UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

IGOR DE ALMEIDA MALHEIROS BARBOSA

UM ALGORITMO HÍBRIDO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS DO TIPO *DIAL-A-RIDE* COM FROTA HETEROGÊNEA E MÚLTIPLOS DEPÓSITOS

João Pessoa
2020

IGOR DE ALMEIDA MALHEIROS BARBOSA

UM ALGORITMO HÍBRIDO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS DO TIPO *DIAL-A-RIDE* COM FROTA HETEROGÊNEA E MÚLTIPLOS DEPÓSITOS

Dissertação submetida ao Programa de Pós-Graduação em Informática do Centro de Informática da Universidade Federal da Paraíba, como requisito parcial para obtenção do Grau de Mestre em Informática.

Orientador: Prof. Dr. Anand Subramanian

Co-orientador: Prof. Dr. Teobaldo Leite Bulhões Júnior

João Pessoa

2020

Catalogação na publicação
Seção de Catalogação e Classificação

B238a Barbosa, Igor de Almeida Malheiros.
Um algoritmo híbrido para o problema de roteamento de veículos do tipo dial-a-ride com frota heterogênea e múltiplos depósitos / Igor de Almeida Malheiros Barbosa. - João Pessoa, 2020.
77 f. : il.

Orientação: Anand Subramanian.
Coorientação: Teobaldo Leite Bulhões Júnior.
Dissertação (Mestrado) - UFPB/CI.

1. Informática. 2. Dial-a-ride. 3. Iterated local search. 4. Roteamento de veículos. 5. Meta-heurística. I. Subramanian, Anand. II. Bulhões Júnior, Teobaldo Leite. III. Título.

UFPB/BC CDU 004(043)

Elaborado por Larissa Silva Oliveira de Mesquita - CRB-15/746

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de Igor de Almeida Malheiros Barbosa, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 30 de julho de 2020.

Aos trinta dias do mês de julho do ano de dois mil e vinte, às dezesseis horas e trinta minutos, por meio de videoconferência, reuniram-se os membros da Banca Examinadora constituída para julgar o trabalho do sr. Igor de Almeida Malheiros Barbosa, vinculado a esta Universidade sob a matrícula nº 20181000742, candidato ao grau de Mestre em Informática, na área de "Sistemas de Computação", na linha de pesquisa "Computação Distribuída", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Anand Subramanian (PPGI-UFPB) Orientador e Presidente da Banca, Gilberto Farias de Sousa Filho (PPGI-UFPB), Examinador Interno, Teobaldo Leite Bulhões Junior (UFPB), Examinador Externo ao Programa, Raphael Harry Frederico Ribeiro Kramer (UFPE), Examinador Externo à Instituição. Dando início aos trabalhos, o Presidente da Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse a exposição oral do trabalho de dissertação intitulado: "Um Algoritmo Híbrido para o Problema de Roteamento de Veículos do Tipo Dial-A-Ride com Frota Heterogênea e Múltiplos Depósitos". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Ruy Alberto Pisani Altafim, Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 30 de julho de 2020.

Prof. Dr. Ruy Alberto Pisani Altafim

Prof. Anand Subramanian
Orientador (PPGI-UFPB)

Prof. Gilberto Farias de Sousa Filho
Examinador Interno (PPGI-UFPB)

Prof. Teobaldo Leite Bulhões Junior
Examinador Externo ao Programa (UFPB)

Prof. Raphael Harry Frederico Ribeiro Kramer
Examinador Externo à Instituição (UFPB)

"*Science literacy is the artery through which the solutions of tomorrow's problems flow.*"

Neil deGrasse Tyson

AGRADECIMENTOS

Muitas pessoas me influenciaram durante a realização deste trabalho. Por isso faço um agradecimento:

- Aos meus pais Francinete e João Batista por todo apoio, dedicação e amor que recebi em toda minha. Além de sempre me ensinarem que educação e ciência são os melhores caminhos para um mundo melhor.
- Aos meus irmãos Yuri e Dimitri que sempre foram exemplos de dedicação, inteligência, bondade e humildade para mim. Vocês me influenciam profundamente.
- À Tayná que foi uma companheira essencial ao longo do mestrado, sendo capaz de dar suporte, motivação e dizer palavras de conforto nas horas certas. Agradeço profundamente ao seu apoio.
- À toda minha família, meus tios, primos e especialmente aos meus sobrinhos, Laura e Daniel, que alegraram a casa durante os finais de semana.
- Aos meus amigos que me apoiaram durante essa jornada, Bergh, Wendell, Nackphy, Adalberto, Maria, Amanda, Jéssica, Moana, Júlio, Ian, Eduardo, Hugo, Isadora e Rebeca. Um destaque aos amigos Iago e Andréa que eu sempre mantive maior proximidade, agradeço por todo companheirismo.
- Os amigos e companheiros de pesquisa Bruno Passeti, que me ajudou durante este trabalho, e Rodrigo Ramalho que tive a oportunidade de trabalhar diretamente e presenciar sua evolução como desenvolvedor e pesquisador.
- Aos colegas Luan, Felipe, Carlos, Rafael, Iderval, João, Eduardo e Rubens que tornaram o LASER um ambiente agradável para trabalhar.

- Ao Prof. Bruno Bruck que tive a oportunidade de trocar experiências durante o período de estágio docência na disciplina de análise e projeto de algoritmos.

- Ao Prof. Kris Braekers que foi solicito e essencial para realização deste trabalho, tirando dúvidas e ajudando a resolver problemas que foram descobertos ao longo do caminho.

- Ao Prof. Teobaldo que tive a oportunidade de trabalhar novamente anos depois. Sempre trazendo boas soluções e correções precisas. Além de um amigo do período da graduação, Teobaldo foi um excelente orientador durante o mestrado.

- Ao Prof. Anand por toda dedicação, paciência, motivação, educação e confiança como orientador. Além das excelentes conversas sobre diferentes assuntos que pude desfrutar de Anand como amigo. A amizade criada foi fundamental para realização deste trabalho.

RESUMO

Os problemas de roteamento de veículos emergem em inúmeras situações práticas em logística de transporte. Dentre elas, pode-se destacar o transporte de pessoas entre localizações de origem e destino, esse problema é conhecido como *dial-a-ride problem* (DARP). O DARP consiste em construir rotas de custo mínimo que atendam requisições de coleta e entrega, obedecendo as restrições de capacidade, janela de tempo, máxima duração de rota e tempo máximo de viagem. Este trabalho propõe um algoritmo híbrido para resolver variantes do DARP com demanda e frota heterogêneas, além de veículos que começam suas rotas em diferentes depósitos. O método combina a meta-heurística *iterated local search* com um procedimento exato para resolver o problema de particionamento de conjuntos. Além disso, diversos mecanismos foram implementados para acelerar os procedimentos de buscas locais. Experimentos computacionais foram realizados em instâncias da literatura com o objetivo de avaliar diversos componentes do algoritmo, além de comparar sua performance com o melhor método existente. Os resultados obtidos sugerem que o algoritmo proposto superou métodos do estado da arte produzindo resultados de alta qualidade, inclusive encontrando sete novas melhores soluções, em tempos computacionais competitivos.

Palavras-chave: Meta-heurística. *Dial-a-ride*. *Iterated local search*. Roteamento de veículos.

ABSTRACT

Vehicle routing problems arise in many practical situations in the context of transportation logistics. Among them, we can highlight the problem of transporting customers from origin to destination locations, this problem is known as the dial-a-ride problem (DARP). The DARP consists of designing least-cost routes to serve pickup-and-delivery requests, while meeting capacity, time window, maximum route duration, and maximum ride time constraints. This work proposes a hybrid algorithm to solve DARP variants where both the demands and vehicle fleet are heterogeneous, and the vehicles start their routes from multiple depots. The method combines the iterated local search metaheuristic with an exact procedure based on a set partitioning approach. In addition, several procedures were implemented to speedup the local search phase. Extensive computational experiments were conducted on benchmark instances in order to evaluate the impact of the different components of the algorithm, and to compare its performance with the best existing method. The results obtained suggest that the proposed algorithm outperforms the state-of-art method, producing high quality solutions, even improving seven best known results, in a very competitive runtime.

Keywords: Metaheuristics. Dial-a-ride. Iterated local search. Vehicle routing.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO

ADSs	*Auxiliary Data Structures*
AG	Algoritmos Genéticos
ALNS	*Adaptative Large Neighborhood Search*
B&C	*Branch-and-Cut*
B&P&C	*Branch-and-Price-and-Cut*
BA-DA	*Bees Algorithm with Deterministic Annealing*
BA-SA	*Bees Algorithm with Simulated Annealing*
BKS	*Best Known Solution*
CP	*Constraint Programming*
DA	*Deterministic Annealing*
DARP	*Dial-A-Ride Problem*
DARP-EV	*Dial-A-Ride Problem with Electric Vehicle*
FTS	*Forward Time Slack*
GC	Geração de Colunas
HDARP	*Heterogeneous Dial-A-Ride Problem*
ILS	*Iterated Local Search*
LB	*Lower Bound*
LNS	*Large Neighborhood Search*
MDHDARP	*Multi-Depot Heterogeneous Dial-A-Ride Problem*
MDMTHDARP	*Multi-Depot Multi-Trip Heterogeneous Dial-A-Ride Problem*
MIP	*Mixed-Integer Programming*
MILP	*Mixed-Integer Linear Programming*
MF-HDARP	*Mixed Fleet Heterogeneous Dial-A-Ride Problem*
OC	Otimização Combinatória

GLOSSÁRIO

PCV	Problema do Caixeiro Viajante
PD	Programação Dinâmica
PDP	*Pickup and Delivery Problem*
PO	Pesquisa Operacional
PRV	Problema de Roteamento de Veículos
RVND	*Randomized Variable Neighborhood Descent*
SCP	*Set Covering Problem*
SP	*Set Partitioning*
TS	*Tabu Search*
TSP	*Traveling Salesman Problem*
TSPTW	*Traveling Salesman Problem with Time Windows*
VND	*Variable Neighborhood Descent*
VNS	*Variable Neighborhood Search*

Capítulo 1

Introdução

1.1 Definição do tema

Muitos problemas práticos que aparecem em logística de transporte, como por exemplo distribuição de mercadorias, coleta de lixo, entrega de jornais e transporte privado de pessoas, podem ser modelados por uma classe de problemas conhecida como *Problema do Roteamento de Veículos* (PRV). Como solução de um PRV, procura-se um conjunto de rotas de veículos com custo mínimo e que atendam um certo número de clientes distribuídos geograficamente. Uma rota deve começar em um ponto geralmente denominado de depósito, visitar uma sequência de clientes e retorna para sua origem, o depósito (Toth e Vigo, 2002).

Dentre as diversas variantes dessa classe de problemas, uma aplicação prática aparece em serviços de transporte de pessoas. Usuários que usam esses serviços requisitam que transportes façam coleta e entrega de pessoas em determinados lugares, com a possibilidade de compartilhamento de viagem com outros usuários. Além disso, são estabelecidas restrições de duração da viagem e uma janela de tempo para a saída e chegada do cliente. Esse tipo de aplicação vem se tornando cada vez mais popular nos últimos anos (Henao, 2017).

Designar quais veículos devem atender quais requisições, em qual ordem, por quais caminhos e obedecendo restrições de capacidade, janela de tempo e duração máxima

de viagem para cada cliente pode se tornar uma tarefa muito complicada à medida que as demandas crescem. Além disso, tal designação deve-se preocupar em minimizar os custos de transporte. O problema que resolve esse tipo de tarefa é conhecido como *Dial-A-Ride Problem* (DARP) (Cordeau e Laporte, 2003a).

Problemas como o DARP e PRV podem ser modelados como um problema de Otimização Combinatória (OC), que significa que as decisões para uma solução são feitas sobre valores discretos. Ambos são generalizações de um problema clássico, o *problema do caixeiro viajante* (do inglês, *Traveling Salesman Problem* – TSP).

