportamento do ALNS-DPDP, será feita uma análise interna do algoritmo.

Antes de iniciar as análises é importante contextualizar o que caracteriza uma solução do problema, para isso a Figura 19 apresenta graficamente as soluções finais dos problemas lc101, lrc101, lr101, lc201, lrc201 e lr201 encontradas pelo ALNS-DPDP. Essas instâncias selecionadas permitem visualizar as diferentes características do problema: distribuição geográfica dos nós e tamanho do horizonte de tempo. Na Figura 19, cada cor de linha representa a rota de um veículo, cada número representa o ID de um nó, os nós em vermelho representam pontos de coleta, os nós em azul os pontos de entrega, NV o número de veículos da solução e TC o custo total da solução. Analisando o aumento da dispersão dos nós que ocorre na ordem "clusterizado < semi-clusterizado < aleatório", percebe-se um aumento do emaranhamento das rotas, ou seja, as rotas dos veículos se cruzam mais frequentemente. Além disso, analisando o horizonte de tempo longo em relação ao horizonte de tempo curto, percebe-se uma redução no número de veículos e um aumento no tamanho das rotas. Isso ocorre dado ao maior tempo disponível para atender os clientes.

4.2.0.1 Analise dos resultados absolutos

Os resultados absolutos do *benchmark* estático são apresentados na Tabela 14. Essa tabela é composta pelas melhores soluções da literatura, e também pela média e melhor solução das instâncias do *benchmark* estático. A coluna literatura apresenta as melhores soluções encontradas e registradas no site Sintef (2019). A coluna ALNS-DPDP apresenta a média, o desvio padrão e a melhor solução para os resultados do *benchmark* estático. O valor da melhor solução encontrada está destacado entre parênteses. Como existem muitas instâncias foi necessário sumarizar os resulta-

Figura 19 – Exemplos de soluções estáticas para os problemas com 100 nós lc101, lrc101 e lr101. As instâncias estão organizadas pela distribuição geográfica para analisar que uma distribuição mais esparsa dos nós leva a um maior emaranhamento das rotas



Fonte: Imagem criada pelo autor.

dos para melhorar a visualização, ou seja, são apresentadas aqui a média dos valores agrupados pelo número de nós e pela distribuição geográfica. Para essa análise foi utilizada a média e o desvio padrão das soluções encontradas, pois o método proposto utiliza uma heurística, o que confere a ele um certo grau de não determinismo, portanto, essas medidas se tornam mais adequadas para fazer essa análise.

Com base na média geral apresentada na Tabela 14, pode-se observar que o número de veículos dos resultados obtidos é maior do que a literatura, 24.4 versus 23.72, enquanto que o custo total é menor, 14452.81 versus 14912.13. Isso indica uma relação inversa entre o número de veículos e custo total. Essa relação é um efeito do objetivo hierárquico do problema, onde o reaproveitamento dos veículos (reduzir NV) tende a fazer com os veículos andem maiores distâncias (cruzando o mapa) para atender mais clientes (aumentar TC). Isso é corroborado ao analisar os resultados em função da distribuição geográfica dos nós. Por exemplo, na Tabela 14, para as instâncias com 1000 nós, a distribuição clusterizada tem 61.15 veículos a um custo total de 30320.96, a distribuição semi-clusterizada tem 40.39 veículos a um custo total de 37527.85 e a distribuição aleatória tem 31 veículos a um custo total de 46338.86.

Tabela 14 – Comparação das soluções da literatura versus valores médios, desvio padrão ($\mu \pm \sigma$) e melhor solução (em parênteses) das instâncias do benchmark estático obtidos pelo ALNS-DPDP. Os resultados foram compactados pelo número de nós e pelo tipo de distribuição geográfica.

Nós	Tipo	Número de veículos (NV)		Custo total (TC)	
		Lit.	ALNS-DPDP	Lit.	ALNS-DPDP
100	LC	6.53	6.53±0.00 (6.53)	740.35	741.52±1.53 (740.35)
	LRC	7.38	7.38±0.00 (7.38)	1259.93	1265.21±6.45 (1260.82)
	LR	7.52	7.53±0.02 (7.52)	1100.64	1100.70±0.88 (1100.64)
200	LC	12.35	12.40±0.02 (12.35)	2346.72	2321.16±14.01 (2346.81)
	LRC	9.20	9.32±0.08 (9.20)	3063.67	3097.81±55.48 (3081.70)
	LR	8.45	8.68±0.04 (8.65)	3779.18	3640.73±56.40 (3619.32)
400	LC	24.40	24.58±0.04 (24.50)	5786.48	5717.72±43.89 (5755.49)
	LRC	17.70	18.48±0.23 (18.10)	7299.18	7027.67±207.54 (7124.03)
	LR	14.40	15.05±0.12 (14.85)	8888.55	8486.85±196.65 (8410.41)
600	LC	36.65	37.20±0.19 (36.90)	11594.08	11331.14±281.28 (11492.21)
	LRC	23.95	25.14±0.27 (24.80)	14805.12	14063.29±375.77 (14126.12)
	LR	20.45	21.09±0.21 (20.90)	18441.77	18122.71±574.28 (17779.40)
800	LC	48.75	49.33±0.14 (49.05)	19359.26	18967.34±321.56 (19029.82)
	LRC	31.50	33.29±0.45 (32.60)	24537.12	23228.74±564.64 (23866.94)
	LR	25.20	26.20±0.31 (25.65)	31228.67	30057.21±915.07 (30407.24)
1000	LC	61.15	61.85±0.18 (61.50)	30320.96	30195.13±417.19 (30335.99)
	LRC	40.39	42.84±0.47 (41.94)	37527.85	36195.53±666.29 (36884.95)
	LR	31.00	32.42±0.38 (31.95)	46338.86	44590.18±1538.89 (44518.10)
Total		23.72	24.40±0.17 (24.13)	14912.13	14452.81±346.54 (14548.90)

4.2.0.2 Análise comparativa de desempenho com a literatura (GAP)

Como o objetivo da análise inicial é avaliar o ALNS-DPDP para o problema estático, além da análise dos valores absolutos (Seção 4.2.0.1), também é necessário quantificar o desempenho relativo perante a literatura. Para isso, no primeiro momento a análise será feita sob o ponto de vista da melhor solução encontrada e, em seguida, será feita em termos da média e do desvio padrão. A análise da melhor solução é pertinente para entender o quão especializado o ALNS-DPDP está a identificar o que a literatura considera a melhor solução existente. Enquanto que a análise da média e do desvio padrão indicam quão robusto o ALNS-DPDP está a identificar recorrentemente resultados adequados, apesar da aleatoriedade intrínseca da heurística utilizada.

Na análise da melhor solução encontrada foi aplicado o desvio do custo do melhor resultado do ALNS-DPDP em relação ao custo da literatura, ou seja, o GAP entre as soluções. A Equação 4.2 apresenta o cálculo do GAP, onde N representa um conjunto de instâncias, $i \in N$ representa cada instância, L_i é o custo da literatura da instância i e B_i^{bsf} é o custo da melhor solução encontrada pelo ALNS-DPDP da instância i. Basicamente, essa equação calcula a porcentagem média de desvio que

o método proposto teve em relação a literatura. Nesse contexto, um GAP de zero indica que o método encontrou a mesma solução da literatura, um GAP positivo indica que encontrou um custo maior e um GAP negativo indica que encontrou um custo menor. É importante destacar que esse cálculo é realizado separadamente para cada um dos dois objetivos NV e TC, chamados $GAP_{NV}(N)$ e $GAP_{TC}(N)$, respectivamente.

$$GAP(N) = \frac{\sum_{i \in N} B_i^{bsf}}{\sum_{i \in N} L_i} - 1 \tag{4.2}$$

A análise do GAP dos melhores resultados obtidos é apresentada na Tabela 15. A coluna “Número de nós” agrupa os resultados pelo tamanho do problema, a coluna “NV” apresenta o GAP médio do número de veículos e a coluna “TC” apresenta o GAP médio do custo total. Nessa tabela os resultados foram agrupados para analisar se existe alguma variação do GAP em função do tamanho do problema. Analisando os resultados, percebe-se uma tendência de aumento do GAP em função do aumento do número de nós das instâncias. No caso do *benchmark* com 100 nós, pode-se observar que para todas as instâncias foi encontrado o número ótimo de veículos, porém houve um pequeno desvio de 0.02% em relação ao custo total. Para 200 nós a diferença foi um pouco maior e para os demais problemas ficou entre 1.0% e 3.0%. Por fim, a média geral da melhor solução ficou em torno de 1.35% para o número de veículos e -2.07% para o custo total. Essa diferença inversa na porcentagem está em concordância com a análise anterior da Tabela 14, onde foi observado na média geral uma tendência de que reduzir o número de veículos tende a aumentar o custo total, e vice-versa.

Tabela 15 – Desvio relativo (GAP) em função da melhor solução encontrada (BSF) pelo ALNS-DPDP em relação aos resultados conhecidos na literatura.

Número de nós	**GAP BSF**	
	NV	**TC**
100 nós	0.00%	0.02%
200 nós	0.67%	-1.54%
400 nós	1.68%	-3.11%
600 nós	1.91%	-3.22%
800 nós	1.75%	-2.42%
1000 nós	2.10%	-2.16%
Média geral	1.35%	-2.07%

Como esse trabalho trata de uma heurística, é interessante realizar a análise da média e do desvio padrão em relação as trinta execuções por instância. Para isso, a Equação 4.3 e a Equação 4.4 são aplicadas. A primeira equação calcula o desvio da literatura (L_i) baseado na média dos resultados das 30 execuções para cada instância de teste B_i^{avg}. Já a segunda equação, calcula a porcentagem média de desvio padrão das 30 execuções, termo S_i^{avg}, em relação a média dos resultados das 30 exe-

cuções, termo B_i^{avg}. Assim como na análise da melhor solução, as duas equações são aplicadas separadamente para cada objetivo do problema.

$$GAP(N) = \frac{\sum_{i \in N} B_i^{avg}}{\sum_{i \in N} L_i} - 1 \tag{4.3}$$

$$SD(N) = \frac{1}{|N|} \sum_{i \in N} \frac{S_i^{avg}}{B_i^{avg}} \tag{4.4}$$

A Tabela 16 apresenta as médias e os desvios padrão, calculados a partir das equações Eq. 4.3 e Eq. 4.4, agrupados pelo tamanho do problema. Nessa tabela percebe-se que em nenhum caso o algoritmo chegou na solução da literatura em todas as 30 execuções, para nenhum dos objetivos. Além disso, é interessante notar que as instâncias com menor GAP são as de 100 e 200 nós, enquanto as com maior GAP são as de 400, 600, 800 e 1000 nós. Na média geral, o GAP do número de veículos foi de 2.27% de média com desvio de 0.99% e o custo total foi de -2.36% de média com desvio de 1.95%.

Tabela 16 – Média e desvio padrão do GAP das trinta execuções de cada instância de teste em relação aos resultados conhecidos na literatura. Resultados obtidos pelo ALNS-DPDP.

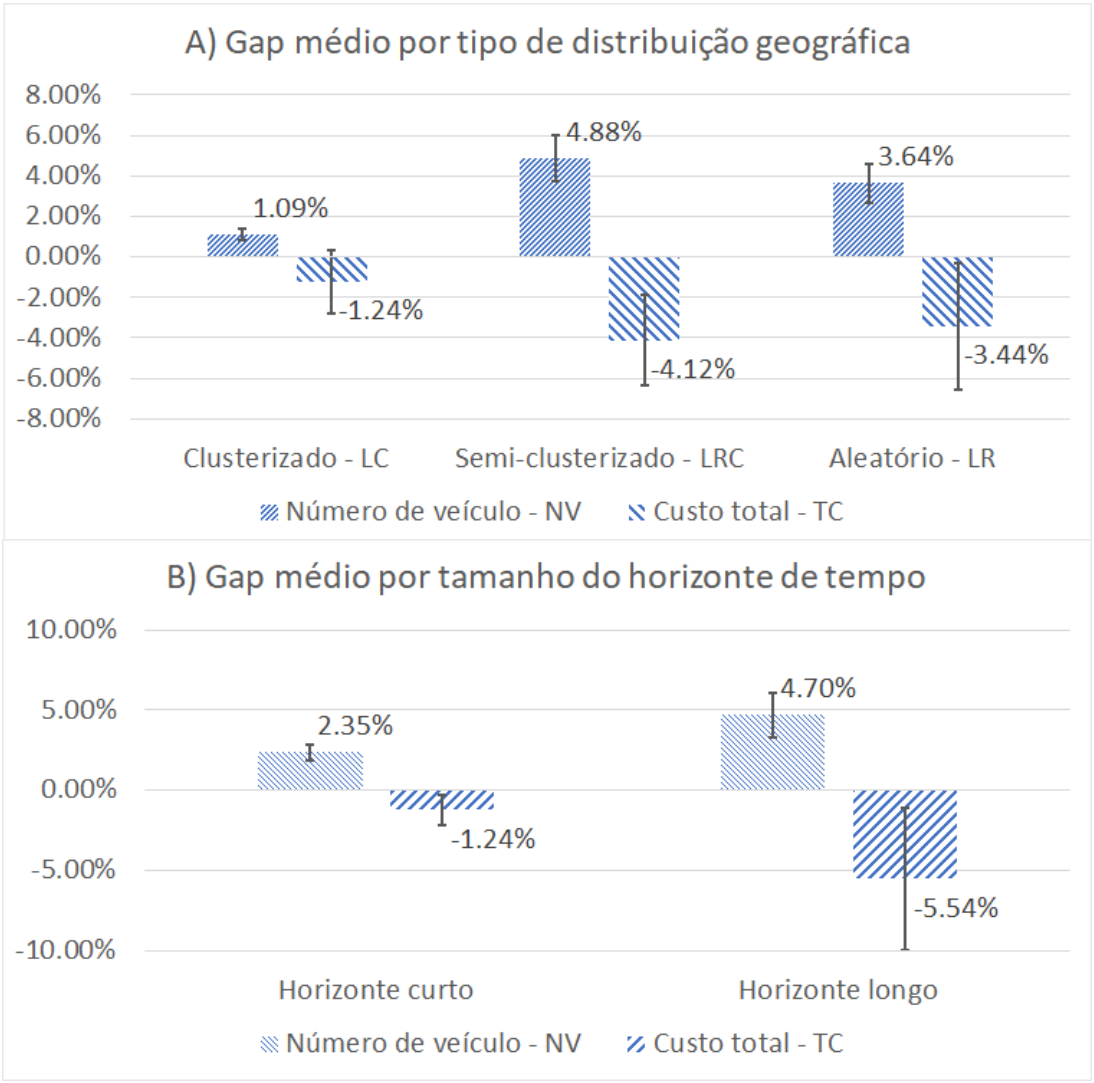
Número de nós	**GAP Média**	
	NV ($\mu \pm \sigma$)	**TC** ($\mu \pm \sigma$)
100 nós	0.03%±0.44%	0.18%±0.26%
200 nós	1.28%±0.64%	-1.41%±1.30%
400 nós	2.84%±0.89%	-3.38%±2.06%
600 nós	2.93%±1.17%	-2.95%±2.96%
800 nós	3.19%±1.50%	-3.82%±2.68%
1000 nós	3.37%±1.31%	-2.78%±2.49%
Média geral	2.27%±0.99%	-2.36%±1.95%

É importante ressaltar que até o momento da escrita desse trabalho, não existe algoritmo universal capaz de encontrar a melhor solução para todas as instâncias. Mais especificamente, desde 2001 dezessete algoritmos foram responsáveis por encontrar as melhores soluções da literatura (SINTEF, 2019). Isso indica que algumas estratégias podem ser mais favoráveis para algumas características do problema do que outras. Por fim, assume-se que os resultados observados na Tabela 16 são aceitáveis para o método proposto, visto que o desvio e o GAP ficaram em torno de 2.3% e que não existe nenhum algoritmo universal capaz de resolver todas instâncias do problema.

Para entender melhor as características do problema que agregam complexidade, a seguir será analisado o GAP médio (Equação 4.3) em função do tipo de distribuição geográfica e do horizonte de tempo. A Figura 20 apresenta os resultados

dessa análise, onde o gráfico A está em função da distribuição geográfica e o gráfico B está em função do tamanho do horizonte de tempo do dia de trabalho.

Figura 20 – Gap médio por diferentes características das instâncias de benchmark. Gráfico A apresenta os resultados agrupados pelo tipo de distribuição geográfica e a o gráfico B apresenta os resultados agrupados pelo tamanho do horizonte de tempo do dia de trabalho.



Fonte: Imagem criada pelo autor.

Como pode ser observado no gráfico A da Figura 20, as instâncias clusterizadas são as com menor GAP, as instâncias com distribuição aleatória são as com GAP intermediário, e as semi-clusterizadas são as com maior GAP. Instâncias clusterizadas tendem a ter um GAP baixo visto que os veículos são direcionados aos *clusters* e isso reduz o tamanho do espaço de busca. Já as semi-clusterizadas tendem a ter o maior custo por não apresentarem um comportamento de distribuição definido, o que faz com que existam maior exploração do algoritmo. Em relação ao gráfico B, o aumento do horizonte de tempo causa um aumento no GAP de NV e TC. Isso acontece porque quando o horizonte de tempo é maior, os veículos têm mais tempo para realizar os atendimentos e podem ser reaproveitados para mais clientes, isso aumenta o espaço

de busca o que dificulta a otimização.

4.2.0.3 Análise interna do ALNS-DPDP

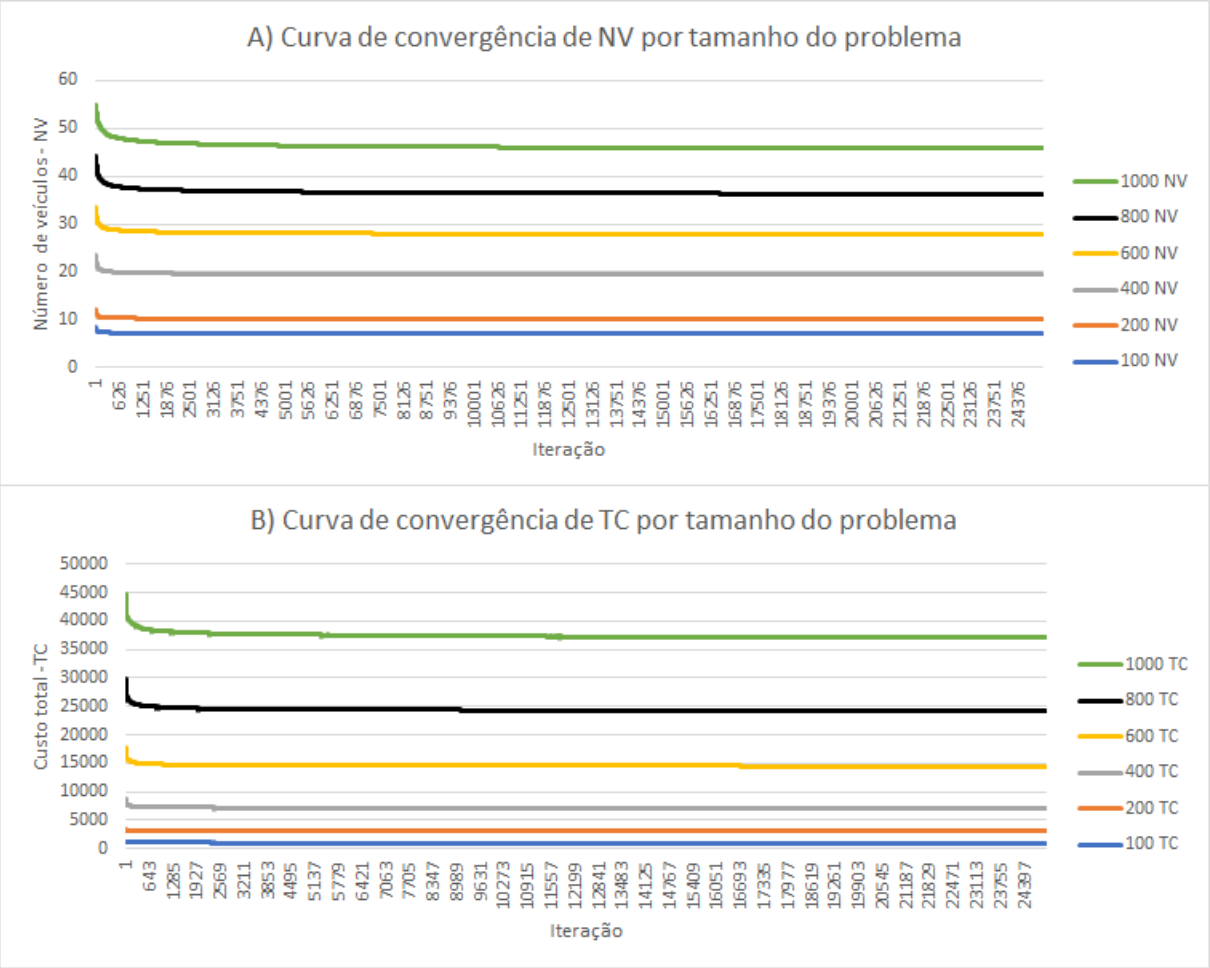
Até então foram analisados os resultados finais obtidos em função das características do problema. Adicionalmente agora será analisado como o ALNS-DPDP se comporta internamente. As analises seguintes serão aplicadas sobre o processo de convergência do algoritmo, ou seja, os dados serão analisados ao longo das iterações. Lembrando que como o *solver* do ALNS-DPDP utiliza dois algoritmos ALNS interno, então se faz necessário analisar os gráficos de convergência separadamente dado que cada ALNS possuí sua parametrização própria.

Para iniciar as análises, a Figura 21 apresenta uma média geral da melhor solução durante a convergência ao longo das 25000 iterações. O gráfico A apresenta a convergência para o objetivo NV enquanto o gráfico B apresenta o objetivo TC. Nesses gráficos, os resultados foram agrupados pelo número de nós, ou seja, foi calculado a média a cada iteração dentre as instâncias com o mesmo número de nós. Com base nos resultados, percebe-se que problemas com 100 nós são os mais rápidos de se resolver, sendo que o algoritmo converge mais rápido. Porém conforme o número de nós vai aumentando a convergência da otimização vai ficando mais lenta. Por exemplo, no caso mais difícil de 1000 nós o maior ganho ocorre até as 3000 iterações porém ao longo as iterações restantes a convergência ainda acontece de forma mais pontual, diferente do problema de 100 nós que fica estagnado após as 3000 iterações. A maior diferença observada é como cada objetivo escala seu valor final (última iteração), por exemplo, com mais número de nós o NV final parece aumentar de forma linear enquanto o TC parece aumentar e de forma exponencial.

Um ponto importante a ser analisado é como a temperatura do $ALNS_{NV}$ e do $ALNS_{TC}$ se comporta ao longo das iterações. A temperatura é importante porque ela determina o nível de exploração da busca exercida pelo algoritmo. Além disso, o ALNS-DPDP executa um mecanismo de recálculo da temperatura quando a melhor solução é alterada, isso serve para sincronizar a execução de ambos algoritmos ALNS com a nova solução. A Figura 22 apresenta o impacto desse recálculo no objetivo principal da solução local de cada algoritmo usando como exemplo a instância LRC1_10_1. Essa instância foi selecionada de forma empírica para mostrar os efeitos do recálculo da temperatura. Nessa Figura o Gráfico A (da esquerda) apresenta os dados do $ALNS_{NV}$ e o Gráfico B (da direita) apresenta os dados do $ALNS_{TC}$. O efeito do recalculo da temperatura pode ser observado no gráfico A quando o número de veículos reduz e a temperatura aumenta. Complementar a isso, pode-se observar como o recálculo da temperatura é propagado do $ALNS_{NV}$ para o $ALNS_{TC}$.

Na Figura 22, é importante atentar-se ao fato de que a temperatura de cada

Figura 21 – Curvas de convergência dos objetivos do problema, número de veículos (NV) e custo total (TC), por número de nós (tamanho do problema). Média da melhor solução a cada iteração do ALNS-DPDP.

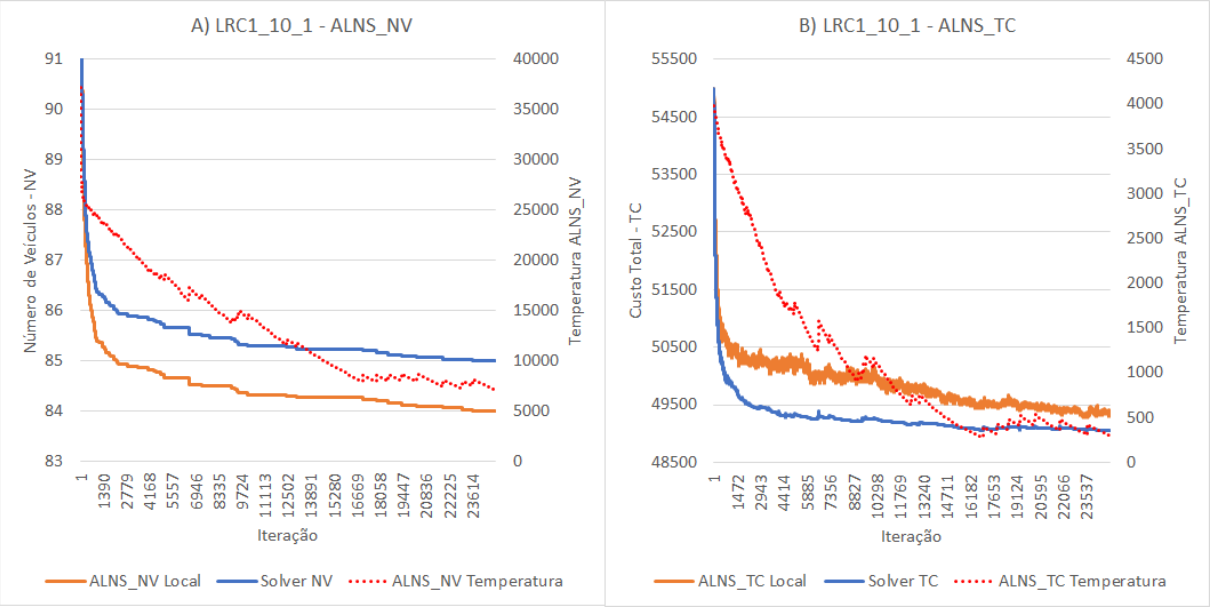


Fonte: Imagem criada pelo autor.

objetivo esta em escalas diferentes. Isso é efeito da parametrização própria dos algoritmos ALNS para atender objetivos diferentes. No caso do $ALNS_{NV}$, a temperatura inicial é maior e a taxa de resfriamento é menor para favorecer a exploração, é isso que faz com que sua temperatura seja mais alta do que a do $ALNS_{TC}$ (40000 versus 4500). Além disso, percebe-se como o controle da exploração ocorre no $ALNS_{TC}$ comparando a melhor solução com a local ao longo das iterações, onde o custo da solução local reduz com a temperatura, porém quando a mesma é recalculada, percebe-se um leve aumento.

Embora o gráfico da Figura 22 apresente detalhes importantes a cerca da convergência do algoritmo, somente essa análise não é suficiente para entender o espectro geral, porque foi considerado somente uma instância. Assim, a Figura 23 apresenta como a temperatura, gráfico A, influencia na exploração do espaço de busca, gráfico B, considerando a média por benchmarks em função do $ALNS_{TC}$. Nesse caso foi se-

Figura 22 – Curvas da melhor solução, da solução local e da temperatura por algoritmo ($ALNS_{NV}$ e $ALNS_{TC}$) durante processo de convergência da instância *LRC1_10_1*, instância com 1000 nós.



Fonte: Imagem criada pelo autor.

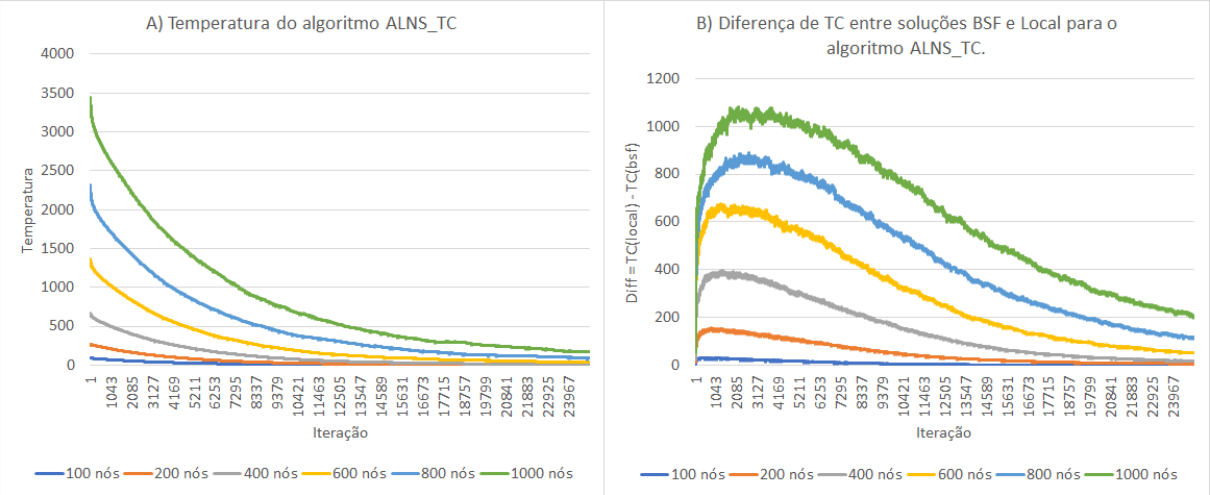
lecionado o $ALNS_{TC}$ porque ele é mais dependente da temperatura para executar a otimização do custo total. Os valores apresentados no gráfico B são a diferença de custo do objetivo TC entre a melhor solução e a solução local. Nesse gráfico, pode-se observar que quanto maior a temperatura maior é a diferença de custo, ou seja, uma alta exploração no começo da busca, que é importante para o algoritmo fugir do mínimos locais. Lembrando que a exploração é feita quando o ALNS aceita soluções de pior custo para tentar encontrar áreas do espaço de busca mais promissoras. O controle da transição da fase explorativa para fase de intensificação é feita pela redução da temperatura do algoritmo. No gráfico B, esse efeito transitório é observado pela redução da diferença de custo.

4.3 ANÁLISE DOS RESULTADOS OBTIDOS NO *BENCHMARK* DINÂMICO

Nesta seção será analisado o impacto da dinamicidade do DPDPTW no ALNS-DPDP. Para isso, será inicialmente apresentado os resultados absolutos e relativos do problema. Em seguida, será apresentada uma análise interna do ALNS-DPDP, conforme descrito na metodologia apresentada na Seção 4.1.4.

A Tabela 17 é composta pela média e desvio padrão das 30 execuções de cada instância do *benchmark* dinâmico. Para melhorar a visualização dos dados, as instâncias foram agrupadas pelo número de nós por meio da média. A parte superior da tabela apresenta a variação do grau de urgência (fator a) e a parte inferior apresenta

Figura 23 – Exploração executada pelo algoritmo $ALNS_{TC}$ com base na diferença de custo da melhor solução e local em relação a temperatura do *simulated annealing*.



Fonte: Imagem criada pelo autor.

a variação do grau de requisições dinâmicas (fator ϱ). Nessa tabela são apresentados os objetivos de otimização NV, TC e FC. Adicionou-se a variável de factibilidade (FC) porque houve casos em que não foi possível obter uma solução completa para o problema, onde o ALNS-DPDP precisou alocar veículos além da frota disponível. Em relação as colunas, a coluna “Literatura” apresenta as melhores soluções da literatura como referência, enquanto as demais apresentam os resultados obtidos para cada grau de dinamicidade. Para destacar a dinamicidade de cada coluna, o parâmetro variado foi marcado em negrito.

A variável de factibilidade foi analisada porque a dinamicidade causada pelas requisições dinâmicas impõe complexidades de otimização que podem extrapolar a capacidade da frota. Na Tabela 17, quando o nível de factibilidade é 100%, isso indica que todos os experimentos conseguiram atender todas requisições usando a frota disponível. Quando a factibilidade é menor, por exemplo 99%, isso indica que 1% dos experimentos precisou alocar mais veículos para atender as requisições porque a frota disponível não foi suficiente. Nos resultados obtidos, pode se observar a redução da factibilidade quando a dinamicidade atingiu graus elevados, ex: $(a = 1.0, \varrho = 1.0)$, $(a = 1.0, \varrho = 9.0)$ e $(a = 1.0, \varrho = 0.75)$. Em um problema real, a extrapolação da frota implicaria na contratação de veículos de terceiros para atender as requisições.