Diversos problemas de OC pertencem à classe de problemas $\mathcal{NP}$-Difícil (Neumann e Witt, 2010). Para resolver problemas dessa classe, duas abordagens são geralmente utilizadas: via algoritmos exatos, que procuram a solução ótima para o problema, porém eles podem exigir tempos computacionais impraticáveis; e via algoritmos heurísticos, que apesar de não fornecerem um certificado de otimalidade, são capazes de gerar soluções aproximadas às ótimas, muitas vezes ótima e com tempos computacionais mais viáveis.

Este trabalho apresenta uma abordagem híbrida, isto é, utilizando estratégias heurísticas combinadas com métodos exatos, para problemas do tipo DARP, podendo possuir múltiplos depósitos e/ou demandas heterogêneas.

1.2 Motivação

Nos últimos anos, preocupações com o meio ambiente e com o congestionamento urbano vêm crescendo cada vez mais. Se, em uma cidade, cada pessoa sair de casa utilizando seu próprio veículo, problemas de tráfego e estacionamento podem ser causados. Além disso, mais veículos nas ruas aumentam significativamente a quantidade de dióxido de carbono na atmosfera. Por isso, sistemas de compartilhamento de viagem vêm ganhando mais espaço como alternativa para amenizar tais problemas (Henao, 2017).

O DARP é comumente motivado por aplicações práticas, cada variação com suas restrições e objetivos que descrevem uma situação real (Ho et al., 2018). Uma das mais conhecidas aplicações é para o transporte de pessoas idosas ou deficientes, no qual as transportadoras devem buscar essas pessoas em casa e deixá-las em seus destinos, tendo

como objetivo minimizar os custos de transporte (Madsen et al., 1995).

Uma outra aplicação conhecida é usada nos sistemas de saúde, em que clientes que precisam se deslocar entre hospitais, ou de suas casas para centros de tratamento, ou ainda de locais de acidentes para hospitais. Nessa variação, os clientes possuem uma demanda heterogênea (*Heterogeneous DARP* – HDARP), isto é, a forma pela qual um cliente é transportado varia. O paciente pode necessitar, por exemplo, de assentos comuns, macas ou cadeiras de rodas (Parragh, 2011).

Incorporando restrições de janela de tempo e de tempo de viagem, o DARP pode ser visto como uma generalização do *problema de coleta e entrega* (do inglês, *Pickup and Delivery Problem* – PDP). Encontrar uma solução viável para o DARP é $\mathcal{NP}$-Difícil, uma vez que ele também é uma generalização do PCV com janela de tempo (*TSP with Time Windows* – TSPTW) (Savelsbergh, 1985; Cordeau, 2006).

Uma abordagem por algoritmos exatos para resolver o DARP pode necessitar de tempos computacionais longos em instâncias grandes, sendo intratáveis para fins práticos. Os algoritmos heurísticos surgem como uma alternativa viável para obter soluções próximas ou iguais às ótimas em tempos computacionais aceitáveis. Embora muitos avanços tenham sido feitos com abordagens exatas, os métodos exatos propostos na literatura fornecem valores ótimos para instâncias de pequeno e médio porte, mas, por executarem em tempos exponenciais, podem se tornar intratáveis instâncias de grande porte. Nenhum algoritmo da literatura resolve de maneira consistente instâncias acima de 80 requisições para o HDARP ou sua variante com múltiplos depósitos (Braekers et al., 2014). Dessa forma, os métodos heurísticos se apresentam como mais adequados para lidar com instâncias de médio e grande porte. Uma vez que eles podem encontrar soluções ótimas ou de alta qualidade de maneira eficiente mesmo em instâncias grandes.

As meta-heurísticas são procedimentos de mais alto nível de abstração usadas de forma eficiente para resolver muitos problemas de OC (Boussaïd et al., 2013). Portanto, meta-heurísticas podem ser vistas como estratégias que servem para orientar a utilização de heurísticas desenvolvidas para um problema específico (Talbi, 2009).

Uma outra abordagem promissora para resolver problemas de OC é combinar meta-heurísticas com métodos exatos, ou seja, criando um algoritmo híbrido. Os métodos híbridos vêm se popularizando cada vez mais principalmente pelos bons resultados

obtidos para resolver PRVs e algumas de suas variações (Subramanian et al., 2013).

Segundo Lourenco et al. (2002), é preferível que as meta-heurísticas sejam simples, eficazes e robustas. Atendendo a todos esses requisitos, este trabalho apresenta uma abordagem híbrida baseada na meta-heurística *Iterated Local Search* (ILS). O procedimento de busca local foi inspirada na descida em vizinhança variável aleatória (*Randomized Variable Neighborhood Descent* – RVND) (Subramanian et al., 2010). Por último, foi incorporado um procedimento exato durante a execução do algoritmo ILS. A ideia é armazenar um conjunto de rotas geradas durante a heurística e, em seguida, resolver o problema do *Set Partitioning* (SP) de forma exata para extrair a melhor combinação de rotas.

1.3 Objetivos

1.3.1 Objetivo geral

Propor um algoritmo híbrido combinando uma meta-heurística baseada em busca local com programação matemática para variantes do DARP.

1.3.2 Objetivos específicos

Os objetivos específicos deste trabalho são os seguintes:

- Revisar literatura sobre DARP, descrevendo aplicações e soluções propostas pela literatura.
- Implementar um procedimento de construção de soluções iniciais.
- Implementar esquemas de vizinhanças e perturbação baseados na literatura.
- Propor estruturas de avaliação de vizinhança eficientes.
- Propor estruturas de aceleração para as buscas locais.
- Testar o algoritmo proposto nas instâncias da literatura.
- Comparar os resultados com outros métodos da literatura.

1.4 Estrutura da dissertação

O restante deste trabalho está organizado da seguinte forma:

- Capítulo 2 apresenta uma revisão dos trabalhos relacionados ao DARP.
- Capítulo 3 descreve o problema e uma formulação matemática.
- Capítulo 4 explica em detalhes a abordagem híbrida e o algoritmo proposto, discutindo seus componentes.
- Capítulo 5 apresenta os experimentos computacionais.
- Capítulo 6 contém as considerações finais e comentários sobre possíveis melhorias para trabalhos futuros.

Capítulo 2

Trabalhos relacionados

Diversos trabalhos foram publicados nos últimos 45 anos sobre DARP e suas variações. Os primeiros trabalhos publicados estudaram um sistema de discagem e carona em tempo real desenvolvido na época (Wilson et al., 1971; Wilson e Weissberg, 1976). A heurística não tratava de restrições de janelas de tempo, mas apenas com restrições de precedências entre a coleta e a entrega. Além disso, o sistema utilizava dos horários de serviço para adquirir informações relacionadas ao tempo de viagem e tempo de espera dos clientes como estatísticas para melhorar a qualidade do algoritmo.

Restrições de janela de tempo nos pontos de coleta e entrega surgiram com o trabalho de Psaraftis (1983), que propôs uma solução exata por Programação Dinâmica (PD) para versões estáticas e dinâmicas do DARP com apenas um veículo.

Uma versão mais moderna do problema foi apresentada por Cordeau e Laporte (2003a) em que, além das restrições de capacidade dos veículos, janela de tempo e precedência entre coleta e entrega, também foram levadas em considerações as restrições de máxima duração de viagem dos clientes e máxima duração da rota. Este capítulo tem como objetivo fazer uma revisão da literatura das variações e suas respectivas soluções para o DARP proposto por Cordeau e Laporte (2003a).

O trabalho de Cordeau e Laporte (2003b) introduziu uma abordagem via metaheurística busca tabu (do inglês, *Tabu Search* – TS), que permite soluções que violem capacidade e janela de tempo durante sua fase de busca. Os autores reportaram difi-

culdades durante as avaliações de vizinhança para satisfazer as restrições de máxima duração de viagem dos clientes e máxima duração da rota. Para resolver esse problema, foi apresentado um algoritmo de 8 passos capaz de verificar, dado uma rota viável em função das janelas de tempo e da capacidade, se ela é viável para o tempo máximo de viagem dos clientes e da duração máxima da rota. Esse algoritmo de avaliação tem como complexidade de tempo $\mathcal{O}(n^2)$, onde n é o tamanho da rota, que dentro de uma busca local pode se tornar uma operação de alto custo computacional. Por último, foram apresentadas instâncias com até 13 veículos e 144 requisições.

Para uma versão dinâmica do DARP, isto é, em que as requisições são apresentadas aos poucos e o objetivo é aceitar o maior número possível de requisições, Attanasio et al. (2004) descreveu o seguinte algoritmo: primeiro, uma solução inicial é construída com base nas requisições já conhecidas até um certo momento. Segundo, quando uma nova requisição chega, executa-se um procedimento de checagem para saber se ela pode ser aceita ou não pelas rotas. Por último, decidido em qual rota a requisição foi aceita, uma pós-otimização é feita sobre a nova rota com o objetivo de minimizar os custos. Essa heurística foi usada dentro de uma TS. Outra contribuição dos autores foi implementar e testar procedimentos de computação paralela para melhorar os tempos de computacionais.

Uma abordagem para o DARP via programação inteira mista (do inglês, *Mixed-Integer Programming* – MIP) e um algoritmo *Branch-and-Cut* (B&C) foram implementados por Cordeau (2006). Além de apresentar uma formulação matemática para o problema, o autor introduziu desigualdades válidas herdadas de problemas como PDP e PRV. O autor também descreveu novas desigualdades válidas para o DARP derivadas de sua estrutura específica. Outra contribuição importante apresentada foram os pré-processamentos utilizados sobre as instâncias do problema, com o objetivo de estreitar as janelas de tempo e eliminar arcos inviáveis entre os vértices. Os testes foram realizados sobre dois novos grupo de instâncias com até 4 veículos e 48 requisições. Como resultado, o autor identificou que o algoritmo B&C foi capaz de resolver instâncias com até 24 requisições.

Ropke et al. (2007) apresentaram um algoritmo B&C para PDP e uma adaptação para o DARP. A formulação matemática para o DARP proposta é mais forte do que

a abordagem de Cordeau (2006), tendo como diferença que as variáveis de decisão possuem apenas dois índices e podem apenas assumir valores binários, enquanto outras formulações possuem variáveis com até três índices e variáveis podendo assumir valores contínuos. Os autores também ampliaram as instâncias apresentadas em Cordeau (2006) para instâncias com até 8 veículos e 96 requisições. Os testes demostraram que o algoritmo foi capaz de resolver instâncias com até 96 requisições de forma exata em até duas horas.

Parragh et al. (2010) tomaram como base os trabalhos de Cordeau e Laporte (2003b) e Cordeau (2006) para resolver o DARP por uma abordagem usando a meta-heurística *Variable neighborhood search* (VNS). Os autores descreveram três novos movimentos de vizinhança inter-rota fundamentais para a busca local. O algoritmo foi também testado para uma variação do problema em que a função objetivo considera não só minimizar os custos de viagem, mas também o total de violação do tempo máximo de viagem e da duração da rota. Testes foram realizados comparando os resultados do trabalho com os de Cordeau e Laporte (2003b) sobre o mesmo conjunto de instâncias. Melhores soluções foram encontradas para 16 instâncias.

Uma interessante variante para o DARP foi introduzida por Parragh (2011), em que os clientes podem demandar diferentes tipos de recursos em um veículo, ou seja, o problema com demanda e frota heterogênea, HDARP. Essa variação foi motivada pela *Austrian Red Cross* (organização nacional da Cruz Vermelha na Áustria) para transportar seus pacientes. Os pacientes podem ser transportados de três formas: sentados, em cadeiras de rodas ou em uma maca. Além disso, a capacidade de recursos disponíveis nos veículos da frota pode variar. O trabalho apresentou duas formulações matemáticas para o HDARP, algoritmos de B&C com desigualdades válidas baseadas em Cordeau (2006); Ropke et al. (2007) e uma meta-heurística VNS, com pequenas modificações da descrita em Parragh et al. (2010) para as novas restrições do problema. Os testes foram realizados em adaptações das instâncias de Cordeau (2006) e mostraram que os métodos exatos encontram soluções ótimas em instâncias com até 40 requisições, enquanto a meta-heurística forneceu resultados iguais ou próximos aos ótimos.