Analisando os resultados da seção superior (”Grau de urgência (fator a)“) da Tabela 17, nota-se que conforme aumenta o grau de urgência os valores dos resultados tendem a aumentar, tanto para o objetivo NV quanto para o TC. Agora analisando os resultados da seção inferior (”Grau de requisições dinâmicas (fator ϱ)“), nota-se que conforme se reduz o grau de requisições dinâmicas, os valores dos resultados tendem a reduzir. Basicamente, quanto menor a urgência ou o grau de requisições

Tabela 17 – Média e desvio padrão dos resultados do *benchmark* DPDPTW. Os resultados das instâncias foram agrupadas pelo tamanho do problema usando a média dos valores. A parte superior da tabela apresenta a variação do grau de urgência (fator a) e a parte inferior apresenta a variação do grau de requisições dinâmicas (fator ϱ).

	Objetivo	Problema	Média literatura	$\boldsymbol{a = 0.1}, \varrho = 1.0$	$\boldsymbol{a = 0.25}, \varrho = 1.0$	$\boldsymbol{a = 0.5}, \varrho = 1.0$	$\boldsymbol{a = 0.75}, \varrho = 1.0$	$\boldsymbol{a = 1.0}, \varrho = 1.0$
Grau de urgência (fator a)	**Número de veículos (NV)**	100	7.18	8.41±0.01	8.27±0.08	8.54±0.13	9.11±0.23	12.77±0.32
		200	10	12.04±0.04	11.57±0.17	11.89±0.29	13.15±0.43	21.78±0.89
		400	18.83	22.97±0.08	22.16±0.32	22.39±0.44	24.37±0.64	38.31±1.18
		600	27.02	32.56±0.15	31.38±0.45	32.29±0.55	34.94±0.76	56.86±1.60
		800	35.15	41.99±0.23	41.15±0.61	42.29±0.66	45.42±0.86	73.98±1.81
		1000	44.31	52.10±0.36	51.61±0.65	53.05±0.71	56.86±0.95	92.77±1.91
	Custo total (TC)	100	1036.78	1098.54±5.31	1128.72±17.26	1207.67±22.83	1331.40±30.99	1736.32±27.62
		200	3063.19	3011.61±25.64	3161.64±70.76	3478.66±103.16	3922.90±128.82	5602.07±146.35
		400	7324.74	7080.79±77.91	7587.71±176.76	8155.38±208.34	9055.99±232.18	12605.78±267.59
		600	14946.99	14614.14±205.11	15571.11±340.43	16772.58±385.39	18472.46±434.72	26063.29±513.11
		800	25041.69	24867.84±374.42	26413.13±546.64	28088.51±582.87	30676.09±595.25	43071.15±746.68
		1000	38081	38417.07±614.45	40309.60±732.50	42610.80±743.08	46345.75±805.60	64786.04±978.98
	Factibilidade (FC)	100	100%	100%±0%	100%±0%	100%±0%	100%±0%	96%±0%
		200	100%	100%±0%	100%±0%	100%±0%	100%±0%	97%±0%
		400	100%	100%±0%	100%±0%	100%±0%	100%±0%	97%±0%
		600	100%	100%±0%	100%±0%	100%±0%	100%±0%	97%±0%
		800	100%	100%±0%	100%±0%	100%±0%	100%±0%	97%±0%
		1000	100%	100%±0%	100%±0%	100%±0%	100%±0%	97%±0%

	Objetivo	Problema	Média literatura	$a = 1.0, \boldsymbol{\varrho = 0.9}$	$a = 1.0, \boldsymbol{\varrho = 0.75}$	$a = 1.0, \boldsymbol{\varrho = 0.5}$	$a = 1.0, \boldsymbol{\varrho = 0.25}$	$a = 1.0, \boldsymbol{\varrho = 0.0}$
Grau de requisições dinâmicas (fator ϱ)	**Número de veículos (NV)**	100	7.18	12.34±0.38	11.86±0.47	10.64±0.48	9.73±0.33	8.68±0.00
		200	10	20.97±0.95	19.81±1.00	17.46±0.88	14.89±0.66	12.48±0.00
		400	18.83	36.52±1.30	34.23±1.30	31.07±1.19	26.81±0.97	23.92±0.00
		600	27.02	54.64±1.55	51.28±1.62	44.59±1.51	38.37±1.18	33.83±0.00
		800	35.15	70.88±1.85	66.38±1.86	59.30±1.72	50.56±1.36	44.38±0.00
		1000	44.31	89.29±2.14	82.90±2.10	74.00±1.89	64.17±1.54	55.14±0.00
	Custo total (TC)	100	1036.78	1731.28±32.27	1670.78±46.86	1497.11±41.03	1335.93±25.12	1097.49±3.95
		200	3063.19	5482.35±150.42	5183.89±166.24	4564.82±142.79	3816.19±95.75	2962.86±14.66
		400	7324.74	12272.38±311.71	11601.81±300.71	10437.23±283.65	8685.85±194.21	7000.73±55.83
		600	14946.99	25388.52±534.11	23964.94±522.89	20888.18±483.41	17571.04±345.57	14187.26±125.83
		800	25041.69	41638.33±794.64	39401.02±792.64	34893.47±687.19	29315.42±520.17	24010.06±227.71
		1000	38081	62762.35±1058.49	58994.31±1079.84	52527.08±918.88	44820.52±735.26	37028.31±349.66
	Factibilidade (FC)	100	100%	0.96±0%	100%±0%	100%±0%	100%±0%	100%±0%
		200	100%	97%±0%	97%±0%	100%±0%	100%±0%	100%±0%
		400	100%	97%±0%	100%±0%	100%±0%	100%±0%	100%±0%
		600	100%	97%±0%	98%±1%	100%±0%	100%±0%	100%±0%
		800	100%	97%±0%	100%±1%	100%±0%	100%±0%	100%±0%
		1000	100%	97%±0%	100%±0%	100%±0%	100%±0%	100%±0%

dinâmicas, mais próximo da coluna literatura os valores dos resultados se encontram. Esse efeito é principalmente causado porque com menos dinamicidade, a maior parte do problema é conhecida mais cedo durante a otimização, o que leva a uma solução de melhor qualidade, que no problema em tela significa menor custo. Ou seja, essa variação dos valores indica uma tendência de aumento da complexidade quando o grau de urgência ou grau de requisições dinâmicas aumenta.

Na Tabela 17 pode-se analisar que para o objetivo NV o desvio padrão é zero quando $a = 1.0$, e $\varrho = 0.0$. Esse efeito é causado pela heurística de inserção que gera a solução inicial e pelo veículo em movimento. Nesse cenário, como todas requisições do *benchmark* são estáticas ($\varrho = 0.0$), a heurística de inserção sempre gera uma solução inicial completa (que atende todos os clientes). Como a heurística de inserção é um processo determinístico que não sofre aleatoriedade, então toda solução inicial será gerada da mesma forma. Nesse sentido, se a instância do problema for a mesma, serão alocados os mesmos veículos para atender as rotas da solução inicial. Além disso, como os veículos iniciam a operação do dia de trabalho na primeira iteração, as restrições temporais (Seção 3.1) impedem que esses veículos sejam cancelados da solução nas iteração subsequentes do ALNS, porque eles já firmaram o atendimento com o primeiro cliente. E ainda, como todas requisições já estão alocadas, também não serão adicionados novos veículos. Portanto, esses dois fatores impedem que o número de veículos sofra variações na primeira iteração e nas iteração seguintes, tornando o desvio padrão igual a zero.

Para analisar quanto as soluções encontradas pelo ALNS-DPDP desviaram das instâncias estáticas da literatura, a Tabela 18 apresenta as melhores soluções encontradas dentre as 30 execuções de cada instância do *benchmark* dinâmico. Nessa tabela, a parte superior se refere a variação do grau de urgência (fator a) e a parte inferior se refere a variação do grau de requisições dinâmicas (fator ϱ). Os valores foram agrupados pelo número de nós usando a média dos resultados. A factibilidade indica a porcentagem dos benchmarks que conseguiu atender todas requisições com a frota disponível. O propósito dessa tabela é apresentar um novo benchmark com custo da melhor solução encontrada pelo ALNS-DPDP para o DPDPTW.

4.3.1 Análise dos resultados relativos em função das instâncias estáticas da literatura

A Figura 24 apresenta uma visão do desempenho do algoritmo no problema dinâmico em relação aos dados estáticos da literatura. Assim como na análise do *benchmark* estático, foi verificado a média e o desvio médio em relação a literatura (GAP). Os resultados obtidos foram primeiramente discretizados pelo grau de dinamicidade (fator a e ϱ) para então aplicar as Equações 4.3 e 4.4. No gráfico as barras represen-

Tabela 18 – Tabela com os valores da melhor solução encontrada para o DPDPTW em relação a média das instâncias do PDPTW da literatura. Foi usado a média para compactar os resultados. A factibilidade indica se todas requisições foram atendidas com a frota disponível.

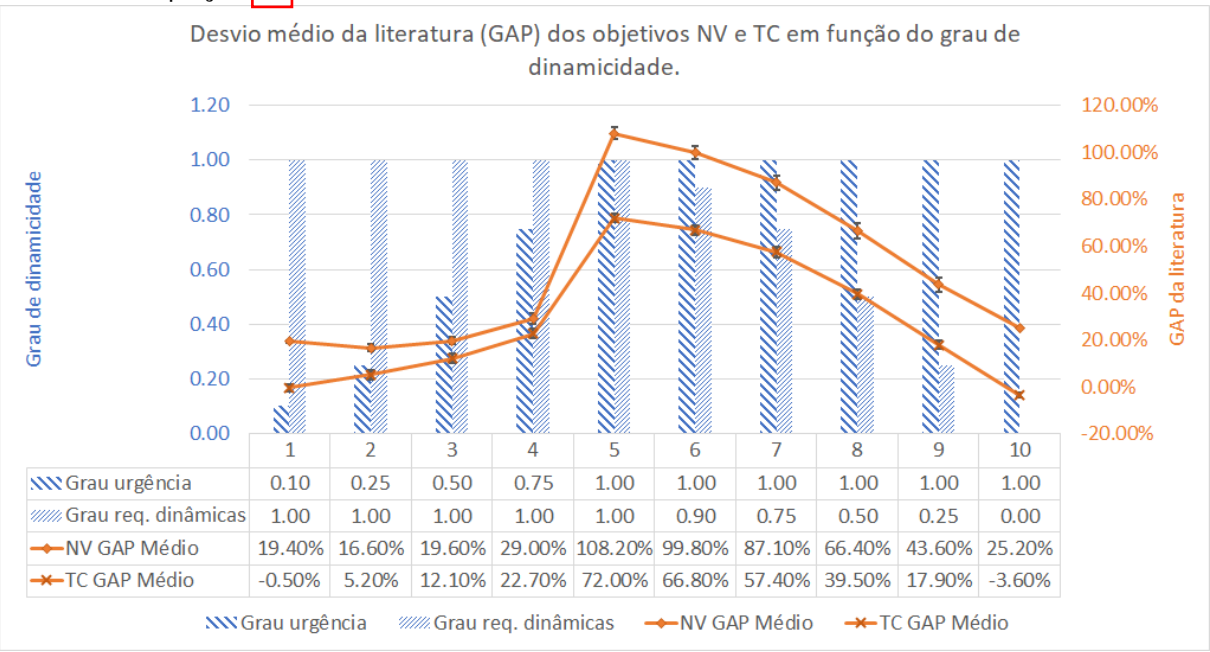
Grau de urgência (fator a)	Objetivo	Nós	Literatura	$\boldsymbol{a = 0.1}$ $\varrho = 1.0$	$\boldsymbol{a = 0.25}$ $\varrho = 1.0$	$\boldsymbol{a = 0.5}$ $\varrho = 1.0$	$\boldsymbol{a = 0.75}$ $\varrho = 1.0$	$\boldsymbol{a = 1.0}$ $\varrho = 1.0$
	Número de veículos (NV)	100	7.18	8.41	8.16	8.43	8.89	12.29
		200	10.00	12.00	11.42	11.55	12.57	20.25
		400	18.83	22.92	21.70	21.73	23.37	36.13
		600	27.02	32.37	30.70	31.35	33.68	53.80
		800	35.15	41.68	40.17	41.17	43.93	70.42
		1000	44.31	51.52	50.41	51.76	55.10	88.81
	Custo Total (TC)	100	1036.78	1092.30	1117.86	1171.62	1287.53	1704.58
		200	3063.19	2980.94	3069.38	3332.56	3730.93	5450.87
		400	7324.74	6963.17	7348.24	7879.48	8760.15	12327.32
		600	14946.99	14340.94	15200.54	16298.20	17962.49	25450.03
		800	25041.69	24371.56	25814.81	27479.50	29982.89	42237.87
		1000	38081	37744.94	39688.34	41700.10	45270.90	63663.36
	Factibilidade (FC)	100	100%	100%	100%	100%	100%	96%
		200	100%	100%	100%	100%	100%	97%
		400	100%	100%	100%	100%	100%	97%
		600	100%	100%	100%	100%	100%	97%
		800	100%	100%	100%	100%	100%	97%
		1000	100%	100%	100%	100%	100%	97%

Grau de requisições dinâmicas (fator ϱ)	Objetivo	Nós	Literatura	$a = 1.0$ $\boldsymbol{\varrho = 0.9}$	$a = 1.0$ $\boldsymbol{\varrho = 0.75}$	$a = 1.0$ $\boldsymbol{\varrho = 0.5}$	$a = 1.0$ $\boldsymbol{\varrho = 0.25}$	$a = 1.0$ $\boldsymbol{\varrho = 0.0}$
	Número de veículos (NV)	100	7.18	11.80	11.05	9.91	9.25	8.68
		200	10.00	19.30	18.07	15.77	13.75	12.48
		400	18.83	34.17	31.77	28.88	25.12	23.92
		600	27.02	51.62	48.27	41.70	36.23	33.83
		800	35.15	67.13	62.85	55.92	48.07	44.38
		1000	44.31	84.98	78.60	70.47	61.10	55.14
	Custo total (TC)	100	1036.78	1700.37	1623.61	1465.60	1313.26	1093.21
		200	3063.19	5378.00	5028.56	4486.40	3718.00	2944.34
		400	7324.74	12013.63	11288.28	10145.58	8488.20	6916.50
		600	14946.99	24897.70	23397.56	20314.21	17209.75	13968.33
		800	25041.69	40743.47	38675.43	34102.82	28668.51	23614.10
		1000	38081	61544.83	57764.48	51496.90	43943.58	36366.25
	Factibilidade (FC)	100	100%	96%	100%	100%	100%	100%
		200	100%	97%	97%	100%	100%	100%
		400	100%	97%	100%	100%	100%	100%
		600	100%	97%	98%	100%	100%	100%
		800	100%	97%	100%	100%	100%	100%
		1000	100%	97%	100%	100%	100%	100%

tam os fatores de dinamicidade, as linhas representam o GAP dos objetivos NV e TC e os números de 1 à 10 no eixo X abaixo das barras se refere ao *benchmark* relacionado a aqueles dados. Referente as escalas, a ordenada da esquerda representa os dados de dinamicidade e a ordenada da direita representa o GAP em porcentagem dos objetivos. Para exemplificar como esse gráfico deve ser lido, o *benchmark* número 1 teve um GAP médio de $NV = 19.40\%$ e $TC = -0.50\%$ em função de um grau de urgência de 10% (fator $a = 0.10$) e de um grau de requisições dinâmicas de 100% (fator $\varrho = 1.00$). Além dos dados de GAP, para cada *benchmark* é apresentado o desvio

padrão médio (barras pretas nas linhas de NV e TC) dos resultados calculados pela Equação 4.4. Por fim, para facilitar a observação dos valores foi adicionada uma tabela na parte inferior com os valores numéricos.

Figura 24 – GAP dos objetivo de minimização do DPDPTW em função do grau de dinamicidade. O GAP é calculado com base na média dos resultados finais, por meio da Equação 4.3 e da Equação 4.4.



	1	2	3	4	5	6	7	8	9	10
Grau urgência	0.10	0.25	0.50	0.75	1.00	1.00	1.00	1.00	1.00	1.00
Grau req. dinâmicas	1.00	1.00	1.00	1.00	1.00	0.90	0.75	0.50	0.25	0.00
NV GAP Médio	19.40%	16.60%	19.60%	29.00%	108.20%	99.80%	87.10%	66.40%	43.60%	25.20%
TC GAP Médio	-0.50%	5.20%	12.10%	22.70%	72.00%	66.80%	57.40%	39.50%	17.90%	-3.60%

Fonte: Imagem gerada pelo autor

Analisando o gráfico da Figura 24, pode-se observar que o GAP tende a ser menor para graus de urgência menor ($0.10 \leq a \leq 0.75$) e aumenta rapidamente a medida que o grau de urgência ultrapassa 0.75. Além disso, percebe-se a dificuldade em reduzir o custo de NV e TC quando $a = 1.0$, mesmo reduzindo o valor de ϱ. Adicionalmente, pode-se observar que o problema mais difícil é dado quando o *benchmark* é executado sob $a = 1.0, \varrho = 1.0$. A dificuldade em otimizar problemas com urgência e dinamicidade alta é causada pelo curto tempo de reação disponível para o algoritmo. Portanto, quando uma requisição dinâmica entra no problema o ALNS-DPDP tem menos iterações para otimizar a solução, o que muitas vezes implica na alocação de um novo veículo para atender essa requisição dinâmica. Dessa forma, quando não é possível encontrar uma solução factível que inclua a nova requisição dinâmica com o mesmo número de veículos, um novo veículo tem que ser alocado para atender a requisição em tempo hábil, o que tende a aumentar NV e TC. Embora o algoritmo se beneficie do tempo de preparação (*setup time*), que é o tempo usado para preparar a requisição em um veículo, esse tempo não é suficiente para garantir uma solução otimizada. O aumento do grau de urgência e dinamicidade das requisições implica na diminuição da eficiência do algoritmo na geração de soluções de longo prazo. Isso é ocasionado pelo fato de que ele não conhece as requisições que chegarão no futuro.

Portanto, a busca acaba tendo que focar em resolver o problema em um curto espaço de tempo sem a possibilidade de um planejamento melhor.

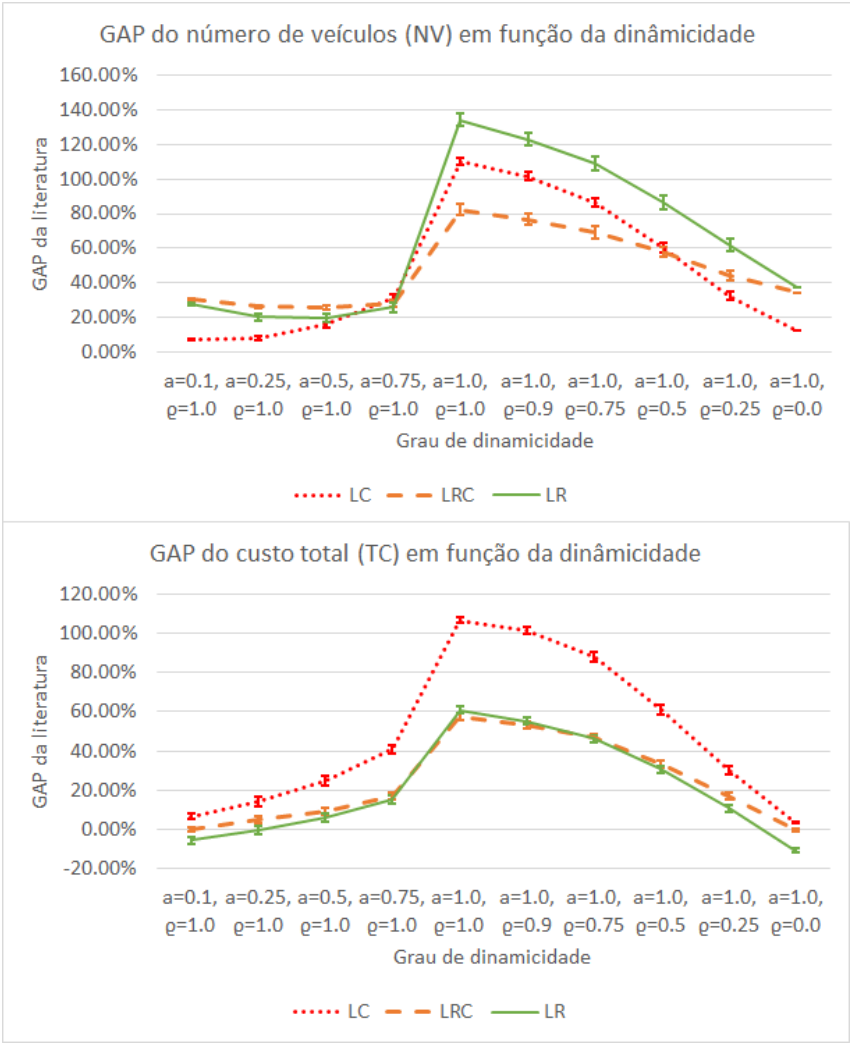
Outro aspecto que pode ser analisado no gráfico da Figura 24 é que entre os *benchmarks* 1 e 2 ($a = 0.10$ e $a = 0.25$) houve uma leve redução do número de veículos. Em um primeiro momento isso é contra intuitivo visto que o *benchmark* 1 deveria ter um NV menor em relação ao *benchmark* 2, já que a urgência é menor. Entretanto, essa redução acontece devido a restrição do *setup time*, que é o tempo mínimo necessário que a empresa exige para conseguir operacionalizar um veículo no atendimento de uma requisição dinâmica. No *benchmark* 1, o *setup time* sofre implicações porque as requisições dinâmicas entram mais cedo no problema. Por exemplo, se o dia de trabalho começa as 8h e uma requisição tem um *setup time* de 30min para ser preparada, caso uma requisição entre no problema as 8:10h, o algoritmo terá menos tempo para tentar reaproveitar a frota, ou seja, será mais propenso a adicionar novos veículos para solução. Em resumo, quanto mais tempo o ALNS-DPDP puder executar antes de iniciar o atendimento de uma nova requisição, melhor tenderá a ser a rota de cada veículo porque as requisições serão melhor alocadas.

Os GAPs apresentados da Figura 24 podem ser interpretados como uma análise de pior caso, pois qualquer análise que aplique outros graus de dinamicidade não devem acarretar em custos piores do que os observados no gráfico. Essa afirmação é suportada pelo fato de que na análise apresentada na Figura 24 sempre é mantido um dos fatores de dinamicidade no grau máximo (valor 1.0), enquanto o outro fator é variado. Portanto, para qualquer outro teste, as condições de dinamicidade vão sempre ser de mais simples solução. Isso pode ser constatado pelo custo observado no *benchmark* número 5, onde $a = 1.0, \varrho = 1.0$, considerado o de pior custo.

Com o objetivo de analisar os resultados em função do tipo de distribuição geográfica, o gráfico da Figura 25 apresenta o GAP médio dos *benchmarks* em função das distribuições clusterizada (LC), semi-clusterizada (LRC) e aleatória (LR). O GAP é analisado do ponto de vista do número de veículos (gráfico superior) e do custo total (gráfico inferior). Nesses gráficos os rótulos do eixo horizontal apresentam os graus de dinamicidade (fator a e fator ϱ).

Os resultados apresentados na Figura 25 permitem duas análises distintas. A primeira é em relação ao número de veículos (gráfico superior), as instâncias LC são as que sofrem menor influência da urgência quando $a \leq 0.5$. Isso é razoável para instâncias LC, porque uma urgência menor implica em mais requisições sendo conhecidas no começo da execução, o que facilita a identificação dos *clusters* para alocar os veículos. A segunda análise é em relação ao custo total (gráfico inferior), nesse caso, o GAP das instâncias LC é sempre maior em relação as instâncias LR e LRC. O motivo é que o custo de desviar os veículos de seus respectivos *clusters* para

Figura 25 – GAP da média dos resultados dos objetivos de minimização do DPDPTW organizados pelo tipo de distribuição geográfica dos nós.



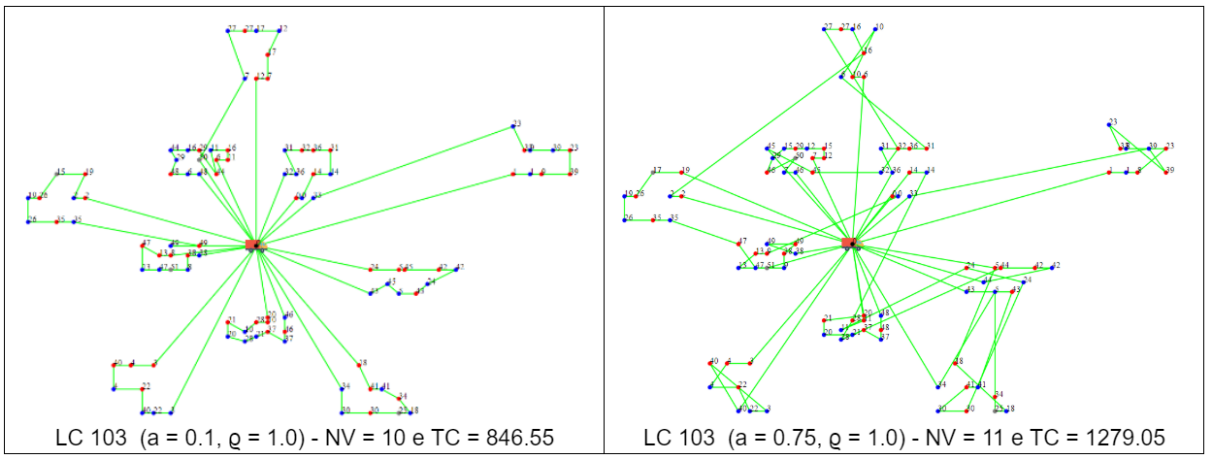
Fonte: Imagem gerada pelo autor

anteder requisições dinâmicas é elevado. A Figura 26 apresenta esse efeito, onde foi selecionada a instância LC103 empiricamente para comparar as urgências de $a = 0.1$ e $a = 0.75$. Nessa figura pode-se perceber como o roteamento se torna mais emaranhado quando a urgência aumenta. A causa disso é o fato do **ALNS-DPDP** focar em reaproveitar os veículos com as requisições dinâmicas sem considerar o desvio que será feito, isso força uma tendência de tirar um veículo do seu *cluster*.

4.3.2 Análise da factibilidade das soluções

No começo da Seção 4.3 foi citado que devido à variação de dinamicidade em alguns casos a frota de veículos estipulada no *benchmark* não foi suficiente para atender a todos os clientes em tempo hábil. Isso pode ser observado no gráfico da Figura 27, que mostra o percentual de factibilidade das soluções para cada grau de dinamicidade analisado. O gráfico segue o mesmo formato do apresentado na Figura

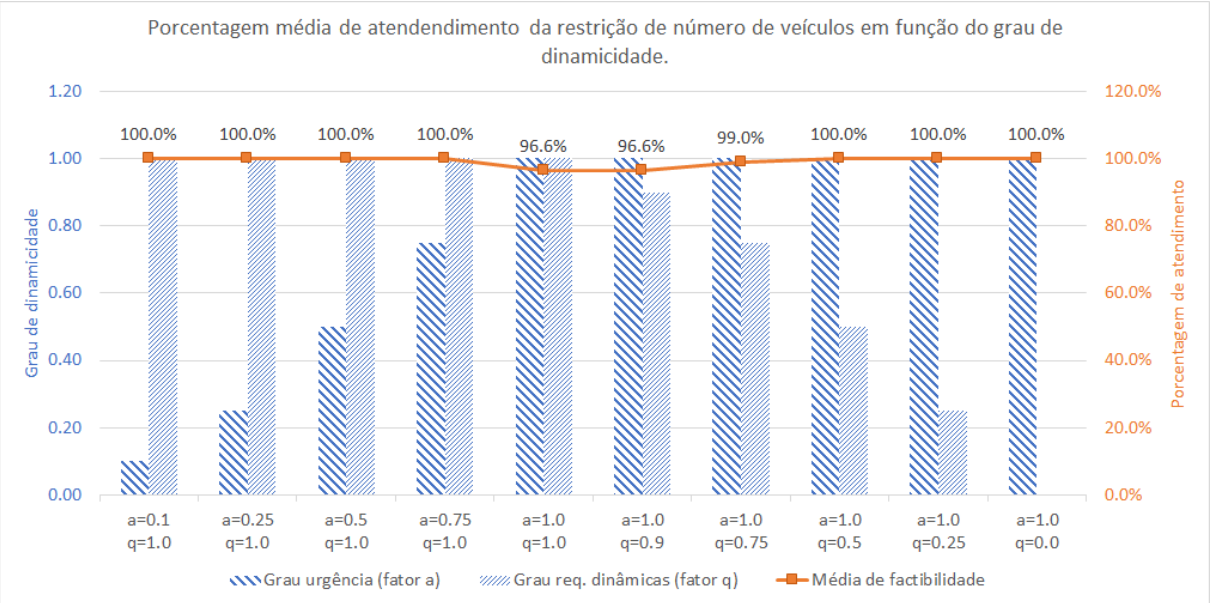
Figura 26 – Efeito colateral causado pela urgência das requisições. Resultado selecionado empiricamente para instância de 100 nós LC103 com $a = 0.1$ e $a = 0.75$.



Fonte: Imagem gerada pelo autor

24, onde cada valor do eixo horizontal se refere a um *benchmark* dinâmico, as barras representam o grau de dinamicidade (urgência e requisições dinâmicas) e a linha representa o nível de factibilidade das soluções. A leitura desse gráfico é feita da seguinte forma, uma média de factibilidade de 100% indica que todas as execuções conseguiram atender os clientes sem alocar mais veículos do que a frota disponível. Uma média de factibilidade de 99% indica que 1% das execuções tiveram que alocar veículos extras.

Figura 27 – Factibilidade média da restrição da capacidade da frota em função do grau de dinamicidade. Um factibilidade de 99% indica que para 1% das instâncias não foi possível atender todos clientes com a frota disponível.



Fonte: Imagem gerada pelo autor

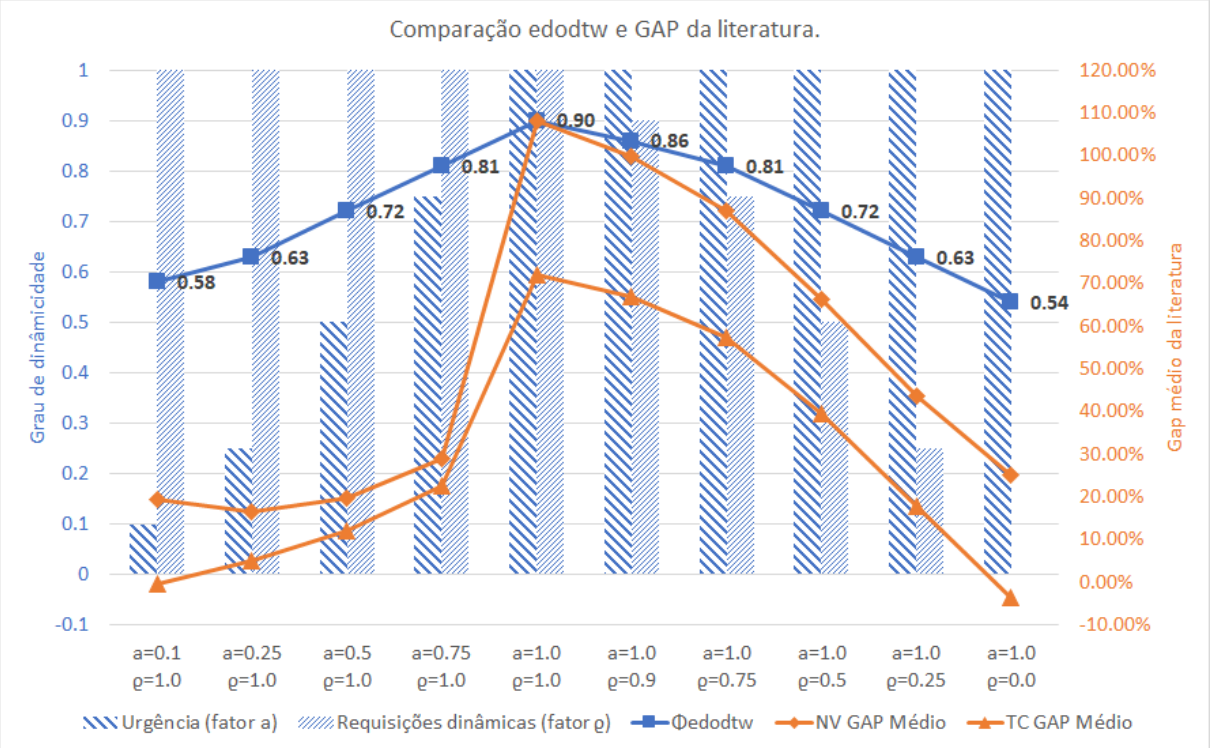
Os resultados mostrados no gráfico da Figura 27 indicam que quando o grau de urgência é máximo ($a = 1.0$) e o grau de requisições dinâmicas é alto ($q = 1.0, 0.9$

ou 0.75) começa a ocorrer a quebra da factibilidade. Isso acontece porque nessas condições a urgência restringe o tempo de otimização, fazendo com que o ALNS-DPDP não tenha tempo hábil para encontrar uma solução melhor. Quando as requisições são anunciadas em cima da hora, o ALNS-DPDP não consegue planejar soluções de forma que os veículos sejam distribuídos adequadamente pela área de abrangência das entregas. Essa dificuldade na distribuição dos veículos faz com que as rotas não tenham folgas disponíveis para novas requisições. Além de aumentar o custo, isso causa também a quebra da factibilidade.