Jain e Van Hentenryck (2011) implementaram a meta-heurística *Large Neighborhood Search* (LNS) combinada com técnicas de *Constraint Programming* (CP) para o DARP.

O procedimento baseado em CP é chamado repetidas vezes para encontrar reinserções de clientes que minimizam o custo da solução. Experimentos foram feitos sobre as instâncias usadas em Cordeau e Laporte (2003a) e Parragh et al. (2010) e apresentaram melhorias em relação às médias das soluções encontradas quando comparado à outros métodos disponíveis na literatura até o momento em que o trabalho foi publicado. Além disso, novas melhores soluções foram reportadas.

Parragh e Schmid (2013) propuseram a resolução do DARP por um algoritmo híbrido formado pela combinação de dois métodos: Geração de Colunas (GC), para resolver o *Set Covering Problem* (SCP), em que uma heurística VNS gera rotas de custo reduzido negativo para a GC; e LNS, aplicado à solução gerada pela GC. Os testes foram realizados nas instâncias da literatura do DARP. Os resultados obtidos mostraram-se competitivos ao estado da arte daquele período, encontrando novas melhores soluções em algumas instâncias.

Uma nova variante para o HDARP em que os veículos podem sair de diferentes depósitos foi apresentada por Braekers et al. (2014), o problema foi denominado *Multi-Depot Heterogeneous DARP* (MDHDARP). Foi proposta uma adaptação do algoritmo de B&C para HDARP publicado em Parragh (2011) para resolver o MDHDARP. Além de adequar as novas restrições do problema, os autores criaram uma modelagem matemática mais compacta. Os resultados mostraram que o novo algoritmo de B&C conseguiu resolver, de forma exata e em segundos, todas as instâncias do HDARP. Os autores também implementaram a meta-heurística de *Deterministic Annealing* (DA), este algoritmo foi testado para resolver instâncias do DARP, do HDARP e do MDHDARP. Os resultados superaram outras abordagens meta-heurísticas do estado da arte daquele momento tanto em tempo computacional, como em qualidade da solução. Por último, apresentaram novas instâncias para o HDARP e para o MDHDARP com até 8 veículos e 96 requisições baseadas nas instâncias de Ropke et al. (2007).

Uma abordagem via algoritmo de *Branch-and-Price-and-Cut* (B&P&C) foi publicada por Gschwind e Irnich (2015). O método utiliza uma GC na qual as restrições de janela de tempo e tempo máximo de viagem são tratadas no subproblema. Realizaram-se testes em dois grupos de instâncias do DARP com até 96 requisições. O algoritmo encontrou todas as soluções ótimas, com exceção de uma única instância, em tempos

computacionais de no máximo 1 hora. Dessa forma, esse trabalho superou outros trabalhos baseados em métodos exatos tais como (Cordeau, 2006) e (Ropke et al., 2007).

Uma variante do MDHDARP foi introduzida por Masmoudi et al. (2016), em que os condutores de cada veículo devem fazer pausas durante suas rotas, simulando horários de refeição. O problema foi denominado de *Multi-Depot Multi-Trip Heterogeneous Dial-A-Ride Problem* (MDMTHDARP). Além de uma formulação matemática, os autores apresentaram três meta-heurísticas diferentes para solucionar o novo problema: *Bees Algorithm with Deterministic Annealing* (BA-DA), *Bees Algorithm with Simulated Annealing* (BA-SA) e *Adaptative Large Neighborhood Search* (ALNS). Esses algoritmos heurísticos também foram utilizados para resolver as instâncias do MDHDARP trabalhadas em Braekers et al. (2014). Os resultados dos testes computacionais mostraram que os algoritmos BA-DA e BA-SA obtiveram melhores resultados do que o ALNS para ambos os problemas. Por último, apresentaram-se novas melhores soluções para o MDHDARP.

Masmoudi et al. (2017) resolveu o DARP e o HDARP com a união de algoritmos genéticos (AG) e buscas locais, esses últimos têm o objetivo de melhorar a qualidade das soluções geradas a cada geração pelo AG. Os experimentos mostraram que o algoritmo proposto encontrou soluções melhores do que o estado da arte daquele ano. Outra contribuição importante foi gerar novas instâncias para o HDARP com até 13 veículos e até 144 requisições.

Molenbruch et al. (2017a) investigaram efeitos operacionais de possíveis cooperações entre serviços que utilizam soluções para o DARP. No mundo real, podem existir diversas empresas que prestam serviços de transportes de pessoas em uma região. Um cliente escolhe o serviço associado à uma transportadora, que deve planejar o roteamento dos seus clientes. Dessa forma, empresas podem criar sistemas de cooperação com o objetivo de compartilhar clientes para diminuir seus custos operacionais. Foi desenvolvida uma meta-heurística LNS para solucionar o problema. Utilizando as instâncias do MDHDARP (Braekers et al., 2014), os autores realizaram testes com diferentes combinações cooperações entre as transportadoras, onde cada empresa foi associada a um depósito das instâncias. É importante destacar que, quando ocorre 100% de cooperação, o problema é idêntico ao DARP com múltiplos depósitos (do inglês, *Multi-Depot*

DARP – MDDARP). Os resultados apresentados alcançaram soluções competitivas com o estado da arte.

Uma aplicação do DARP para veículos elétricos em que trocas de baterias podem ser realizadas durante o seu percurso (do inglês, *DARP with Electric Vehicle* – DARP-EV) foi proposta por Masmoudi et al. (2018). Ao longo da rota, o consumo de energia de cada veículo é monitorado e caso necessário, o veículo deve passar por pontos de troca de bateria para continuar sua trajetória. Os autores desenvolveram um algoritmo que combina o VNS com abordagens evolutivas utilizadas em meta-heurísticas baseadas em população. Além de experimentos em adaptações de instâncias para o problema, os autores realizaram testes nas instâncias do HDARP (Braekers et al., 2014; Masmoudi et al., 2017). Os resultados demonstraram que para o HDARP, o algoritmo obteve resultados competitivos aos de Masmoudi et al. (2017) em qualidade de solução, porém o método proposto obteve desempenho inferior em tempos computacionais em instâncias de médio e grande porte.

Uma contribuição importante para o DARP foi realizada por Gschwind e Drexl (2019) que propôs uma checagem de viabilidade de rota para as restrições do DARP em tempo $\mathcal{O}(1)$ amortizado. Ou seja, com pré-processamento de estruturas de dados auxiliares, a verificação tem custo de tempo constante independente da quantidade de requisições na rota. Essa rotina de verificação é implementada na meta-heurística ALNS que utiliza duas técnicas para melhorar suas soluções: buscas locais e MIP para resolver o SCP em um subconjunto de rotas geradas durante as buscas. Testes foram realizados sobre instâncias de grande porte do DARP. Como resultado, novas melhores soluções foram apresentadas.

Recentemente, Masmoudi et al. (2020) apresentaram um algoritmo baseado na meta-heurística ALNS para resolver o DARP com frota heterogênea e mista (do inglês, *Mixed Fleet Heterogeneous Dial-A-Ride Problem* – MF-HDARP). Para essa variante do problema, registram-se os gastos de combustível para alguns veículos da frota, assim, o roteamento deve considerar passar por pontos de abastecimento se necessário. Os autores realizaram experimentos em adaptações das instâncias propostas por Masmoudi et al. (2017). Além disso, o algoritmo ALNS foi aplicado nas instâncias do HDARP e como resultado foram reportadas três novas melhores soluções.

Revisões de literatura a cerca do DARP podem ser encontrados nos trabalhos de Molenbruch et al. (2017b) e Ho et al. (2018). A Tabela 2.1 apresenta um resumo dos principais trabalhos sobre o DARP.

Tabela 2.1: Principais trabalhos relacionados ao DARP

Referência	**Abordagem e/ou Contribuição**
Wilson et al. (1971)	Heurística para primeiras versões do DARP
Wilson e Weissberg (1976)	Extensão do trabalho anterior
Psaraftis (1983)	PD para primeiras versões do DARP com janela de tempo
Cordeau e Laporte (2003a)	Definição e modelagem para versão moderna do DARP
Cordeau e Laporte (2003b)	TS para DARP e algoritmo de checagem para tempo máximo de viagem
Attanasio et al. (2004)	DARP dinâmico com computação paralela
Cordeau (2006)	Algoritmo de B&C para DARP
Ropke et al. (2007)	Algoritmo de B&C para DARP
Parragh et al. (2010)	VNS para DARP e variação do DARP com nova função objetivo
Parragh (2011)	B&C e VNS para HDARP
Jain e Van Hentenryck (2011)	LNS com CP para DARP
Parragh e Schmid (2013)	LNS com GC para DARP
Braekers et al. (2014)	B&C e DA para MDHDARP, HDARP e DARP
Gschwind e Irnich (2015)	B&P&C para o DARP
Masmoudi et al. (2016)	BA-DA, BA-SA e ALNS para MDHDARP
Masmoudi et al. (2017)	AG e busca local para HDARP e DARP
Molenbruch et al. (2017a)	LNS para MDDARP
Molenbruch et al. (2017b)	Revisão de literatura do DARP
Masmoudi et al. (2018)	VNS com abordagem evolutiva para DARP-EV
Ho et al. (2018)	Revisão de literatura do DARP
Gschwind e Drexl (2019)	ALNS e SCP para DARP e checagem de viabilidade em tempo constante
Masmoudi et al. (2020)	ALNS para resolver o MF-HDARP

Capítulo 3

Definição do problema e formulação matemática

Neste capítulo serão definidos os aspectos gerais do DARP e suas variações (HDARP e MDHDARP), introduzindo notação e explicação dos dados, conjuntos de variáveis, função objetivo e restrições envolvidas no problema. Por fim, apresenta-se uma formulação matemática.

3.1 Definição do problema

Seja n o número de requisições e v o número de veículos. O conjunto $P = \{1, \ldots, n\}$ representa os vértices de coleta, o conjunto $D = \{n+1, \ldots, 2n\}$ representa os vértices de entrega, os conjuntos $M_o = \{2n+1, \ldots, 2n+v\}$ e $M_e = \{2n+v+1, \ldots, 2n+2v\}$ representam os conjuntos de vértices de depósito de origem e fim de rota, respectivamente, associados à cada veículo. Ou seja, para cada veículo, existe um depósito representando sua origem e um depósito representando seu último vértice da rota. Porém, mais de um veículo pode ter o mesmo depósito, o que significa que M_o e M_e podem conter vértices que indicam o mesmo depósito. Para cada requisição, existe um vértice de coleta $i \in P$ e um vértice de entrega $i + n \in D$. Definem-se os conjuntos $N = P \cup D$, os vértices de coleta e entrega, e $V = N \cup M_o \cup M_e$, vértices de coleta,

entrega e os depósitos de origem e fim de cada veículo.

Os arcos entre os vértices são representados pelos seguintes conjuntos: A^{mp} contém todos arcos que conectam um depósito $m \in M_o$ a um vértice de coleta $i \in P$; A^{pd} são todos arcos que conectam um vértice de coleta ou entrega $i \in N$ a outro vértice de coleta ou entrega $j \in N$ tal que $i \neq j$ e $j + n \neq i$, ou seja, não existem arcos de um vértice para ele mesmo, nem de um vértice de entrega para sua coleta; A^{dm} são todos os arcos que conectam um vértice de entrega $j \in D$ a um vértice de depósito $m' \in M_e$; e A^{mm} são os arcos de um depósito de origem de M_o para seu respectivo depósito final em M_e, para o caso de rotas vazias. Por fim, define-se $A = A^{mp} \cup A^{pd} \cup A^{dm} \cup A^{mm}$ como o conjunto de arcos.