4.3.3 Grau de dinamicidade efetiva com janela de tempo

Na seção 2.3.1.3 foi apresentada a medida do grau de dinamicidade efetiva com janela de tempo Φ_{edodtw}, que serve para medir o grau de dinamicidade no domínio $0 \leq \Phi_{edodtw} \leq 1$). Com base nisso, o objetivo da análise a seguir é comparar o comportamento do GAP observado em relação ao valor de Φ_{edodtw}, verificando assim se existe uma correlação entre os valores. Para realizar essa avaliação, os cálculos foram agrupados conforme a variação dos fatores a e ϱ. A Figura 28 apresenta os resultados obtidos. Nesse gráfico são apresentados os graus de dinamicidade de a e ϱ (barras azuis), os valores medidos de Φ_{edodtw} (linha azul) e os GAPs observados (linhas laranja).

Figura 28 – Resultado do *benchmark* DPDPTW organizados por grau de dinamicidade, comparação da medida Φ_{edodtw} com o GAP dos resultados obtidos pelo ALNS-DPDP.



Fonte: Imagem gerada pelo autor

Um dos resultados observados no gráfico da Figura 28 é que mesmo conhecendo todas as requisições *a priori* (*benchmark* onde $a = 1.0$ e $\varrho = 0.0$), ainda assim existe um grau considerável de dinamicidade quando se analisa a medida Φ_{edodtw}, pois ela resulta em um valor de dinamicidade de 0.54. Isso ocorre porque o cálculo de Φ_{edodtw} é baseado no tempo de reação, que é calculado usando como base o tempo de início do dia de trabalho e as janelas de tempo das requisições. Entretanto, apesar do valor considerável de dinamicidade Φ_{edodtw} no *benchmark* ($a = 1.0$ e $\varrho = 0.0$), este ainda é considerado de baixa complexidade, pois tem o maior tempo de reação dentre todos os outros cenários analisados.

Em relação ao grau de dinamicidade, pode-se observar na Figura 28 que a medida Φ_{edodtw} aumenta em função dos fatores urgência e requisições dinâmicas, chegando ao seu pico quando $a = 1.0$ e $\varrho = 1.0$. Esse comportamento está condizente com o esperado, pois espera-se que o tempo de reação, na média, reduza com o aumento do grau de urgência e diminua com a diminuição da quantidade de requisições dinâmicas.

Ao comparar os valores de Φ_{edodtw} observados na Figura 28 com o GAP das soluções, pode-se observar que eles têm um comportamento semelhante, exceto no objetivo NV do *benchmark* 1 ($a = 0.10$ e $\varrho = 1.0$), onde seu GAP é menor que o observado no *benchmark* 2 ($a = 0.25$ e $\varrho = 1.0$). A maior diferença entre o GAP e Φ_{edodtw} está na taxa de aumento do GAP, pois, enquanto a medida Φ_{edodtw} segue um crescimento quase linear, os GAPs aumentam e diminuem de forma não linear. Embora existam essas diferenças, a correlação de Pearson entre Φ_{edodtw} e o GAP_{NV} é de 0.78 e entre Φ_{edodtw} e o GAP_{TC} é de 0.90. Nesse sentido, o uso da medida Φ_{edodtw} é interessante porque ela é capaz de abstrair os fatores de dinamicidade do problema em uma visão única. Embora essa análise esteja comparando medidas de contextos diferentes, ainda assim é possível usar a medida Φ_{edodtw} para indicar a complexidade introduzida pela utilização da dinamicidade.

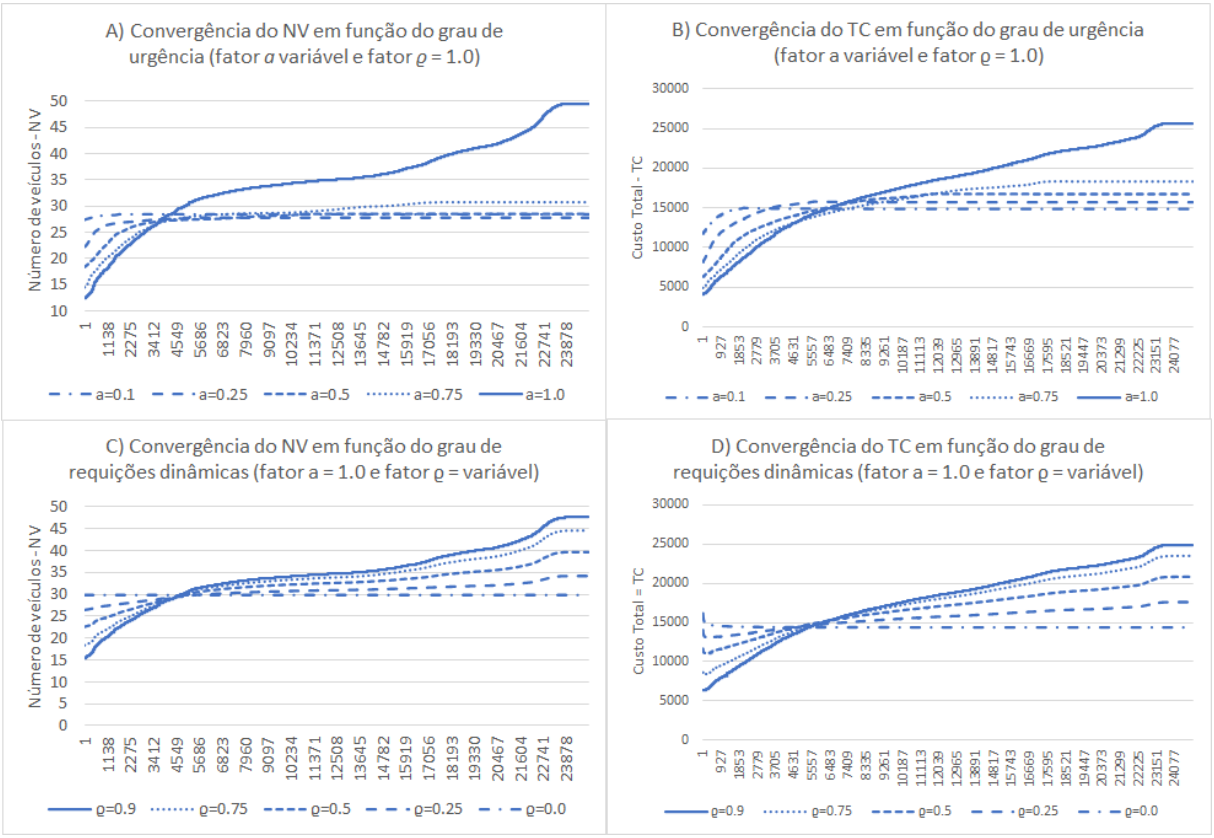
4.3.4 Análise interna do ALNS-DPDP em problemas dinâmicos

Para analisar o comportamento interno do ALNS-DPDP em problemas dinâmicos é necessário analisar as curvas de convergência das variáveis NV e TC. Porém, isso deve ser feito diferenciando os graus de dinamicidade. Além disso, como o ALNS-DPDP é baseado no ALNS, faz-se necessário analisar as seguintes variáveis: melhor solução do *solver* (*BSF*), soluções locais e temperatura do ALNS. Como o *solver* possuí dois ALNS diferentes, as variáveis devem ser analisadas em relação ao objetivo do algoritmo. Portanto, no caso do $ALNS_{NV}$ deve ser analisado o objetivo NV, já no caso do $ALNS_{TC}$ deve ser analisado o objetivo TC.

Os gráficos da Figura 29 apresentam as curvas de convergência dos objetivos

da melhor solução discretizados pelo grau de dinamicidade. Nesses gráficos, para cada iteração do algoritmo calculou-se a média daquele benchmark. Os gráficos A e C apresentam os resultados do objetivo NV e os gráficos B e D apresentam os resultados do objetivo TC. Além disso, os gráficos A e B apresentam os resultados variando-se o grau de urgência e os gráficos C e D apresentam os resultados variando-se o grau de requisições dinâmicas.

Figura 29 – Curvas de convergência de todas execuções do ALNS-DPDP agrupados por grau de dinamicidade. Para cada grau de dinamicidade foi calculado a média dos objetivos de minimização da melhor solução a cada iteração do algoritmo.



Fonte: Imagem gerada pelo autor

Analisando os gráficos da Figura 29, pode-se observar que o comportamento das curvas de convergência varia em função do tipo de da urgência ou das requisições dinâmicas. No caso da urgência, gráficos A e B para fatores de $a = [0.1, 0.25, 0.5]$, a diferença de custo entre curvas na primeira iteração é maior em relação a diferença de custo na última iteração. Isso indica que o algoritmo não sofre muito impacto pela urgência quando utilizados esses fatores, dado que o custo final para eles é próximo. Porém, quando a urgência é igual a 0.75, começa-se a perceber uma diferença mais significativa no custo da solução. Ainda, quando a urgência é máxima, igual a 1.0, o impacto final no custo é muito significativo. Outro ponto que merece destaque é que quanto maior é a urgência do problema, menor é o custo na primeira iteração, porque o número de requisições conhecidas *a priori* é reduzido. Entretanto, maior é a urgência

e maior é o custo final da solução porque as requisições tardias são mais difíceis de serem alocadas por causa das reduzidas janela de disponibilidade dos veículos.

Em termos do grau de requisições dinâmicas, nos gráficos C e D da Figura 29 percebe-se que o custo da iteração final é melhor distribuído no espectro do melhor e do pior caso ($\varrho = 0.9$ e $\varrho = 0.0$), diferente dos gráficos de urgência A e B. Nesse caso, a dinamicidade máxima ($a = 1.0$ e $\varrho = 0.9$) torna mais difícil para o *solver* lidar com o problema em tempo hábil. Conforme o fator ϱ diminui, a complexidade do problema é suavizada porque mais requisições são conhecidas *a piori*. Isso acontece porque com mais requisições conhecidas *a priori* o *solver* consegue gerar soluções com um arranjo mais otimizado no começo da execução, o que reduzir o impacto da urgência alta.

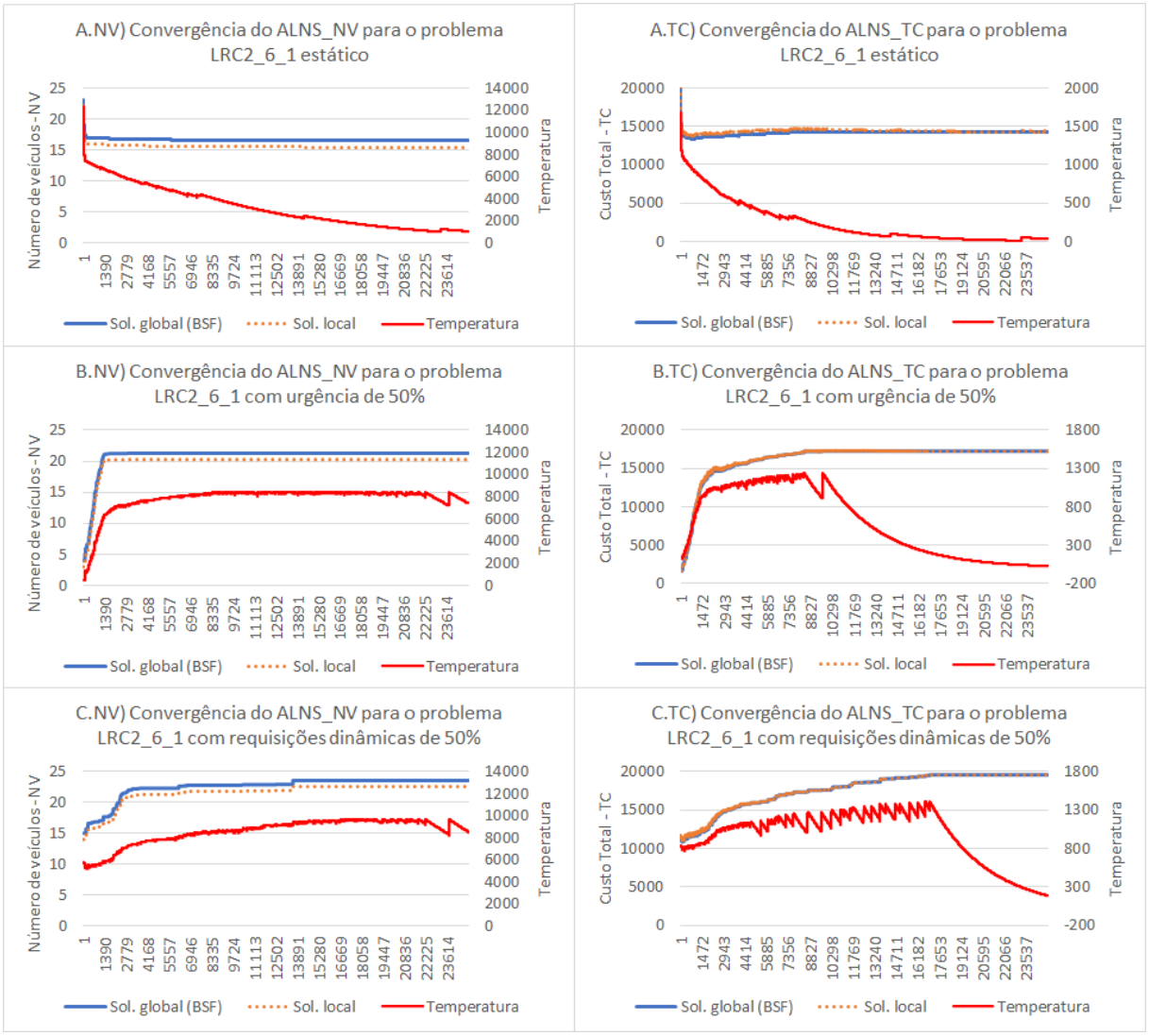
Um padrão observado nas curvas da Figura 29 é o constante aumento das curvas de TC e NV ao longo do tempo, com exceção do gráfico D onde ocorrem algumas reduções do TC nas primeiras iterações de algumas curvas ($\varrho = [0.75, 0.5, 0.25, 0.0]$). Nesse cenário, conforme o número de requisições dinâmicas diminuí (fator ϱ), mais requisições são conhecidas *a priori*. Isso significa que ao longo do tempo menos requisições serão anunciadas e menos intervenções no roteamento irão acontecer, deixando o algoritmo otimizar o custo de forma mais eficiente.

4.3.5 Análise interna do ALNS-DPDP para um problema específico

Essa seção tem por objetivo avaliar como os gatilhos de controle da temperatura funcionam no ALNS-DPDP para o problema dinâmico. Para realizar essa avaliação, foram selecionadas empiricamente três instâncias de 600 nós do *benchmark* LRC2_6_1: versão estática (LRC2_6_1), versão com 50% de urgência (LRC2_6_1_a_0.5) e versão com 50% de requisições dinâmicas (LRC2_6_1_ϱ_0.5). Essas instâncias permitiram fazer uma análise mais profunda dos gatilhos de controle da temperatura. A Figura 30 apresenta os gráficos de convergência do ALNS-DPDP para as instâncias em questão. Nessa figura, os gráficos A.NV e A.TC são referentes ao *benchmark* estático, os gráficos B.NV e B.TC são referentes ao *benchmark* com grau de urgência de 50% e os gráficos C.NV e C.TC são referentes ao *benchmark* com grau de requisições dinâmicas de 50%. Além disso, os gráficos A.NV, B.NV e C.NV são referentes ao objetivo NV do $ALNS_{NV}$ e os gráficos A.TC, B.TC e C.TC são referentes ao objetivo TC do $ALNS_{TC}$.

Antes de iniciar a análise da temperatura, é interessante notar que no gráfico A.TC o custo total da solução é otimizado no primeiro momento e em seguida aumenta levemente. O esperado seria que, em um problema estático, o custo total sempre reduzisse. Entretanto, nesse problema quando o $ALNS_{NV}$ consegue otimizar o número de veículos, a tendência é que a realocação das requisições do veículo removido au-

Figura 30 – Gráficos de convergência do ALNS-DPDP e dos objetivos de minimização para diferentes graus de dinamicidade. Os resultados são calculados usando a média das 30 execuções do experimento de 600 nós LRC2_6_1.



Fonte: Imagem gerada pelo autor

mente o custo das rotas dos outros veículos.

No ALNS-DPDP o recálculo da temperatura é realizado toda vez que um dos seguintes gatilhos é disparado: 1) o número de veículos é minimizado pelo $ALNS_{NV}$, 2) uma requisição dinâmica é anunciada ou 3) um veículo em movimento termina de atender um cliente e se compromete com o próximo (ver Seção 3.2.2). Esse recálculo é importante para escalar a temperatura em função da melhor solução quando ela sofre atualizações. Quando a temperatura não está na escala certa a exploração é prejudicada. Portanto, o controle correto da temperatura é fundamental para que o algortimo consiga explorar adequadamente todo o espeço de busca. Os gatilhos 1, 2 e 3 são ativados no $ALNS_{NV}$, enquanto apenas os gatilhos 1 e 2 são ativados no $ALNS_{TC}$. Dada essa configuração dos gatilhos, é esperado que o $ALNS_{NV}$ sofra mais recálculos de temperatura do que o $ALNS_{TC}$, por ter um gatilho a mais. Esse

efeito pode ser observado nos gráficos B.NV e C.NV da Figura 30 onde o veículo em movimento está ativado.

Na Figura 30, toda vez que a temperatura aumenta no $ALNS_{NV}$, pelos gatilhos 1 e 2, ela também aumenta no $ALNS_{TC}$. Isso fica mais evidente para o problema estático, que não sofre influência das requisições dinâmicas e do veículo em movimento (gráficos A.NV e A.TC). Além disso, toda elevação de temperatura propagada do $ALNS_{NV}$ para o $ALNS_{TC}$ é feita em escalas diferentes. As escalas variam de acordo com o objetivo de cada algoritmo. Pode-se observar que no gráfico A.NV a escala vai de 0-14000, para favorecer mais a exploração, enquanto no gráfico A.TC a escala vai de 0-2000, para favorecer mais a intensificação.

Em relação ao gatilho das visitas dos clientes por veículos em movimento (gatilho 3), na Figura 30 pode-se observar que a temperatura dos gráficos B.NV e C.NV são constantemente atualizados até próximo do final da execução (23000 iterações), enquanto que os gráficos B.TC e C.TC sofrem menos atualizações (até 10000 e 18000 iterações, respectivamente). Isso ocorre porque, quando um cliente é visitado o algoritmo do *solver* deve garantir que a melhor solução entre *solver* e os ALNS sejam iguais, caso contrário será aberto brechas para geração de inconsistências na otimização.

A diferença na atualização das temperaturas entre os gráficos da Figura 30 é causada principalmente pela forma de atualização do $ALNS_{NV}$ e $ALNS_{TC}$, que é diferente. No caso do $ALNS_{NV}$, toda vez que um veículo é removido suas requisições se tornam avulsas, isso abre brecha para na próxima iteração, se um cliente de uma das requisições avulsas for visitado, gerar uma inconsistência. Dessa forma, é necessário a atualização da solução e da temperatura do $ALNS_{NV}$. Já no caso do $ALNS_{TC}$, como o algoritmo não força a remoção de um veículo, a cada iteração é garantido que se um cliente for visitado, a solução entre o *solver* e o $ALNS_{TC}$ serão iguais.

Além da visita dos clientes, outro ponto que gera atualização da temperatura é a entrada de requisições dinâmicas. É interessante notar que no gráfico B.TC, por ser uma instância com 50% de urgência, até aproximadamente metade da execução há novas requisições sendo inseridas no problema. Já no caso do gráfico C.TC, por ser uma instância com 50% de requisições dinâmicas e 100% de urgência, até aproximadamente o final da execução há novas requisições sendo inseridas. Portanto, esse comportamento indica que problemas com maior dinamicidade de urgência dificultam a diminuição da temperatura do *solver*.

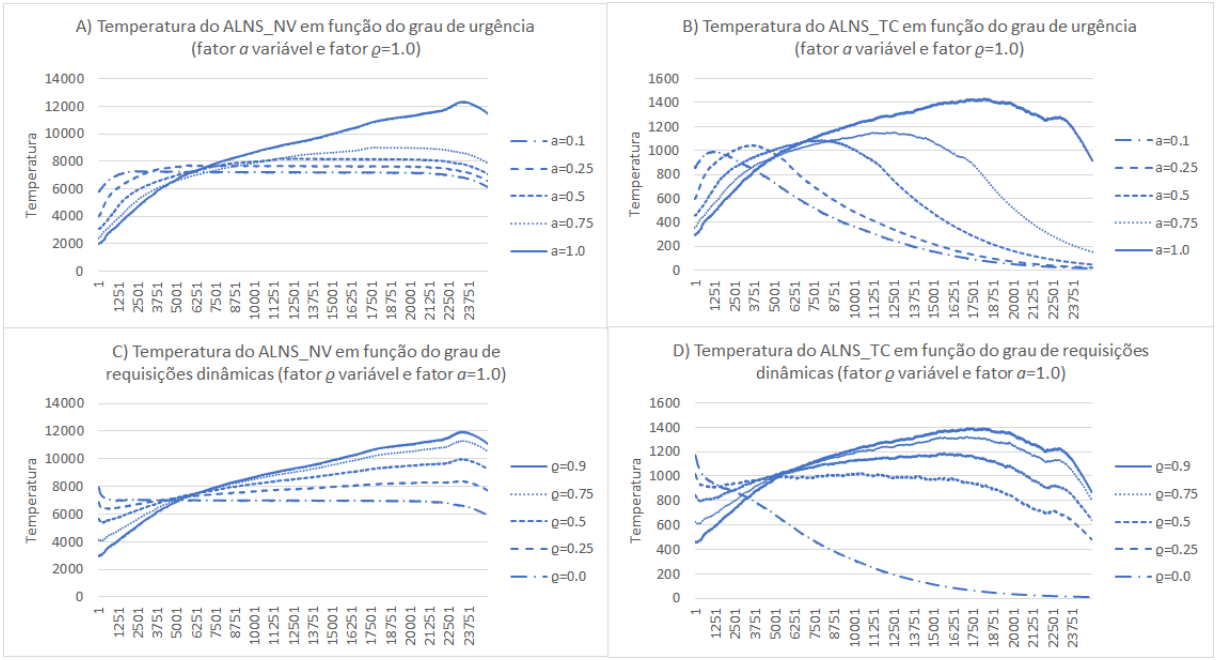
O impacto da temperatura também pode ser observado comparando-se a solução local em relação a melhor solução na Figura 30. Como o nível de exploração dos algoritmos é determinado pela temperatura, quando a temperatura é maior, a solução

local tem um maior descolamento em relação a melhor solução. O efeito da exploração pode ser observado nos gráficos A.TC, B.TC e C.TC, onde a melhor solução (linha pontilhada) tende a ter um custo maior em relação a melhor solução (linha contínua). Conforme a temperatura vai reduzindo, a solução local tende a reduzir sua variação, tornando assim a busca mais intensa. No caso do $ALNS_{NV}$, como esse algoritmo trabalha otimizando com um veículo a menos, a variação desse objetivo ocorre de forma que a melhor solução tenha sempre o custo de um veículo a mais em relação a solução local.

4.3.6 Análise da temperatura em função da dinamicidade

Para verificar como a temperatura se comporta no cenário dinâmico, a Figura 31 apresenta os gráficos de temperatura em função de diferentes graus de dinamicidade. Em termos de tipo de dinamicidade, os gráficos A e B se referem ao fator do grau de urgência, os gráficos C e D se referem ao fator do grau de requisições dinâmicas. Em relação aos objetivos do problema, os gráficos A e C se referem ao $ALNS_{NV}$ enquanto os gráficos B e D se referem ao $ALNS_{TC}$.

Figura 31 – Gráficos da temperatura em diferentes graus de dinamicidade do ALNS-DPDP. Os gráficos estão organizados pelo objetivo de minimização e pelo tipo de dinamicidade. Os gráficos superiores são referentes a variação da urgência e os gráficos inferiores são referentes a variação do número de requisições dinâmicas



Fonte: Imagem gerada pelo autor

Na Figura 31 pode-se observar que a temperatura do algoritmo $ALNS_{NV}$ (gráficos A e C) apresenta uma tendência de aumento até aproximadamente 22000 iterações e depois começa a reduzir. Isso acontece por causa dos gatilhos de recálculo da temperatura, onde para manter o $ALNS_{NV}$ atualizado com a solução usada pelo *sol-*

ver, é necessário constantemente atualizar o algoritmo quando um cliente é visitado. Entretanto, quando os veículos visitam o último cliente e não há novas requisições a serem otimizadas, a temperatura começa a reduzir. Isso acontece próximo das 22000 iterações. Ainda em relação ao algoritmo $ALNS_{NV}$, a temperatura no gráfico A da curva de fator $a = 1.0$ tem um maior distanciamento em relação as outras curvas. Ou seja, o fator $a = 1.0$ caracteriza o cenário dinâmico mais complexo.

Em relação a temperatura do $ALNS_{TC}$ apresentada nos gráficos B e D da Figura 31, a primeira análise a ser feita é acerca da temperatura em função do aumento de urgência (gráfico B). Nesse caso, como a temperatura é recalculada na anunciação de requisições dinâmicas, ela tende a reduzir a partir do momento que elas param de ser anunciadas. Por exemplo, para uma urgência de 10%, a temperatura aumenta até aproximadamente 10% das iterações (que é aproximadamente 2500 iterações), depois disso a temperatura começa a reduzir, permitindo assim que a busca seja intensificada. A segunda análise é em relação ao comportamento da temperatura do $ALNS_{TC}$ em função do grau de requisições dinâmicas (gráfico D). Nesse cenário, como a urgência é máxima, existe a possibilidade de se anunciar novas requisições até o final da execução. Esse efeito dificulta o processo de intensificação do algoritmo porque a temperatura é mantida elevada durante mais tempo. Entretanto, reduzindo o número de requisições dinâmicas (fator ϱ), observa-se a redução da temperatura média dado que o custo das soluções encontradas tende a ser menor. Nesse cenário o melhor caso é onde não existem requisições dinâmicas ($\varrho = 0.0$).

De forma geral, a temperatura é o principal fator que influencia na exploração e intensificação do espaço de busca. Porém, como ela é determinada pelo custo total da solução inicial, variações que impactam no custo da solução inicial devem atualizar os ALNS com a nova temperatura. Um exemplo de sincronização ocorre quando o $ALNS_{TC}$ está melhorando uma solução local (baixa temperatura) e o $ALNS_{NV}$ encontra uma solução com menos veículos. Nesse caso, a temperatura do $ALNS_{TC}$ precisa ser recalculada para a escala de custo da nova solução do $ALNS_{NV}$, visto que a solução do $ALNS_{NV}$ segue um arranjo diferente, pois foca em um objetivo diferente do $ALNS_{TC}$. Outro exemplo é quando uma requisição dinâmica é anunciada, nesse momento o custo da solução será acrescido visto que novas requisições adicionam custo a solução. Como essas variações de custo mudam o contexto da busca, as temperaturas devem ser recalculadas. Por fim, é possível inferir que quanto maior é a dinamicidade, mais aleatória se torna a busca, pois o método tem menos tempo para executar a otimização pensando no longo prazo, o que torna a busca suscetível a resolver o problema parcialmente.

4.4 ANÁLISE DO TEMPO DE EXECUÇÃO

Esta seção visa avaliar o tempo de execução dos algoritmos. Análises de tempo de execução requerem que todos os experimentos sejam feitos em ambiente homogêneo, tanto de software como de hardware, e sejam executados por um número suficiente de vezes, geralmente mais 30 execuções por instância. Entretanto, dado o elevado tempo de experimentação requerido para executar 30 vezes cada instância do *benchmark* utilizado neste trabalho, optou-se por utilizar uma abordagem diferente. A título de exemplo, contando que o *benchmark* tem 3.540 instâncias de teste, a realização de 30 execuções resultaria em aproximadamente 2 meses e meio de experimentação. Para reduzir o tempo necessário para realizar essa análise, foi feita uma única execução para cada instância. Embora essa não seja a análise ideal, ao menos ela permite ter uma visão parcial do tempo de execução do algoritmo.

O hardware disponível para realizar a análise de tempo é o mesmo utilizado na realização dos outros experimentos (ver Tabela 4). O critério usado foi o desempenho do processador. O PC1 tem 8 núcleos e o maior *clock* de todos, entretanto o seu hardware é do ano de 2014 e já está defasado. Já o PC3 possui 80 núcleos de processamento, porém seu *clock* é baixo, o que resulta em tempos de execução maiores. Por fim, o PC2 possuí 8 núcleos, um *clock* razoável e um hardware de 2019. Portanto, ele foi o escolhido para realizar os experimentos. Como o PC2 só possuí 8 núcleos, as execuções tiveram de ser paralelizadas em 6 experimentos por vez, essa estratégia resultou em 3 dias de experimentação para uma execução de cada uma das 3.540 instâncias testadas.

A Tabela 19 apresenta os resultados da experimentação em minutos. Os agrupamentos foram feitos pelo número de nós (coluna *Problema*) e pela distribuição geográfica e tamanho do horizonte de tempo (coluna *Grupo*). As colunas seguintes representam cada tipo de instância, um coluna para as instâncias estáticas, cinco para as dinâmicas com variação do grau de urgência (colunas com o fator a em negrito) e, por fim, cinco para as dinâmicas com variação do grau de requisições dinâmicas (colunas com o fator ϱ em negrito). Os resultados em cada célula são a média e o desvio padrão da única execução das instâncias do grupo. A última linha da tabela apresenta a média geral de cada coluna, incluindo o desvio padrão.

Os resultados da Tabela 19 indicam que as instâncias estáticas consomem o maior tempo de execução, com média total de 6.20 minutos. Como as instâncias estáticas não possuem o veículo em movimento (que implica na redução do espaço de busca a medida que os clientes são visitados), em toda iteração todos os nós podem ser permutados, isso resulta em mais computações e um maior tempo de execução. Em relação ao grupo do problema, em todos os problemas as instâncias LR2 con-

Tabela 19 – Tempos de execução em minutos do ALNS-DPDP. Valores de uma única execução do algoritmo sumarizados por tamanho e grupo. Análise realizada com o PC2 da Tabela 4.