Com as definições dos conjuntos V e A, define-se $G = (V, A)$ como um grafo orientado com conjunto de vértices V e conjunto de arcos A. Para cada arco $(i, j) \in A$, existem valores t_{ij} e c_{ij}, que indicam o tempo e o custo do arco. Assume-se que t_{ij} e c_{ij} satisfazem a desigualdade triangular.

Outros conjuntos usados no problema são: $K = \{1, \ldots, v\}$, correspondente aos veículos, e $R = \{0, \ldots, 3\}$, que indica os recursos solicitados por uma requisição. Atribui-se 0 para assentos de acompanhantes, 1 para assentos de pacientes, 2 para macas e 3 para cadeiras de rodas. Uma demanda de assento para acompanhante pode consumir os recursos $0, 1$ ou 2; uma demanda de assento para paciente pode consumir os recursos 1 ou 2; e demandas de viagem por maca ou cadeira de rodas só podem consumir seus respectivos recursos.

Para cada vértice $i \in V$, existe um tempo de serviço d_i, uma janela de tempo $[e_i, l_i]$ e uma demanda q_i^r, $r \in R$. Os vértices de coleta e entrega possuem a relação de que $q_{i+n}^r = -q_i^r$, $i \in P$, $r \in R$ e os vértices de depósito possuem as relações de que $q_{2n+k}^r = q_{2n+v+k}^r = 0$, $k \in K$, $r \in R$ e de que $d_{2n+k} = d_{2n+v+k} = 0$, $k \in K$. Para cada $i \in P \cup M_o$ existe um tempo máximo de viagem L_i. A diferença entre o tempo de começo de serviço do vértice de entrega e a chegada ao vértice de entrega deve ser menor que o L_i. Essa mesma restrição é usada entre o tempo de saída e de retorno ao depósito que não deve exceder o tempo máximo de duração da rota.

Outros conjuntos auxiliares de arcos usados para facilitar a escrita da formulação matemática são: $\delta^+(i, k)$, que representa os arcos que saem de $i \in V$ e podem ser usados

pelo veículo $k \in K$, e $\delta^-(i,k)$, que representa os arcos que incidem em $i \in V$ e podem ser usados pelo veículo $k \in K$. Os arcos que envolvem um depósito não podem ser atravessados por qualquer veículo, mas apenas pelo veículo associado àquele depósito, daí a necessidade do índice k na definição desses conjuntos. Por último, define-se o conjunto $K(a) \subseteq K$, onde $a \in A$, o conjunto de veículos que podem atravessar o arco a. Para cada veículo $k \in K$, existe uma capacidade C^{rk}, $r \in R$, associada.

As constantes e os conjuntos definidos acima são necessários para o MDHDARP. Para o HDARP, considera-se que todo o conjunto M_o e o conjunto M_e são réplicas de um mesmo depósito. Para o DARP, os recursos também são restringidos para $R = \{1\}$.

Rotas devem ser criadas para atender todas as requisições do problema, tendo como objetivo a minimização dos custos. Para isso, as rotas devem satisfazer as seguintes restrições:

- Uma requisição i deve ser atendida por exatamente um veículo. Este veículo deve visitar o vértice i, para fazer a coleta da demanda, antes de visitar o vértice $i+n$, onde ele fará a entrega da demanda;
- Cada veículo deve retornar ao seu local de origem, o seu depósito, ao final da rota;
- As capacidades C^{rk}, $r \in R$, $k \in K$, não podem ser excedidas em nenhum trecho da rota;
- Cada requisição deve ter a sua coleta e a sua entrega atendidas dentro de suas janelas de tempo, $[e_i, l_i]$ e $[e_{i+n}, l_{i+n}]$, para todo $i \in P$;
- O tempo total entre a coleta e a entrega da requisição i não deve exceder seu tempo máximo de viagem L_i, $i \in P$.

3.2 Formulação matemática proposta

A formulação proposta foi inspirada nos trabalhos de Cordeau (2006) e Parragh (2011).

A variável de decisão x_{ij}^k assume o valor 1 quando o veículo $k \in K$ passa pelo arco $(i,j) \in A$, e 0 caso contrário. A variável contínua b_i indica o tempo de início da visita ao vértice $i \in V$, podendo assumir qualquer valor não negativo. A variável contínua Q_i^r representa a carga de um recurso $r \in R$ no vértice $i \in V$, podendo assumir qualquer valor não negativo.

$$min \sum_{a \in A} \sum_{k \in K(a)} c_a x_a^k \tag{3.1}$$

Sujeito a:

$$\sum_{k \in K} \sum_{a \in \delta^+(i,k)} x_a^k = 1 \qquad i \in P \tag{3.2}$$

$$\sum_{a \in \delta^+(i,k)} x_a^k - \sum_{a \in \delta^+(n+i,k)} x_a^k = 0 \qquad i \in P, k \in K \tag{3.3}$$

$$\sum_{a \in \delta^+(i,k)} x_a^k - \sum_{a \in \delta^-(i,k)} x_a^k = 0 \qquad i \in N, k \in K \tag{3.4}$$

$$\sum_{a \in \delta^+(2n+k,k)} x_a^k \leq 1 \qquad k \in K \tag{3.5}$$

$$b_j \geq b_i + t_{ij} + d_i - B(1 - \sum_{k \in K((i,j))} x_{ij}^k) \qquad (i,j) \in A \tag{3.6}$$

$$e_i \leq b_i \leq l_i \qquad i \in V \tag{3.7}$$

$$b_{n+i} - (b_i + d_i) \leq L_i \qquad i \in P \tag{3.8}$$

$$b_{2n+v+i} - (b_i + d_i) \leq L_i \qquad i \in M_o \tag{3.9}$$

$$Q_j^r \geq Q_i^r + q_j^r - W(1 - \sum_{k \in K((i,j))} x_{ij}^k) \qquad (i,j) \in A, r \in R \tag{3.10}$$

$$\sum_{r'=r}^{2} Q_i^{r'} \leq \sum_{r'=r}^{2} \sum_{k \in K} \sum_{a \in \delta^-(i,k)} C^{r'k} x_a^k \qquad i \in V, r \in R \setminus \{3\} \tag{3.11}$$

$$Q_i^3 \leq \sum_{k \in K} \sum_{a \in \delta^-(i,k)} C^{3k} x_a^k \qquad i \in V \tag{3.12}$$

$$x_a^k \in \{0,1\} \qquad a \in A, k \in K(a) \tag{3.13}$$

$$b_i \geq 0 \qquad i \in V \tag{3.14}$$

$$Q_i^r \geq 0 \qquad i \in V, r \in R \tag{3.15}$$

A função objetivo (3.1) minimiza a soma dos custos de viagem de todos os veículos. As restrições (3.2) garantem que cada requisição só pode ser servida uma única vez. As restrições (3.3) afirmam que o mesmo veículo que serve um vértice de coleta i também deve servir seu respectivo vértice de entrega $n+i$. As restrições (3.4) asseguram que um mesmo veículo que entra em um vértice i deve sair dele. As restrições (3.5) garantem que um veículo só pode sair no máximo uma única vez do depósito. As restrições (3.6)–(3.7) fixam os valores para o tempo de visita em cada vértice. As restrições (3.8)–(3.9) referem-se aos tempos máximo de viagem para cada par de coleta/entrega e máxima duração da rota. As restrições (3.10)–(3.12) asseguram as restrições de capacidade do HDARP. Por fim, as restrições (3.13)–(3.15) dizem respeito aos possíveis valores que as variáveis podem assumir. Por fim, é importante destacar que as restrições (3.6) e (3.10) são inicialmente não lineares. Por isso, foram utilizadas as constantes B e W, respectivamente, para a linearização de suas restrições.

As desigualdades válidas (3.16), baseadas em Desrochers e Laporte (1991), removem sub-rotas da solução. Elas são adicionadas dinamicamente ao modelo. Embora sejam redundantes, essas inequações possibilitam o fortalecimento a relaxação linear.

$$\sum_{j \in S} \sum_{k \in K} \sum_{\substack{(i,j) \in \delta^-(j,k) \\ i \notin S}} x_{ij}^k \geq 1 \qquad \forall S \subseteq N \tag{3.16}$$

Capítulo 4

Uma abordagem híbrida

Este capítulo apresenta a abordagem proposta para resolver uma classe de problemas de roteamento do tipo *dial-a-ride*. O método consiste em um algoritmo híbrido que combina a meta-heurística ILS com método exato para resolver o problema SP. Durante o ILS, são realizadas buscas locais baseadas no método RVND proposto por Subramanian et al. (2010). As próximas seções descrevem o algoritmo em detalhes, incluindo a ideia geral da meta-heurística, o modelo do SP, a construção de soluções iniciais, o algoritmo de busca local, os movimentos de vizinhança, as avaliações dos movimentos, estruturas de aceleração e os procedimentos de pertubação.

4.1 *Iterated local search*

O Algoritmo 1 descreve o funcionamento do ILS, apresentando cada um dos seus componentes. A partir de uma solução inicial gerada por um procedimento de construção, buscas locais são aplicadas nessa solução com o objetivo de melhorar sua qualidade. Os mecanismos de perturbação promovem aleatoriedade ao algoritmo ajudando-o a fugir de ótimos locais. Em outras palavras, o ILS iterativamente alterna entre buscas locais (intensificação) e procedimentos de perturbação (diversificação) para encontrar soluções de alta qualidade.

Algoritmo 1 ILS

```
1: Procedimento ILS
2: s_0 ← GeracaoSolucaoInicial()
3: s* ← BuscaLocal(s_0)
4: while condição de parada não satisfeita do
5:    s' ← Perturbacao(s* )
6:    s*' ← BuscaLocal(s')
7:    s* ← CriterioDeAceitacao(s*, s*')
8: return s*
```

4.2 Algoritmo proposto ILS-SP$_{\text{DARP}}$

Esta seção descreve o funcionamento do algoritmo proposto, chamado de ILS-SP$_{\text{DARP}}$, o pseudocódigo é apresentado no Algoritmo 2. O procedimento começa realizando um pré-processamento sobre a instância de entrada (linha 2). Baseado nos trabalhos de Cordeau (2006) e Parragh (2011), são feitos estreitamentos das janelas de tempo e uma análise sobre pares de requisições para proibir arcos que violam janela de tempo ou tempo máximo de viagem. Além disso, modificações sobre as representações das demandas das requisições e das capacidades dos veículos são realizadas para que cada recurso possa ser processado e avaliado de forma independente durante a execução do método.

O laço de iteração principal do ILS-SP$_{\text{DARP}}$ (linhas 7–17) é executado I_R vezes, indicando a quantidade de inicializações da heurística. Uma solução inicial é gerada pelo método construtivo (linha 9). Em seguida, o ILS$_{\text{DARP}}$ procura melhorar qualidade da solução corrente por meio de procedimentos de buscas locais e mecanismos de perturbação (linha 10). Durante a execução do ILS$_{\text{DARP}}$, as rotas provenientes de soluções ótimas locais são armazenadas em duas estruturas (*pool* e $pool_{temp}$ de rotas) para serem usadas posteriormente nos métodos que resolvem o SP. O $pool_{temp}$ é reiniciado em cada nova execução do laço de iteração (linha 8), o procedimento de resolução do SP é acionado utilizando as rotas armazenadas em $pool_{temp}$ ao final de cada iteração do laço (linhas 12 e 13). É importante destacar que as rotas da melhor solução encontrada naquela iteração são armazenadas em uma terceira estrutura, o $pool_{bests}$ (linha 14). O $pool_{temp}$ é formada pelas rotas de soluções ótimas locais geradas pelo ILS$_{\text{DARP}}$ da iteração corrente e as rotas armazenadas em $pool_{bests}$.