Problema	Grupo	Estático	$\boldsymbol{a = 0.1}$ $\varrho = 1.0$	$\boldsymbol{a = 0.25}$ $\varrho = 1.0$	$\boldsymbol{a = 0.5}$ $\varrho = 1.0$	$\boldsymbol{a = 0.75}$ $\varrho = 1.0$	$\boldsymbol{a = 1.0}$ $\varrho = 1.0$	$a = 1.0$ $\boldsymbol{\varrho = 0.9}$	$a = 1.0$ $\boldsymbol{\varrho = 0.75}$	$a = 1.0$ $\boldsymbol{\varrho = 0.5}$	$a = 1.0$ $\boldsymbol{\varrho = 0.25}$	$a = 1.0$ $\boldsymbol{\varrho = 0.1}$
100	LC1	0.16±0.03	0.10±0.02	0.09±0.02	0.08±0.01	0.06±0.01	0.05±0.01	0.05±0.01	0.06±0.01	0.07±0.01	0.09±0.01	0.10±0.02
	LC2	0.43±0.17	0.16±0.03	0.13±0.02	0.08±0.01	0.05±0.00	0.03±0.01	0.04±0.01	0.05±0.01	0.08±0.01	0.11±0.02	0.17±0.04
	LR1	0.18±0.05	0.10±0.01	0.11±0.02	0.11±0.01	0.10±0.01	0.09±0.01	0.09±0.01	0.09±0.01	0.09±0.01	0.10±0.01	0.11±0.01
	LR2	0.49±0.31	0.16±0.05	0.14±0.05	0.10±0.02	0.06±0.00	0.03±0.01	0.03±0.01	0.04±0.01	0.07±0.02	0.10±0.03	0.17±0.06
	LRC1	0.15±0.02	0.10±0.01	0.10±0.01	0.10±0.01	0.10±0.01	0.09±0.01	0.09±0.01	0.09±0.01	0.09±0.01	0.10±0.01	0.10±0.01
	LRC2	0.29±0.09	0.12±0.02	0.12±0.02	0.08±0.01	0.05±0.00	0.03±0.00	0.04±0.01	0.05±0.01	0.07±0.01	0.08±0.01	0.12±0.02
200	LC1	0.75±0.14	0.38±0.08	0.35±0.05	0.28±0.02	0.22±0.02	0.16±0.02	0.17±0.02	0.20±0.03	0.26±0.04	0.33±0.03	0.40±0.06
	LC2	1.17±0.40	0.48±0.13	0.38±0.08	0.26±0.04	0.15±0.01	0.08±0.02	0.10±0.02	0.14±0.04	0.22±0.03	0.35±0.06	0.50±0.11
	LR1	0.63±0.09	0.31±0.04	0.29±0.03	0.23±0.01	0.18±0.01	0.14±0.02	0.15±0.02	0.17±0.02	0.21±0.02	0.27±0.03	0.32±0.05
	LR2	2.06±1.88	0.49±0.21	0.40±0.16	0.27±0.10	0.14±0.02	0.07±0.02	0.08±0.02	0.11±0.02	0.20±0.03	0.30±0.06	0.49±0.17
	LRC1	0.54±0.10	0.27±0.04	0.26±0.04	0.23±0.02	0.19±0.01	0.15±0.02	0.16±0.02	0.18±0.02	0.21±0.02	0.23±0.02	0.27±0.04
	LRC2	1.21±0.63	0.31±0.06	0.28±0.09	0.21±0.08	0.13±0.01	0.08±0.02	0.09±0.02	0.11±0.02	0.15±0.02	0.23±0.02	0.32±0.07
400	LC1	3.49±0.41	1.44±0.43	1.29±0.31	1.03±0.19	0.76±0.12	0.53±0.11	0.59±0.09	0.71±0.15	0.94±0.19	1.22±0.27	1.47±0.36
	LC2	3.50±1.10	1.36±0.33	1.12±0.27	0.77±0.11	0.45±0.02	0.24±0.05	0.30±0.05	0.41±0.04	0.73±0.07	1.11±0.21	1.51±0.39
	LR1	2.06±0.23	1.14±0.16	1.03±0.09	0.80±0.05	0.62±0.05	0.44±0.08	0.48±0.07	0.56±0.10	0.73±0.07	0.92±0.10	1.13±0.17
	LR2	5.58±3.76	1.83±1.02	1.49±0.84	0.92±0.39	0.47±0.07	0.21±0.06	0.26±0.07	0.36±0.07	0.69±0.15	1.24±0.48	1.93±1.04
	LRC1	2.32±0.41	0.96±0.20	0.94±0.16	0.78±0.07	0.65±0.04	0.54±0.07	0.56±0.07	0.62±0.05	0.74±0.06	0.84±0.11	0.97±0.18
	LRC2	4.35±2.96	1.18±0.60	1.03±0.55	0.72±0.38	0.43±0.06	0.28±0.06	0.31±0.06	0.37±0.07	0.53±0.06	0.85±0.22	1.20±0.52
600	LC1	5.95±0.61	3.32±0.53	2.95±0.30	2.48±0.24	1.95±0.19	1.29±0.25	1.51±0.22	1.87±0.24	2.55±0.21	3.15±0.36	3.35±0.55
	LC2	6.33±1.80	2.95±0.54	2.52±0.42	1.83±0.24	1.16±0.08	0.59±0.11	0.75±0.13	1.11±0.20	1.96±0.16	2.81±0.44	3.26±0.72
	LR1	5.53±1.40	3.02±0.49	2.71±0.35	2.06±0.21	1.58±0.19	1.05±0.19	1.19±0.23	1.50±0.24	2.15±0.22	2.73±0.46	3.16±0.57
	LR2	14.54±10.84	7.23±7.24	5.33±4.80	3.26±2.38	1.54±0.71	0.61±0.19	0.73±0.18	1.07±0.27	2.41±0.95	5.66±4.41	7.53±5.56
	LRC1	5.43±1.48	2.96±0.90	2.73±0.55	2.31±0.22	1.91±0.13	1.51±0.24	1.58±0.24	1.80±0.18	2.18±0.19	2.58±0.49	2.93±0.76
	LRC2	10.56±8.13	3.32±1.77	3.02±2.15	2.06±1.20	1.21±0.20	0.78±0.18	0.87±0.15	1.07±0.14	1.72±0.45	2.76±1.40	3.64±2.10
800	LC1	8.72±0.51	4.35±0.60	4.04±0.38	3.57±0.28	3.06±0.28	2.33±0.49	2.65±0.47	3.13±0.30	3.92±0.36	4.34±0.48	4.48±0.64
	LC2	8.46±2.01	3.93±0.70	3.39±0.50	2.56±0.28	1.84±0.13	0.92±0.17	1.30±0.23	1.87±0.18	3.28±0.45	4.12±0.84	4.54±0.84
	LR1	8.37±1.99	6.07±2.27	4.93±1.58	3.80±0.90	2.92±0.80	2.03±0.64	2.49±0.90	3.00±0.87	4.47±1.51	5.56±2.16	6.53±2.40
	LR2	20.06±16.03	8.79±7.06	6.75±5.06	4.27±2.68	2.10±0.49	0.86±0.21	1.06±0.23	1.70±0.13	3.91±1.40	6.42±3.99	9.11±6.31
	LRC1	7.64±2.19	3.73±0.94	3.55±0.72	3.08±0.38	2.43±0.19	2.03±0.27	2.16±0.27	2.45±0.16	2.90±0.28	3.39±0.56	3.75±1.00
	LRC2	12.10±8.99	4.50±2.56	3.98±2.68	3.05±2.29	1.63±0.27	1.03±0.20	1.15±0.21	1.43±0.16	2.31±0.25	3.45±1.20	4.62±2.52
1000	LC1	10.27±0.92	4.87±0.59	4.61±0.41	4.19±0.25	3.69±0.24	3.10±0.49	3.40±0.33	3.92±0.19	4.44±0.40	4.85±0.57	5.02±0.67
	LC2	9.82±2.01	4.70±0.91	4.06±0.80	3.19±0.53	2.39±0.36	1.33±0.31	1.78±0.32	2.61±0.31	4.00±0.66	4.78±0.93	5.26±1.10
	LR1	10.12±1.97	6.68±2.17	5.86±1.77	4.78±1.55	3.84±1.47	2.71±1.22	3.25±1.50	4.12±1.69	5.50±2.28	6.38±2.99	6.91±2.99
	LR2	25.42±19.58	9.04±6.03	7.82±5.75	4.69±2.55	2.43±0.50	0.96±0.22	1.30±0.23	2.20±0.22	4.72±1.65	7.26±3.62	10.11±6.81
	LRC1	8.88±2.35	4.42±0.99	4.22±0.69	3.59±0.36	2.92±0.24	2.40±0.30	2.58±0.25	2.86±0.15	3.43±0.27	3.97±0.79	4.49±1.09
	LRC2	15.52±14.56	6.51±6.13	5.59±5.55	3.34±2.04	1.83±0.31	1.09±0.21	1.25±0.20	1.65±0.12	2.80±0.72	4.24±2.10	5.71±3.94
Média total		**6.20±3.06**	**2.81±1.27**	**2.43±1.03**	**1.81±0.56**	**1.26±0.20**	**0.83±0.18**	**0.96±0.19**	**1.21±0.18**	**1.80±0.37**	**2.41±0.82**	**2.94±1.22**

somem o maior tempo de execução. Isso ocorre porque essas instâncias possuem os nós distribuídos mais uniformemente e um horizonte de trabalho grande, ou seja é mais fácil trocar uma requisição de veículo porque eles passam mais próximos um do outro e possuem mais tempo disponível durante o dia, logo o espaço de busca é maior, o que resulta em mais combinações possíveis.

Os tempos de execução das instâncias dinâmicas por sua vez tendem a ser menores, porque a medida que o veículo em movimento avança, mais requisições são visitadas, ou seja, as combinações válidas diminuem, reduzindo o espaço de busca. Isso pode ser observado na média total em relação ao grau de dinamicidade (Tabela 19), onde quanto maior o fator a ou fator ϱ menor tende a ser o tempo de execução. Por fim, é interessante notar que entre o experimento estático e o dinâmico com $a = 0.1$ e $\varrho = 1.0$, a redução do tempo chega a ser de 1/6 do tempo da estática. Portanto, é possível afirmar que o veículo em movimento reduz significativamente o tempo de execução do método como um todo.

4.5 CONSIDERAÇÕES SOBRE A ANÁLISE DOS RESULTADOS

As análises apresentadas neste capítulo foram feitas comparando os resultados das instâncias dinâmicas com os resultados das instâncias estáticas encontradas na literatura. Essa estratégia foi adotada pelo fato de não existir um *benchmark* dinâmico para o DPDPTW. Portanto, os experimentos e resultados aqui relatados apresentam três contribuições principais. A proposta de um método baseado no ALNS para resolver o DPDPTW. A construção de um *dataset* baseado no proposto por Li e Lim (2001) adaptado para realizar experimentos com DPDPTW. Resultados obtidos nos experimentos do DPDPTW que podem servir de *benchmark* para trabalhos futuros.

A condução dos experimentos aqui apresentados teve que ser restringida para lidar com o massivo número de instâncias. Embora os graus de dinamicidade selecionados para os fatores a e ϱ tenham sido amplos, ainda assim se faz necessário analisar outras combinações de valores para essas variáveis para aumentar a abrangência dos resultados. Por exemplo, ao invés de fixar o valor de um dos fatores e variar apenas o outro, poderia ser testada uma combinação aleatória de valores para esses dois fator a fim de analisar o comportamento da solução.

Durante a análise dos resultados foi possível observar alguns padrões emergentes. O principal deles é o comportamento do objetivo NV em relação ao objetivo TC. Foi observado que para o mesmo problema as soluções com menor NV tendem a apresentar maior TC, uma relação inversamente proporcional entre os objetivos. Esse comportamento pode ser melhor entendido ao se analisar os resultados em função da distribuição geográfica dos nós. Quanto maior a concentração de nós em *clusters*

maior tende a ser o número de veículos e menor tende a ser o custo total da solução. Isso ocorre porque é menos eficiente um veículo atender dois *clusters*. Portanto, o mecanismo que reduz o número de veículos acaba, por consequência, aumentando o custo total da solução.

Outro ponto relevante é que o tipo de distribuição e o tamanho do horizonte de tempo do problema influenciam no tamanho do espaço de busca. Por exemplo, quanto mais clusterizados forem os nós e quanto menor for o horizonte do dia de trabalho, menor tende a ser o espaço de busca, tornando assim mais fácil a otimização. Isso ocorre porque nesse tipo de cenário menos combinações são possíveis. Todos esses comportamentos são relevantes porque permitem entender como o algoritmo se comporta em função das características do problema. Com base na análise dos resultados, foi possível perceber que o **ALNS-DPDP** possuí limitações que poderiam ser mitigadas se ele utilizasse as características do problema para definir melhores estratégias de otimização. Por exemplo, para problemas mais clusterizados a heurística de inserção poderia inserir as requisições com base em um fator de proximidade entre os nós. Esse fator poderia ser calculado analisando diversos cenários de pior e melhor caso.

Na realização dos experimentos com o *benchmark* foi possível constatar que a média dos melhores resultados apresentados pelo **ALNS-DPDP** em relação a literatura obteve uma diferença de 1.35% para o NV e -2.07% para o TC. Esses resultados são muito próximos do estado da arte do PDPTW. Vale destacar que os melhores resultados do estado da arte foram alcançados por vários algoritmos diferentes. Além disso, a média e o desvio padrão são considerados baixos, 2.27$\pm$0.99% para NV e -2.36$\pm$1.95% para TC, o que indica uma boa capacidade para encontrar soluções robustas recorrentemente. Com base nesses resultados, é possível considerar o **ALNS-DPDP** como um algoritmo competitivo em relação aos apresentados na literatura.

Embora na média os resultados sejam considerados bons, os casos de maior complexidade apresentam lacunas para melhoria. Por exemplo, quando o problema envolve uma distribuição semi-clusterizada dos nós o desempenho é degradado (4.88% para NV e -4.12% para TC). Isso evidência o aumento da dificuldade de otimização quando o espaço de busca é maior. O **ALNS-DPDP** proposto foi concebido usando os operadores da versão original do ALNS de Pisinger e Ropke (2007). Entretanto, na literatura existem vários outros operadores que devem ser considerados, como a remoção com base no critério histórico (LUTZ, 2014). Outra lacuna em aberto é a parametrização, os valores aqui utilizados não passaram por nenhum processo de *tunning* ou adaptação. Portanto, com a aplicação de processos de *tunning* e um controle maior sobre os valores dos parâmetros acredita-se que é possível melhorar a qualidade dos resultados. Por exemplo, aumentar e reduzir o ruído dos operadores de inserção em

função da temperatura.

Em relação a análise das instâncias dinâmicas, os experimentos permitiram assimilar como a dinamicidade influencia no desempenho das soluções. De forma geral, o GAP da literatura permitiu perceber que a maior dificuldade da otimização é lidar com graus altos de urgência, onde para $a \geq 0.75$ as soluções tiveram um GAP de até 108% no custo de NV e TC. Nesses cenários, mesmo aumentando-se a quantidade de requisições conhecidas *a priori*, não houveram ganhos expressivos nos resultados da otimização. Não obstante, nos *benchmarks* onde a dinamicidade era mais extrema, a frota disponível não foi suficiente para atender todos os clientes.

Uma das limitações do **ALNS-DPDP** em lidar com urgências altas está na visão de longo prazo da otimização. Por exemplo, se no início da otimização muitas requisições são anunciadas em um extremo da área de roteamento, então o **ALNS-DPDP** irá direcionar o roteamento para essa região sem considerar que no futuro próximo requisições urgentes podem ser anunciadas no outro extremo da área de roteamento, ou seja, quando as requisições são apresentadas ao problema, o algoritmo busca modelar o problema em mãos para otimizar o custo sem se preparar para possíveis eventos futuros. Esse efeito pode ser percebido nas análises das Figura 25 e Figura 26 feitas na Seção 4.3.1. Nesse sentido, acredita-se que os veículos deveriam ser mantidos bem distribuídos no tempo e espaço (coordenadas geográficas) para permitir a geração de soluções mais robustas às interferências da dinamicidade. Por exemplo, mesmo que a maior parte das requisições estejam em uma extremidade da área de roteamento, alguns veículos poderiam considerar aumentar o custo e passar perto do outro extremo porque, no caso de uma urgência, haveria veículos nas proximidades para fazer o atendimento.

Por fim, outra limitação do algoritmo se refere ao uso do tempo de espera. No modelo atual, se um veículo chegar no cliente antes do inicio do atendimento, ele espera para fazer coleta/entrega da carga. Para problemas estáticos o tempo de espera não é um problema, porque ele não influencia no resultado final. Entretanto, para problemas dinâmicos isso é uma limitação, porque, como as decisões são tomadas com base no estado de atendimento das requisições, adiar o máximo possível a troca de estado permite ao algoritmo ter mais tempo para fazer as computações. Por exemplo, se um veículo já sabe o tempo que terá de esperar no próximo cliente e, se antes dele partir para esse cliente ele esperasse no cliente atual, então o algoritmo atrasaria a troca de estado do próximo cliente de “Em espera” para “Em transição” o máximo possível, isso permitiria ao **ALNS-DPDP** ter mais tempo para otimizar o próximo cliente. Uma abordagem para essa limitação já foi proposta em (PUREZA; LAPORTE, 2008), ela baseada em uma política de uso do tempo de espera e uma estratégia de *buffering* de requisições dinâmicas. A implementação dessas políticas não fizeram parte

do escopo deste trabalho em tela.