A última operação (linha 19), resolve o SP utilizando *pool* que armazena todas as

rotas de soluções ótimas locais encontrados durante a execução do algoritmo. Caso seja obtida uma melhora, o valor da função objetivo é atualizado.

Algoritmo 2 ILS-SP$_{\text{DARP}}$

```
1: Procedimento ILS-SP_DARP (I_R, I_ILS, I_ρ)
2: preprocessamento()
3: pool ← ∅
4: pool_bests ← ∅
5: s* ← ∅
6: f* ← ∞
7: for i ← 1 to I_R do
8:    pool_temp ← ∅
9:    s ← gerarSolucaoInicial()
10:   s ← ILS_DARP(I_ILS, I_ρ, pool, pool_temp, s)
11:   pool_temp ← pool_temp ∪ pool_bests
12:   modeloSP ← criaModeloSP(pool_temp)
13:   s ← SP(modeloSP, s, I_ILS, I_ρ, pool, pool_temp)
14:   pool_bests ← pool_bests ∪ rotas de s
15:   if f(s) ≤ f* then
16:      s* ← s
17:      f* ← f(s)
18: modeloSP ← criaModeloSP(pool*)
19: s* ← SP(modeloSP, s*, I_ILS, I_ρ, pool, pool_temp)
20: if f(s*) < f* then
21:    f* ← f(s*)
22: return s*
```

4.3 Algoritmo ILS$_{\text{DARP}}$

O Algoritmo 3 tem o objetivo de explorar o espaço de soluções a partir de uma solução s. Por meio dos procedimentos de busca local (linha 6) e mecanismos de perturbação (linha 13), procura-se explorar e melhorar a qualidade da solução corrente. Após cada busca local, armazenam-se as rotas da solução encontrada em duas estruturas diferentes, *pool* e $pool_{temp}$ (linhas 7 e 8). O parâmetro I_{ILS} indica o número máximo de perturbações consecutivas permitidas sem que ocorra melhora da solução.

4.4 *Set partitioning*

Rotas de soluções ótimas locais provenientes das buscas locais do ILS$_{\text{DARP}}$ são armazenadas para serem usadas como entrada para o SP. O método procura a combinação

Algoritmo 3 ILS_{DARP}

```
1: Procedimento ILS_DARP (I_ILS, I_ρ, pool, pool_temp, s)
2: s* ← s
3: f* ← f(s)
4: iter ← 0
5: while iter ≤ I_ILS do
6:    s ← BL(s, iter, I_ILS)
7:    pool ← pool ∪ rotas de s
8:    pool_temp ← pool_temp ∪ rotas de s
9:    if f(s) < f* then
10:      s* ← s
11:      f* ← f(s)
12:      iter ← 0
13:   s ← perturbar(s*, I_ρ)
14:   iter ← iter + 1
15: return s*
```

de rotas de menor custo que satisfazem as restrições do problema. A solução para o modelo SP é encontrada de forma exata por um resolvedor de MIP.

Os veículos com mesma capacidade e mesmo tempo máximo de viagem são agrupados em um conjunto de tipos de veículos $\mathcal{K}$. O conjunto U_k^m para $k \in \mathcal{K}$, $m \in M_o$ indica a quantidade de cada tipo de veículo pertencente a cada depósito, ou seja, a frota de veículos de cada depósito.

Seja $\mathcal{R}$ o conjunto de todas as rotas armazenadas no *pool*, $\mathcal{R}_i \subseteq \mathcal{R}$ é definido como o subconjunto de rotas que atendem a requisição $i \in P$. Define-se $\mathcal{R}_k^m \subseteq \mathcal{R}$ como o subconjunto de rotas que aparecem o veículo do tipo $k \in \mathcal{K}$ no depósito $m \in M_o$. A variável y_j assume valor 1 caso a rota $j \in \mathcal{R}$ faça parte do particionamento, ou 0 caso contrário. A constante c_j representa o custo da rota $j \in \mathcal{R}$.

$$\min \sum_{j \in \mathcal{R}} c_j y_j \tag{4.1}$$

Sujeito a:

$$\sum_{j \in \mathcal{R}_i} y_j = 1 \qquad i \in P \tag{4.2}$$

$$\sum_{j \in \mathcal{R}_k^m} y_j \leq U_k^m \qquad k \in \mathcal{K}, m \in M_o \tag{4.3}$$

$$y_j \in \{0, 1\} \qquad j \in \mathcal{R}. \tag{4.4}$$

A função objetivo (4.1) minimiza a soma dos custos das rotas. As restrições (4.2) garantem que para cada vértice $i \in P$ existe uma única rota $j \in \mathcal{R}_i$. As restrições (4.3) indicam que a quantidade de veículos de um certo tipo e pertencente a um determinado depósito não deve ultrapassar o limite da frota. As restrições (4.4) referem-se ao domínio das variáveis de decisão y_j.

O Algoritmo 4 demostra o funcionamento para resolução do SP, que por sua vez é baseado em Subramanian et al. (2013). Primeiro, armazena-se o valor do custo da solução na variável de $cutoff$. Em seguida, utiliza-se um resolvedor MIP para o modelo (linha 3). O procedimento `MIPSolver` recebe como parâmetros o modelo que será resolvido, uma solução para ser utilizada para o *MIP start*, o valor dessa solução e o procedimento `incCbk`. Esse procedimento realiza uma chamada do ILS_{DARP} para cada solução incumbente encontrada pelo `MIPSolver`, sendo essa solução incumbente a solução inicial do ILS (linhas 6–11). Caso o ILS_{DARP} encontre uma solução melhor do que a incumbente, atualiza-se a melhor solução e o valor de $cutoff$ para o `MIPSolver` (linhas 9–11).

Algoritmo 4 SP

```
1: Procedimento SP(modeloSP, s, I_ILS, I_ρ, pool, pool_temp)
2: cutoff ← f(s)
3: s ← MIPSolver(modeloSP, s, cutoff, incCbk(s, I_ILS, I_ρ, pool, pool_temp))
4: return s
5:
6: Procedimento incCbk(s*, I_ILS, I_ρ, pool, pool_temp)
7: s ← solução incumbente
8: s ← ILS_DARP(I_ILS, I_ρ, pool, pool_temp, s)
9: if f(s) < f(s*) then
10:    s* ← s
11:    cutoff ← f(s)
```

4.5 Procedimento construtivo

O procedimento guloso aleatório usado para criação da solução inicial é descrito no Algoritmo 5. O algoritmo começa com uma lista de elementos candidatos (CL) a serem inseridos em alguma rota. Cada elemento dessa lista corresponde à uma requisição, ou seja, um vértice de coleta e um vértice de entrega (linha 4). Para cada veículo,

é inicializada uma rota com somente dois vértices, o depósito de começo da rota e o depósito final da rota (linha 8). Em seguida, para cada rota, um elemento da *CL* é escolhido aleatoriamente para ser inserido entre os depósitos, garantindo diversificação na solução inicial (linha 9).

Assumindo que cada rota possui depósitos de começo e fim de rota, e um par coleta/entrega, o algoritmo segue a seguinte abordagem gulosa: escolhe-se um elemento da lista de candidatos (linha 13); avalia-se em todas as rotas e em todas as posições de uma rota qual inserção tem menor custo e não viola restrições do problema (linhas 16–28). Cada elemento da CL é composto por um par, o que significa que avaliar a posição de inserção de um elemento significa encontrar a melhor posição para inserir uma coleta e uma entrega, onde a entrega sempre vem após sua respectiva coleta. Encontrada a melhor posição de inserção, o par é inserido na rota e removido da lista de candidatos (linhas 29–31). O procedimento se repete até que a lista de candidatos esteja vazia. Caso não seja possível inserir elementos da *CL* em nenhuma das rotas sem a violação das restrições, uma nova construção é iniciada.

4.6 Busca local

A busca local foi inspirada no RVND do trabalho de Subramanian et al. (2010). O RVND consiste em uma variação do *Variable Neighborhood Descent* (VND) (Hansen e Mladenović, 2001) de forma que as vizinhanças são escolhidas de forma aleatória. O objetivo desse método é melhorar a qualidade de uma solução corrente. Para isso, o procedimento utiliza uma lista de vizinhanças enumeradas, $\mathcal{N} = \{\mathcal{N}^{(1)}, \mathcal{N}^{(2)}, \ldots, \mathcal{N}^{(n)}\}$, em que cada elemento representa um movimento. O algoritmo escolhe uma das vizinhanças de forma aleatória e aplica-o sobre a solução corrente. Caso não se obtenha melhora na função objetivo, a vizinhança escolhida é removida da lista; caso contrário, a lista é reiniciada. Neste trabalho, a vizinhança é composta por seis movimentos inter-rota e dois movimentos intra-rota.

A estrutura de vizinhança $\mathcal{N}$ foi dividida em três listas de vizinhanças: $\mathcal{N}_{inter}$, $\mathcal{N}_{intra}$ e $\mathcal{N}_{blocos}$ que representam, respectivamente, movimentos inter-rota, movimentos intra-rota e movimentos inter-rota de blocos. Esse último conjunto de vizinhanças consiste

Algoritmo 5 Construção de soluções iniciais

```
1: Procedimento gerarSolucaoInicial()
2: repeat
3:    s ← ∅
4:    Inicializa CL
5:    inserido ← true
6:    for all k ∈ K do
7:       rota ← ∅
8:       rota ∪ depositos[k]
9:       rota ∪ Seleção aleatória em CL //Inserido entre os depósitos
10:      Atualiza CL //Remove o elemento de CL
11:      s ∪ rota
12:   while CL ≠ ∅ and inserido do
13:      for all req ∈ CL do
14:         inserido ← false
15:         custo* ← ∞
16:         for all rota ∈ s do
17:            for all pos_c ∈ rota do
18:               if req(coleta) na posição pos_c viola restrições de rota then
19:                  continue
20:               custo ← getCusto(req(coleta), pos_c )
21:               for all pos_e ∈ rota | pos_e > pos_c do
22:                  if req(entrega) na posição pos_e viola restrições de rota then
23:                     continue
24:                  custo ← custo + getCusto(req(entrega), pos_e )
25:                  if custo < custo* then
26:                     custo* ← custo
27:                     inserido ← true
28:                     info ← (req, pos_c, pos_e, rota) //Informações para inserção
29:         if inserido then
30:            //Insere na rota da solução com as informações armazenadas em info
31:            Atualiza CL
32: until CL ≠ ∅
33: return s
```

em movimentos que envolvem uma subsequência de vértices contíguos cuja carga do veículo é igual a zero antes de visitar o primeiro vértice e depois de visitar último vértice desse conjunto, esse tipo de subsequência foi denominado de *blocos de soma zero*. A busca local intercala entre quatro movimentos inter-rota, dois movimentos intra-rota e dois movimentos inter-rota de bloco.

Diversas combinações envolvendo as vizinhanças foram testadas e uma das que demostraram resultados promissores é apresentada a seguir. Devido à instâncias com janelas de tempo muito apertadas, vizinhanças intra-rota muitas vezes geravam rotas que as demais vizinhanças não conseguiam modificar. Por isso, optou-se por executá-las com menor frequência. Além disso, os testes demostraram que as vizinhanças $\mathcal{N}_{blocos}$ podem ser acionadas menos vezes sem perdas da qualidade da solução e economizando tempo computacional.

O Algoritmo 6 ilustra o funcionamento da busca local descrita acima. Primeiro, a estrutura de vizinhança $\mathcal{N}_{blocos}$ é inicializada (linha 3), escolhe-se aleatoriamente uma vizinhança e a remove da estrutura (linhas 5 e 6). Em seguida, encontra-se o melhor vizinho para a solução corrente utilizando a vizinhança selecionada no passo anterior (linha 7). Caso uma melhor solução seja gerada, a estrutura $\mathcal{N}_{blocos}$ é reiniciada (linha 10) e a estrutura $\mathcal{N}_{inter}$ é inicializada (linha 12). Similar aos passos anteriores referentes à $\mathcal{N}_{blocos}$, escolhe-se aleatoriamente uma vizinhança de $\mathcal{N}_{inter}$, remove-a da estrutura e encontra-se o melhor vizinho para a solução corrente (linhas 14–16). Caso obtenha melhora na solução, reinicia-se $\mathcal{N}_{inter}$ (linha 19). Os procedimentos se repetem até que as duas estruturas de vizinhanças fiquem vazias.

Quando *iter* é igual ao I_{ILS}, ou seja, após a aplicação de I_{ILS} perturbações e buscas locais na solução corrente sem haver melhora na função objetivo, o procedimento de busca local utiliza de mais uma estrutura de vizinhança, as vizinhanças intra-rota. A estrutura $\mathcal{N}_{intra}$ é inicializada (linha 23) e assim como nas estruturas apresentadas anteriormente, escolhe-se aleatoriamente uma vizinhança de $\mathcal{N}_{intra}$, remove-se da estrutura e encontra-se o melhor vizinho para a solução corrente (linhas 25–27). Caso ocorra uma melhora na solução, reinicia-se $\mathcal{N}_{intra}$ (linha 30). Mais uma vez, o procedimento se repete enquanto houver elementos em $\mathcal{N}_{intra}$. Se em algum momento a vizinhança $\mathcal{N}_{intra}$ melhorar a solução, o algoritmo retorna ao início (linha 33).

Para cada movimento de melhora da solução, atualiza-se as estruturas de dados auxiliares (do inglês, *Auxiliary Data Structures* – ADSs). As ADSs permitem que operações para encontrar a melhor vizinhança sejam feitas de forma mais eficiente. Mais detalhes sobre as ADSs serão apresentadas na Subseção 4.6.2 e na Seção 4.7.

Algoritmo 6 Busca local

```
1: Procedimento BL(s, iter, I_ILS)
2: Atualiza ADSs
3: Inicializa vizinhanças N_blocos
4: while N_blocos ≠ ∅ do
5:    Escolhe aleatoriamente uma vizinhança N_blocos^(i) ∈ N_blocos
6:    Remove N_blocos^(i) de N_blocos
7:    Encontre melhor vizinho s' ∈ N_blocos^(i)(s)
8:    if f(s') < f(s) then
9:       s ← s'
10:      Reinicializa vizinhanças N_blocos
11:      Atualiza ADSs
12:      Inicializa vizinhanças N_inter
13:      while N_inter ≠ ∅ do
14:         Escolhe aleatoriamente uma vizinhança N_inter^(j) ∈ N_inter
15:         Remove N_inter^(j) de N_inter
16:         Encontre melhor vizinho s' ∈ N_inter^(j)(s)
17:         if f(s') < f(s) then
18:            s ← s'
19:            Reinicializa vizinhanças N_inter
20:            Atualiza ADSs
21: if iter = I_ILS then
22:    f ← f(s)
23:    Inicializa vizinhanças N_intra
24:    while N_intra ≠ ∅ do
25:       Escolhe aleatoriamente uma vizinhança N_intra^(i) ∈ N_intra
26:       Remove N_intra^(i) de N_intra
27:       Encontre melhor vizinho s' ∈ N_intra^(i)(s)
28:       if f(s') < f(s) then
29:          s ← s'
30:          Reinicializa vizinhanças N_intra
31:          Atualiza ADSs
32:    if f(s) < f then
33:       goto linha 3
34: return s
```

4.6.1 Movimentos de vizinhança

Todos os movimentos de vizinhança utilizam o critério de *best improvement*, o que significa dizer que sobre todas as rotas de uma solução, compara-se todas as possibilidades daquele movimento e escolhe-se a que mais diminui os custos da solução. É importante destacar que os movimentos exploram apenas sobre o espaço de soluções viáveis (com exceção das rotas geradas pelos procedimentos de perturbação apresentadas na Seção 4.8). A seguir são detalhados os movimentos utilizados na busca local, foi atribuída a notação i^+ para indicar vértices de coleta, i^- para indicar vértices de entrega de uma requisição i qualquer.

- ***Relocate-inter*** – $\mathcal{N}^{(1)}_{inter}$: Os vértices de coleta e entrega de uma mesma requisição são movidos de uma rota π_1 para outra rota π_2.

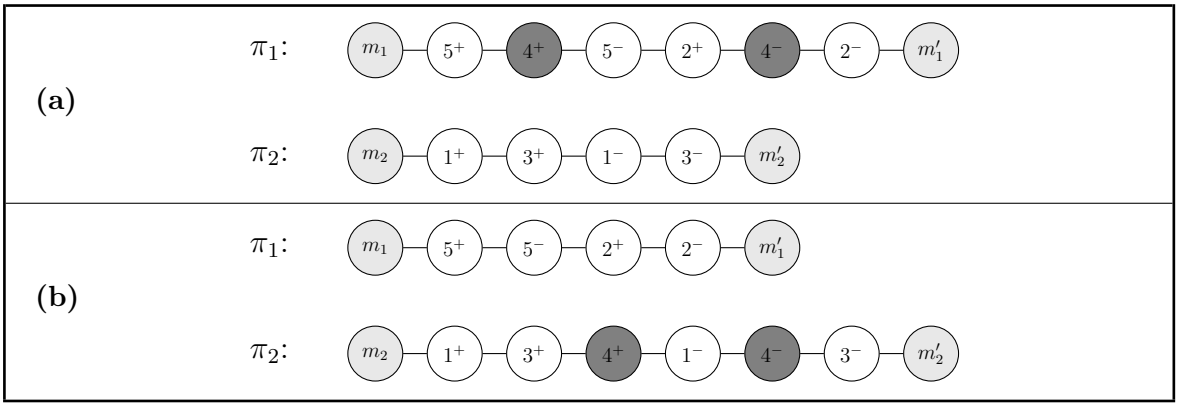

Figura 4.1: **(a)** – É selecionada a requisição 4 pertencente à rota π_1. **(b)** – Os vértices de coleta e entrega da requisição selecionada são inseridos em π_2.

- ***Exchange-inter*** – $\mathcal{N}^{(2)}_{inter}$: Duas requisições, sendo um pertencente a uma rota π_1 e a outra a uma segunda rota π_2, são permutadas. Ou seja, acontece a permuta entre os vértices de coleta e entre os de entrega.

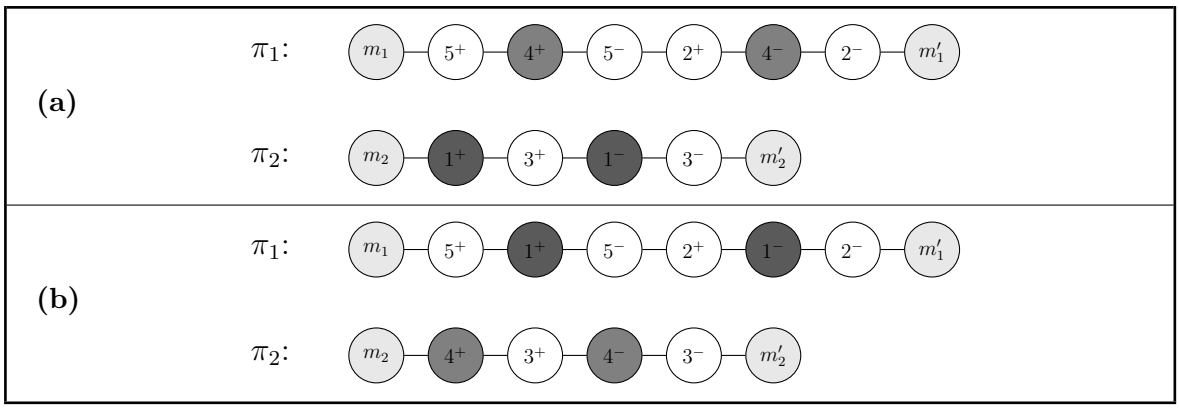

Figura 4.2: **(a)** – É selecionada a requisições 4 pertencente à rota π_1 e a requisição 1 pertencente à π_2. **(b)** – Permutam-se as requisições selecionadas.

- ***2-opt**** – $\mathcal{N}_{inter}^{(3)}$: Dois arcos $(i,\ j)$ e $(u,\ m_1')$ pertencentes a um rota π_1, e dois arcos $(i',\ j')$ e $(u',\ m_2')$ pertencentes a rota π_2 são removidos. Em seguida, criam-se quatro novos arcos: $(i,\ j')$, $(u',\ m_1')$, $(i',\ j)$ e $(u,\ m_2')$. Esse movimento só é realizado quando o arco dos vértices envolvidos tem carga zerada.

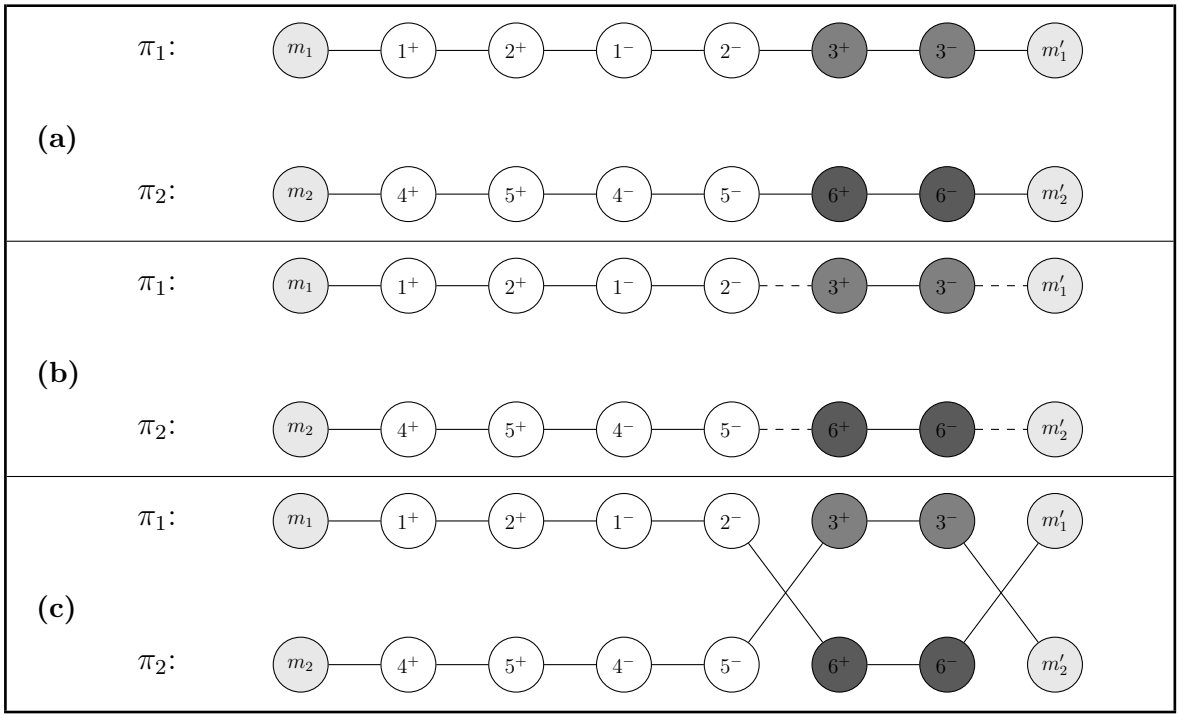

Figura 4.3: **(a)** – Selecionam-se os arcos $(2^-,\ 3^+)$ e $(3^-,\ m_1')$ pertencentes à rota π_1 e os arcos $(5^-,\ 6^+)$ e $(6^-,\ m_2')$ pertencentes à rota π_2. **(b)** – Os arcos selecionados são removidos. **(c)** – Constroem-se os novos arcos $(2^-,\ 6^+)$ e $(6^-,\ m_1')$ para a rota π_1 e $(5^-,\ 3^+)$ e $(3^-,\ m_2')$ para a rota π_2.