5 CONCLUSÕES

O advento do modelo de VRP chamado de *last mile* introduziu algumas dificuldades aos processos logísticos. Dentre eles, o principal diz respeito a possibilidade dos clientes solicitarem coletas e entregas após o início do dia de trabalho. Esse tipo de solicitação é chamada de requisição dinâmica. Atualmente, a grande maioria dos métodos desenvolvidos para problemas de VRP são aplicados a cenários estáticos, onde a otimização do roteamento é feita antes de iniciar a operação diária dos veículos. Poucos trabalhos consideram cenários dinâmicos que permitem aos clientes solicitarem atendimento depois que a operação é iniciada. Portanto, para contribuir na solução dos problemas de *last mile*, este trabalho propõem o desenvolvimento de um modelo de otimização capaz de lidar com a entrada de requisições dos clientes não conhecidas antes do início do dia de trabalho.

Na literatura, o problema abordado neste trabalho se enquadra como um problema de coleta e entrega com janelas de tempo e requisições dinâmicas, o chamado DPDPTW. O problema DPDPTW endereçado neste trabalho tem como função objetivo a minimização de duas variáveis, o número de veículos e o custo total de viagem dos veículos. Ainda, foram definidas as seguintes restrições a serem respeitadas: precedência entre coleta e entrega de uma mesma encomenda, limite de capacidade de carga máxima dos veículos, janela de tempo de chegada ao cliente, tempo de serviço no cliente e a entrada de requisições dinâmicas durante o dia de trabalho.

Para resolver o problema em questão foi proposta a utilização da heurística ALNS. Entretanto, foi necessário conceber um modelo baseado no ALNS que fosse capaz de lidar com o tratamento de requisições dinâmicas. Antes de desenvolver o modelo proposto, buscou-se entender quais características deveriam ser consideradas em um modelo dinâmico. Nessa etapa, foram encontrados os conceitos de restrições temporais e dinâmicas do problema, restrições essas que estão relacionadas ao estado de atendimento dos clientes e a validade das requisições dinâmicas.

No método proposto as restrições temporais indicam o estado de atendimento das requisições dos clientes e visam limitar as otimizações que podem ser feitas pelo ALNS. Foram definidos três estados de atendimento: “Em espera” (*Idle*), “Em transição” (*Transition*) e “Atendido” (*Committed*). As requisições com o estado “Em espera” indicam clientes que podem ser otimizados pelo ALNS, enquanto que requisições com estado “Em transição” ou “Atendido” indicam clientes que foram ou estão sendo atendidos e devem ser ignorados durante a otimização do ALNS. Além disso, para atender as limitações impostas pelos estados de atendimento, foram propostos novos operadores

de inserção, remoção e busca local do ALNS capazes de lidar com essa dinâmica.

A versão proposta do ALNS para o problema DPDPTW é diferente da versão aplicada ao problema PDPTW proposta na literatura por intercalar a otimização dos dois objetivos do problema ao mesmo tempo. No ALNS estático, a minimização do número de veículos precede a minimização do custo total, isso torna o método ineficiente para problemas dinâmicos porque o tempo para trocar de objetivo é muito longo. Em problemas dinâmicos, as soluções devem ser capazes de se adaptar rapidamente as novas condições, isso requer do algoritmo um tempo de resposta menor. Para isso foram desenvolvidos dois ALNS's, $ALNS_{NV}$ e $ALNS_{TC}$ (um para cada objetivo), e um algoritmo *solver*, responsável por intercalar a execução entre ambos os ALNS's. O algoritmo final foi chamado de ALNS-DPDP.

O algoritmo proposto para o *solver*, além de intercalar a execução dos dois ALNS's, também é responsável por gerenciar a dinamicidade do DPDPTW. Esse desacoplamento tornou o método mais simples, pois retirou a lógica de controle do estado das requisições e a lógica da entrada de novas requisições dos algoritmos ALNS. A ideia do *solver* é gerenciar todo o dia de trabalho, acompanhando as visitas realizadas pelos veículos, aceitando e inserindo as requisições dinâmicas no problema, invocando as execuções do $ALNS_{NV}$ e $ALNS_{TC}$, e decidindo como e quando a solução global pode ser atualizada.

Durante a fase de experimentos foi aplicado um método para converter os cenários estáticos (PDPTW) para o cenário dinâmico (DPDPTW). As instâncias de teste foram separadas em dois grupos, o grupo $P1$ considerava variação da dinamicidade referente ao grau de urgência e o grupo $P2$ considerava variação da dinamicidade referente ao grau de requisições dinâmicas. O grau de urgência define quanto tempo disponível um veículo tem para realizar o atendimento após a solicitação do cliente. Nesse cenário, se um cliente ligar muito próximo do tempo limite de atendimento, a urgência é alta e a otimização é mais difícil. Por outro lado, o grau de requisições dinâmicas indica a porcentagem de requisições que serão anunciadas durante a operação do dia de trabalho. Quanto maior for o grau de requisições dinâmicas, mais difícil se torna encontrar boas soluções.

A primeira análise de resultados comparou o ALNS-DPDP em relação às soluções da literatura. O método proposto apresentou um desvio (GAP) baixo em relação as soluções da literatura, a média dos *benchmarks* foi de 2.27% no número de veículos e -2.36% no custo total. Enquanto que em relação a melhor solução encontrada o desvio médio foi de 1.35% para o número de veículos e -2.07% para o custo total. Esses desvios são considerados adequados pelo fato das melhores soluções da literatura não terem sido encontradas pelo mesmo algoritmo, o que sugere que não existe um algoritmo soberano para todas as instâncias.

Durante a análise dos resultados pode-se perceber que o grau de urgência gera uma maior dificuldade de otimização do que o grau de requisições dinâmicas. Um fator de grande influência é o horário de início da operação do veículo. Basicamente, quando um veículo recebe uma rota e saí do depósito para iniciar o atendimento ele fixa a visita ao referido cliente e por esse motivo não pode mais ser removido, isso dificulta a minimização de veículos. Um ponto que permite mitigar essa dificuldade é a adoção do tempo de preparação (*setup time*), que é um tempo mínimo requisitado pela empresa antes de despachar um veículo para a operação. Portanto, o modelo proposto utiliza o *setup time* como um artifício para tentar não adicionar novos veículos. Neste trabalho, o *setup time* foi definido empiricamente como 90 unidades de tempo.

Na data de realização desse trabalho, não se conhece um *benchmarks* dinâmico da literatura que permita uma comparação dos resultados. Portanto, foi adotado a estratégia de comparação dos resultados com os resultados encontrados na literatura do *benchmark* estático. A comparação permitiu observar que existe um aumento de 108% no uso de veículos e um aumento de 72% no custo total para o cenário dinâmico mais complexo, onde a urgência é de 100% e o grau de requisições dinâmicas é de 100%. Outro fator importante foi a insuficiência de veículos para os cenários dinâmicos mais complexos, onde alguns casos necessitaram mais veículos do que a capacidade da frota. Com base nisso, entende-se que para cenários dinâmicos muito complexos existe a chance do modelo não conseguir otimizar o atendimento de todos os clientes com a frota disponível.

Por fim, como o ALNS-DPDP apresentou resultados satisfatórios em relação ao *benchmark* do PDPTW e também se mostrou capaz de lidar com o DPDPTW, com exceção de alguns graus de dinamicidade mais elevados onde houve a quebra da factibilidade da frota, é possível inferir que o modelo proposto é aplicável a problemas dinâmicos. Outra contribuição importante deste trabalho é que os resultados apresentados podem servir de base comparativa (*benchmark*) para o desenvolvimento de propostas futuras em DPDPTW.

5.0.1 Trabalhos Futuros

Como sequência ao trabalho apresentado, os seguintes itens ficam em aberto para trabalhos futuros:

- Analisar o impacto do *setup time* no ALNS-DPDP e propor uma forma de cálculo que permita a definição automática desse parâmetro. No modelo proposto esse parâmetro foi definido de forma empírica, porém observou-se que ele é determinante para a qualidade das soluções geradas;

- Adaptar o modelo proposto para que ao finalizar o atendimento de um cliente, e antes de partir para o próximo, o veículo possa usar tempo de espera do próximo cliente no cliente atual, conforme proposto por (PUREZA; LAPORTE, 2008);
- Adaptar o modelo proposto para que ele distribua o roteamento dos veículos de forma mais abrangente pela região do mapa. A ideia é propor um roteamento que não sofra quebras de continuidade abruptas nas rotas dos veículos com a anunciação de requisições dinâmicas;
- Estudar a aplicação de um controle de temperatura mais eficiente para o ALNS-DPDP, onde não seja necessários tantos gatilhos de recálculo;
- Propor a aplicação da auto-parametrização do ALNS-DPDP. Essa necessidade visa reduzir a complexidade de parametrização visto que o ALNS por si contém muitos parâmetros.

REFERÊNCIAS

ADEWUMI, A. O.; ADELEKE, O. J. 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.

AZIEZ, I.; CôTé, J.-F.; COELHO, L. A branch-and-cut algorithm for the multi-pickup and delivery problem with time windows. fev. 2019.

BALDACCI, R.; BARTOLINI, E.; MINGOZZI, A. An exact algorithm for the pickup and delivery problem with time windows. **Operations Research**, INFORMS, v. 59, n. 2, p. 414–426, 2011. ISSN 0030364X, 15265463.

BANKER, S. **The UPS-Workhorse Group Deal Highlights Advances In Last-Mile Routing**. Forbes Magazine, 2018. Disponível em: <https://www.forbes.com/sites/stevebanker/2018/02/23/the-upsworkhorse-group-deal-highlights-advances-in-last-mile-routing>.

BENT, R.; HENTENRYCK, P. V. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. **Computers & Operations Research**, Elsevier BV, v. 33, n. 4, p. 875–893, apr 2006.

BERBEGLIA, G. et al. Rejoinder on: Static pickup and delivery problems: a classification scheme and survey. **TOP**, Springer Nature, v. 15, n. 1, p. 45–47, abr. 2007.

BRAEKERS, K.; RAMAEKERS, K.; Van Nieuwenhuyse, I. 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.

CARABETTI, E. G. et al. An application of the ant colony system metaheuristic to the vehicle routing problem with pickup and delivery and time windows. In: **2010 Eleventh Brazilian Symposium on Neural Networks**. [S.l.]: IEEE, 2010.

CHEN, S. et al. An adaptive large neighborhood search heuristic for dynamic vehicle routing problems. **Computers and Electrical Engineering**, Elsevier Ltd, v. 67, p. 596–607, 2018. ISSN 00457906.

CORDEAU, J. F. et al. Chapter 6 Vehicle Routing. **Handbooks in Operations Research and Management Science**, v. 14, n. C, p. 367–428, 2007. ISSN 09270507.

CORMEN, T. H. e. a. **Introduction to algorithms**. [S.l.]: MIT Press, 2009.

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.

DRIDI, I. H.; ALAIA, E. B.; BORNE, P. Heuristic approach for the optimization of the dynamic multi-vehicle pickup and delivery problem with time windows. In: **2015 International Conference on Industrial Engineering and Systems Management (IESM)**. [S.l.]: IEEE, 2015.

DUMAS, Y.; DESROSIERS, J.; SOUMIS, F. The pickup and delivery problem with time windows. **European Journal of Operational Research**, Elsevier BV, v. 54, n. 1, p. 7–22, set. 1991.

EDEN, A. H. Three paradigms of computer science. **Minds and machines**, Springer, v. 17, n. 2, p. 135–167, 2007.

HASLE, G.; KLOSTER, O. **Industrial Vehicle Routing**. [S.l.]: Springer Berlin Heidelberg, 2007. 397–435 p.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, American Association for the Advancement of Science (AAAS), v. 220, n. 4598, p. 671–680, maio 1983.

LARSEN, A. **The dynamic vehicle routing problem**. Tese (Doutorado), 12 2000.

Li, H.; Lim, A. A metaheuristic for the pickup and delivery problem with time windows. In: **Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001**. [S.l.: s.n.], 2001. p. 160–167. ISSN null.

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

MARCONI, M. d. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. [S.l.]: 5. ed.-São Paulo: Atlas, 2003.

MCCREA, B. **2019 Transportation Management Systems (TMS) Market Update: Keeping pace with the times**. 2019a. Disponível em: <https://www.logisticsmgmt.com/article/2019_transportation_management_systems_tms_market_update_keeping_pace_with>.

MCCREA, B. **Transportation Best Practices/Trends: Make me a match**. 2019b. Disponível em: <https://www.logisticsmgmt.com/article/transportation_best_practices_trends_make_me_a_match/motorfreight>.

MORIN, E.; MOIGNE, J.-L. L. A inteligência da complexidade. Ed. Fundação Peirópolis, 2000.

NACCACHE, S.; CÔTÉ, J.-F.; COELHO, L. C. The multi-pickup and delivery problem with time windows. **European Journal of Operational Research**, Elsevier BV, v. 269, n. 1, p. 353–362, ago. 2018.

NECULA, R.; BREABAN, M.; RASCHIP, M. Tackling dynamic vehicle routing problem with time windows by means of ant colony system. In: **2017 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.]: IEEE, 2017.

PANKRATZ, G. Dynamic vehicle routing by means of a genetic algorithm. **International Journal of Physical Distribution & Logistics Management**, Emerald, v. 35, n. 5, p. 362–383, jun 2005.

PARRAGH, S. N.; DOERNER, K. F.; HARTL, R. F. A survey on pickup and delivery problems. **Journal für Betriebswirtschaft**, Springer Science and Business Media LLC, v. 58, n. 2, p. 81–117, maio 2008.

PEREIRA, V. G. **Análise de Meta-heurísticas para o Problema de Roteamento de Veículos com Várias Restrições**. Dissertação (Mestrado), Joinville, SC, 11 2019.

PISINGER, D.; ROPKE, S. A general heuristic for vehicle routing problems. **Computers & Operations Research**, Elsevier BV, v. 34, n. 8, p. 2403–2435, ago. 2007.

PSARAFTIS, H. N. Dynamic vehicle routing: Status and prospects. **Annals of Operations Research**, Springer Science and Business Media LLC, v. 61, n. 1, p. 143–164, dez. 1995.

PUREZA, V.; LAPORTE, G. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. **Infor**, v. 46, n. 3, p. 165–175, 2008. ISSN 03155986.

ROPKE, S. **Heuristic and exact algorithms for vehicle routing problems**. Tese (Doutorado), 01 2005.

ROPKE, S.; PISINGER, D. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. **Transportation Science**, v. 40, n. 4, p. 455–472, 2006.

SAVELSBERGH, M. W. P. The vehicle routing problem with time windows: Minimizing route duration. **ORSA Journal on Computing**, Institute for Operations Research and the Management Sciences (INFORMS), v. 4, n. 2, p. 146–154, maio 1992.

SCHMITT, J. P.; BALDO, F.; PARPINELLI, R. S. 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)**. [S.l.]: IEEE, 2018.

SHAW, P. Using constraint programming and local search methods to solve vehicle routing problems. In: **Principles and Practice of Constraint Programming — CP98**. [S.l.]: Springer Berlin Heidelberg, 1998. p. 417–431.

SINTEF. **PDPTW - Li & Lim benchmark**. 2019. Disponível em: <https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/>.

SOLOMON, M. 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.

TALBI. **Metaheuristics : from design to implementation**. Hoboken, N.J: John Wiley & Sons, 2009. ISBN 9780470496909.

TCHOUPO, M. N. et al. Ant colony optimization algorithm for pickup and delivery problem with time windows. In: **Springer Proceedings in Mathematics & Statistics**. [S.l.]: Springer International Publishing, 2017. p. 181–191.

TRIPP, D. Pesquisa-ação: uma introdução metodológica. **Educação e pesquisa**, SciELO Brasil, v. 31, n. 3, 2005.

VEEN, B. van et al. Ant colony algorithms for the dynamic vehicle routing problem with time windows. In: **Natural and Artificial Computation in Engineering and Medical Applications**. [S.l.]: Springer Berlin Heidelberg, 2013. p. 1–10.

WAZLAWICK, R. **Metodologia de pesquisa para ciência da computação**. [S.l.]: Elsevier Brasil, 2017. v. 2.