- ***Exchange-vehicle*** – $\mathcal{N}_{inter}^{(4)}$: Permutam-se os veículos entre um par de rotas π_1 e π_2.

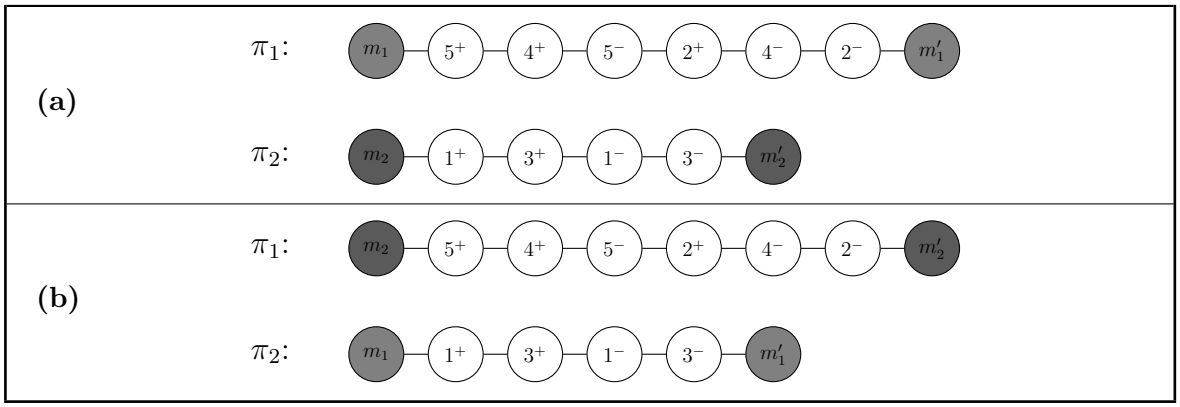

Figura 4.4: **(a)** – Selecionam-se os depósitos de uma rota π_1 e de outra rota π_2. **(b)** – Permutam-se os depósitos e os veículos envolvidos.

- ***Relocate-intra*** – $\mathcal{N}_{intra}^{(1)}$: As posições dos vértices de coleta e entrega de uma mesma requisição são redefinidas dentro da mesma rota.

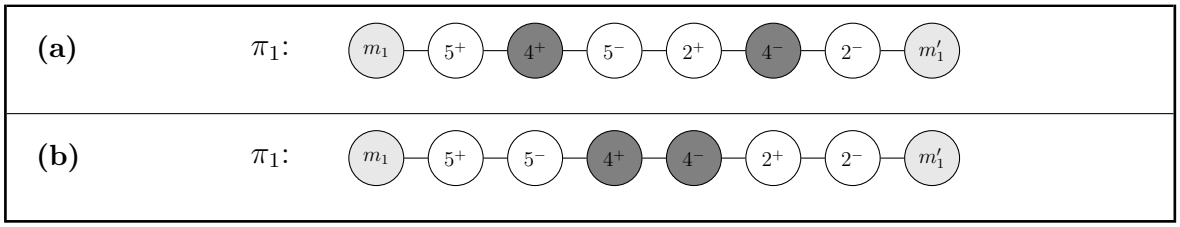

Figura 4.5: **(a)** – A requisição 4 é selecionada. **(b)** – Reinsere os dois vértices da requisição em novas posições em π_1

- ***Exchange-intra*** – $\mathcal{N}_{intra}^{(2)}$: As posições de dois vértices de uma mesma rota são permutadas, podendo eles serem de requisições ou tipos (coleta ou entrega) diferentes.

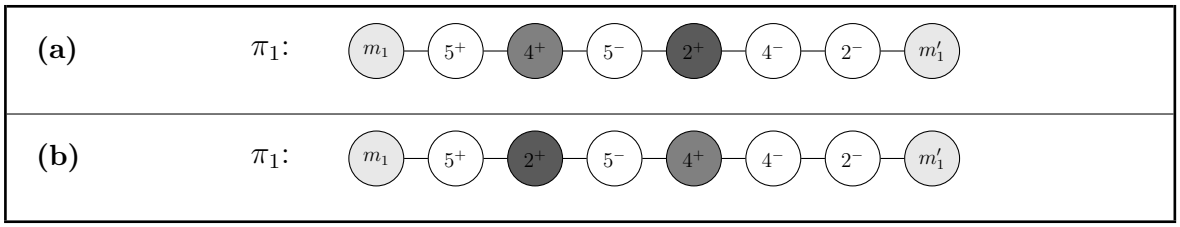

Figura 4.6: **(a)** – Selecionam-se os vértices 4^+ e 2^+. **(b)** – Permutam-se os vértices selecionados

- ***Relocate-block*** – $\mathcal{N}^{(1)}_{blocos}$: Um bloco de soma zero é movido de uma rota π_1 para outra rota π_2.

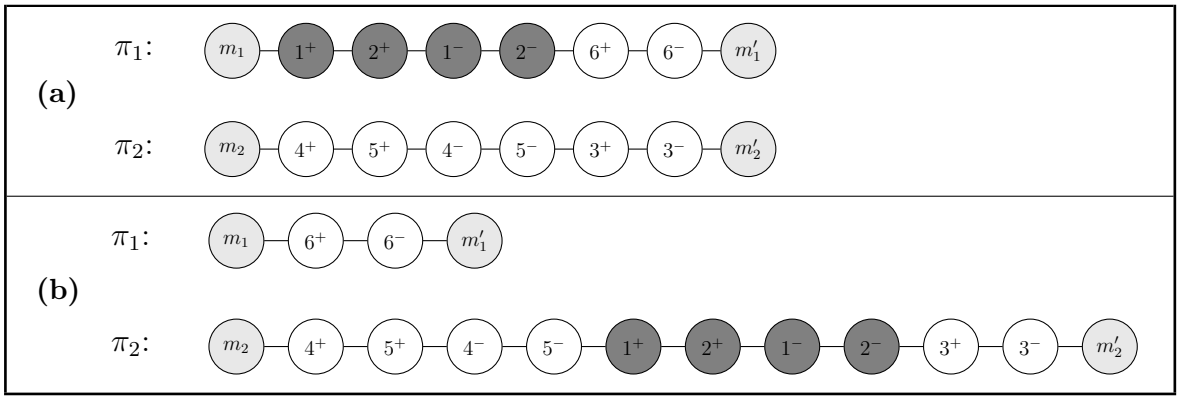

Figura 4.7: **(a)** – Seleciona o bloco de soma zero (1^+, ..., 2^-) em π_1. **(b)** – O bloco selecionado é inserido em π_2

- ***Exchange-block*** – $\mathcal{N}^{(2)}_{blocos}$: Dois blocos de soma zero, sendo um pertencente a uma rota π_1 e a outra a uma rota π_2, são permutadas. Os blocos podem possuir tamanhos diferentes para cada uma das rotas.

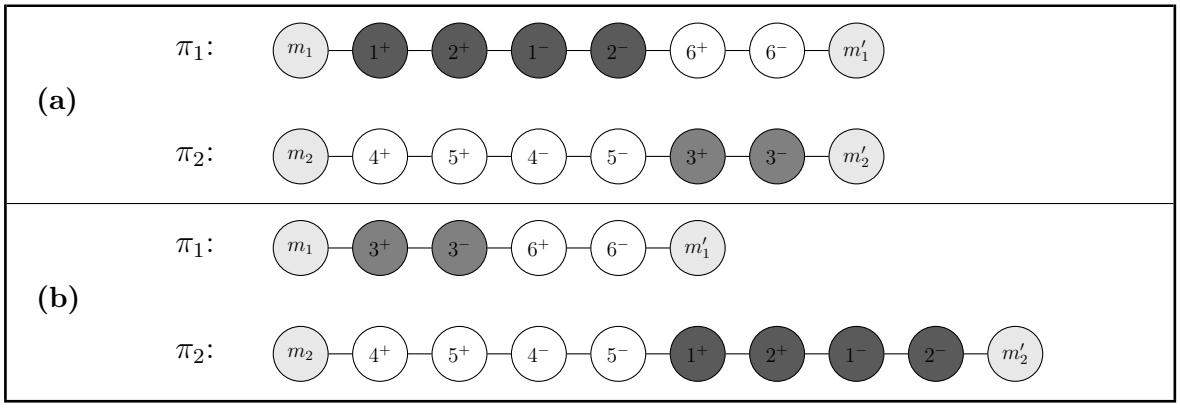

Figura 4.8: **(a)** – Selecionam o bloco de soma zero (1^+, ..., 2^-) em π_1 e outro bloco de soma zero (3^+, 3^-) em π_2. **(b)** – Os blocos selecionados são permutados

As vizinhanças pertencentes às estruturas $\mathcal{N}_{inter}$ e $\mathcal{N}_{blocos}$ são aplicadas para cada par de rotas. Já as vizinhanças pertencentes à $\mathcal{N}_{intra}$, são aplicadas em cada rota. A Tabela 4.1 apresenta o tamanho das vizinhanças de cada movimento proposto.

Tabela 4.1: Tamanho das vizinhanças

$\mathcal{N}_{inter}$	**Tamanho**	$\mathcal{N}_{intra}$	**Tamanho**	$\mathcal{N}_{block}$	**Tamanho**
Relocate-inter	$\mathcal{O}(n^3)$	*Relocate-intra*	$\mathcal{O}(n^3)$	*Relocate-block*	$\mathcal{O}(n^3)$
Exchange-inter	$\mathcal{O}(n^2)$	*Exchange-intra*	$\mathcal{O}(n^2)$	*Exchange-block*	$\mathcal{O}(n^4)$
*2-opt**	$\mathcal{O}(n^2)$				
Exchange-vehicle	$\mathcal{O}(v^2)$				

4.6.2 Avaliação de movimentos

Uma quantidade exaustiva de comparações é realizada durante os movimentos de vizinhança, é importante que essas avaliações sejam realizadas da forma mais eficiente possível. Idealmente com complexidade de tempo constante. As avaliações devem checar não só os novos custos de remoção/inserção de uma requisição, como também verificar a viabilidade das rotas envolvidas.

As restrições de paridade e precedência das requisições são checadas de forma direta. Dado que a construção inicial e os movimentos de busca sempre inserem o par coleta e entrega na mesma operação, onde a posição da entrega na rota é sempre maior do que a posição de coleta na mesma rota. Para as restrições de carga e janela de tempo, o algoritmo utiliza estruturas de dados auxiliares chamadas de *subsequências* que fazem parte das ADSs. As *subsequências* são construídas e pré-processadas em $\mathcal{O}(n^2)$ para cada rota. Elas permitem avaliações em tempo constante durante as buscas, a técnica é baseada nos trabalhos de Vidal et al. (2013) e Bulhões et al. (2018).

Seja $\sigma = (\sigma^0, \ldots, \sigma^{|\sigma|-1})$ uma subsequência, e σ^{ij} uma subsequência que começa na i-ésima posição e termina na j-ésima posição, i.e., $\sigma^{ij} = (\sigma^i, \ldots, \sigma^j)$. Para cada possível σ de uma rota as estruturas armazenam e atualizam os seguintes valores:

- $q^r_{sum}(\sigma)$: soma das cargas de coleta/entrega (carga acumulada) para cada recurso $r \in R$;
- $q^r_{max}(\sigma)$: máxima carga acumulada para cada recurso $r \in R$;
- $T(\sigma)$: mínima duração para percorrer σ;
- $TW(\sigma)$: mínimo *time-warp*, onde *time-warp* é a quantidade de tempo excedida do final da janela de tempo;

- $E(\sigma)$: horário mais cedo de visita ao primeiro vértice tal que a duração e o *time-warp* sejam mínimos;

- $L(\sigma)$: horário mais tarde de visita ao primeiro vértice tal que a duração e o *time-warp* sejam mínimos;

Se a subsequência σ' é composta apenas pelo vértice i qualquer, então $q^r_{sum}(\sigma') = q^r_i$, $q^r_{max}(\sigma') = max(0, q^r_i)$, $T(\sigma') = d_i$, $TW(\sigma') = 0$, $E(\sigma') = e_i$, $L(\sigma') = l_i$. Toma-se o operador $\oplus$ como sinal de concatenação entre duas subsequências distintas. A seguir é apresentado que qualquer subsequência σ, tal que $|\sigma| > 1$, pode ser derivada de duas outras subsequências σ^1 e σ^2, a operação de concatenação é caracterizada pela seguinte atualização de dados:

$$q^r_{sum}(\sigma^1 \oplus \sigma^2) = q^r_{sum}(\sigma^1) + q^r_{sum}(\sigma^2), \forall r \in R \tag{4.5}$$

$$q^r_{max}(\sigma^1 \oplus \sigma^2) = max\{q^r_{max}(\sigma^1), q^r_{sum}(\sigma^1) + q^r_{max}(\sigma^2)\}, \forall r \in R \tag{4.6}$$

$$T(\sigma^1 \oplus \sigma^2) = T(\sigma^1) + T(\sigma^2) + t_{\sigma^1(|\sigma^1|)\sigma^2(1)} + \Delta_{WT} \tag{4.7}$$

$$E(\sigma^1 \oplus \sigma^2) = max\{E(\sigma^2) - \Delta, E(\sigma^1)\} - \Delta_{WT} \tag{4.8}$$

$$L(\sigma^1 \oplus \sigma^2) = min\{L(\sigma^2) - \Delta, L(\sigma^1)\} + \Delta_{TW} \tag{4.9}$$

$$TW(\sigma^1 \oplus \sigma^2) = TW(\sigma^1) + TW(\sigma^2) + \Delta_{TW}, \tag{4.10}$$

onde:

$$\Delta = T(\sigma^1) - TW(\sigma^1) + t_{\sigma^1(|\sigma^1|)\sigma^2(1)} \tag{4.11}$$

$$\Delta_{WT} = max\{E(\sigma^2) - \Delta - L(\sigma^1), 0\} \tag{4.12}$$

$$\Delta_{TW} = max\{E(\sigma^1) + \Delta - L(\sigma^2), 0\}. \tag{4.13}$$

Uma rota é viável para carga e janela de tempo se sua subsequência $\sigma = (\sigma^0, \ldots, \sigma^{|\sigma|-1})$ possui $TW = 0$ e $q^r_{max} \leq C^{rk}$ para todo $r \in R$, onde k é o veículo associado à rota.

A Figura 4.9 ilustra um exemplo para duas concatenações com respeito as atualizações dos dados das janelas de tempo. Em **(a)**, as operações de concatenação são feitas entre as subsequências $\sigma^1 \oplus \sigma^2$. Em **(b)**, as concatenações acontecem entre as sub-

sequências $\sigma^{12} \oplus \sigma^3$, onde $\sigma^{12} = \sigma^1 \oplus \sigma^2$. Durante a concatenação entre $\sigma^1 \oplus \sigma^2$ nota-se que, para esse caso, a subsequência σ^1 deve ser atendida o mais cedo possível para resultar no mínimo *time-warp*. Como consequência, o $L(\sigma^{12})$ deve ser igual ao $E(\sigma^{12})$, ou seja, a subsequência é obrigada a sair o mais cedo possível. Na segunda concatenação, entre $\sigma^{12} \oplus \sigma^3$, a subsequência sempre chega antes do $E(\sigma^3)$, sendo obrigada a esperar até o começo da janela de tempo da subsequência seguinte, como é representado em $WT(\sigma^{13})$. Em **(a)**, é importante destacar que o valor do *time-warp* não é utilizado para calcular o valor de $T(\sigma^{12})$. Já em **(b)**, o valor do tempo de espera gerado pela concatenação é utilizado para calcular o valor de $T(\sigma^{13})$. Ao final, o resultado das três concatenações geram uma rota inviável devido ao *time-warp* gerado pela concatenação em **(a)**.

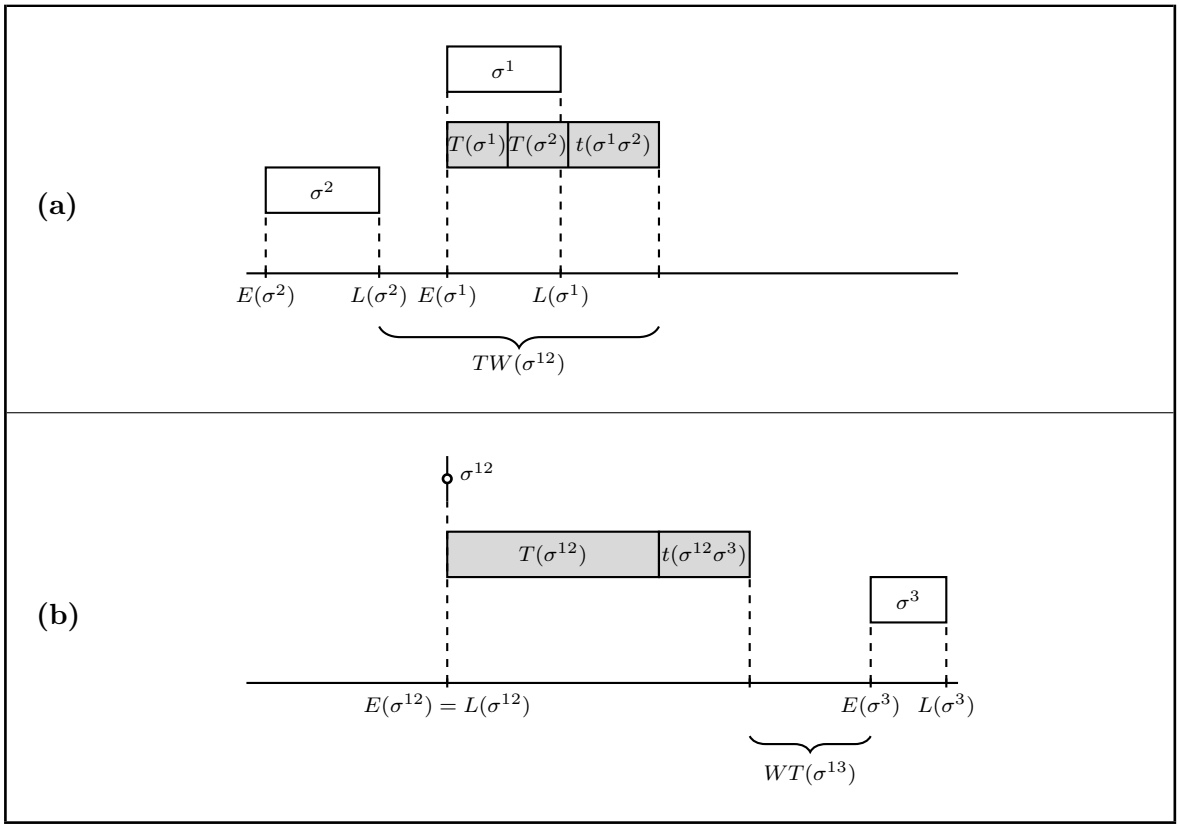

Figura 4.9: Operações de concatenações entre subsequencias com informações das janelas de tempo. Em **(a)** gerando *time-warp* e em **(b)** gerando tempo de espera.

Uma vez que as rotas se tornam viáveis para as restrições de janela de tempo e carga, o procedimento de checagem de tempo máximo de viagem para cada requisição e duração da rota é chamado. O uso desse procedimento é necessário pois os tempos de

viagem podem ser reduzidos sem permutação dos vértices, fazendo apenas uma espera antes de começar o serviço no vértice de coleta. Por isso, é necessário computar o quanto cada vértice de coleta pode esperar ser atendido sem que aconteça violação no seu tempo máximo de viagem.

Seja $\mathcal{W}_i$ o tempo necessário de espera até o começo da janela de tempo do vértice i, l_i o final da janela de tempo de i, $\mathcal{B}_i$ o tempo que indica o começo do serviço no vértice i, L_i o tempo máximo de viagem da requisição i e $\mathcal{P}_i$ o tempo de viagem entre a coleta e a entrega de i, por último, seja m o vértice que representa o depósito inicial e m' o vértice que representa um depósito final para uma rota π, onde $m \in M_0$ e $m' \in M_e$. Cordeau e Laporte (2003b) calculam o quanto um vértice pode atrasar a partida (*forward time slack* – F_i) pela equação:

$$F_i = \min_{i \leq j \leq m'} \{ \sum_{i < p \leq j} \mathcal{W}_p + (min\{l_j - \mathcal{B}_j, L_j - \mathcal{P}_j\})^+ \} \tag{4.14}$$

Caso j seja um vértice de coleta, então $\mathcal{P}_j = -\infty$.

O algoritmo computa o *Forward Time Slack* (FTS) para cada vértice da rota com o objetivo de verificar a viabilidade quanto a restrição de máximo tempo de viagem. Durante o movimento de vizinhança, o procedimento deve atualizar a rota com a inserção do par coleta/entrega. Em seguida o FTS para cada vértice é computado pelo seguinte algoritmo:

Algoritmo 7 Forward-time-slack

1: Procedimento FTS(π)
2: Compute $\mathcal{B}_i$ e $\mathcal{W}_i$, $i \in \pi$, com base no atendimento o mais cedo possível
3: Compute F_m
4: $\mathcal{B}_m \leftarrow e_m + min\{F_m, \sum_{0<p<m'} \mathcal{W}_p\}$
5: Atualize os novos valores de $\mathcal{B}_i$ e $\mathcal{W}_i$, $i \in \pi$, com a mudança do $\mathcal{B}_m$
6: Compute $\mathcal{P}_i$ para todas as requisições
7: **for all** $i \in \pi \mid i$ é vértice de coleta **do**
8: Compute F_i
9: $\mathcal{B}_i \leftarrow \mathcal{B}_i + min\{F_i, \sum_{i<p<m'} \mathcal{W}_p\}$
10: Atualize os novos valores de $\mathcal{B}_j$ e $\mathcal{W}_j$, para todo j que vem depois de i
11: Compute $\mathcal{P}_j$ para todos os pares de requisição j que vem depois de i
12: **return** false

O algoritmo primeiro atualiza as suas estruturas de dados seguindo a abordagem

que atende as requisições sempre o mais cedo possível (linha 2). Em seguida o algoritmo computa o quanto a saída do depósito de origem pode ser atrasada para diminuir a duração total da rota. Isso pode mudar o horário de visita dos vértices subsequentes, logo, precisa-se atualizar novamente as estruturas (linhas 3–6). Por último, o algoritmo itera sobre os vértices da rota calculando o quanto cada coleta pode atrasar a sua partida. Após cada cálculo, as estruturas dos vértices seguintes são atualizadas (linhas 7–11). Ao computar os valores de tempo de viagem de cada requisição (linhas 6 e 11), verifica se a rota é viável, se sim, o algoritmo é finalizado. Caso não seja possível viabilizar todos os tempos máximos de viagem, o algoritmo retorna que a rota é inviável (linha 12).

As verificações para viabilidade da capacidade e da janela de tempo possuem complexidade de tempo constantes, porém são necessários pré-processamento de estruturas auxiliares em $\mathcal{O}(n^2)$ sempre que a solução é modificada. O algoritmo para verificação de tempo máximo de viagem possui complexidade de tempo de $\mathcal{O}(n^2)$ e é independe de pré-processamentos. Com essas complexidades e as informações da Tabela 4.1, a Tabela 4.2 apresenta a complexidade de cada vizinhança sem levar em consideração a atualização das ADSs.

Tabela 4.2: Complexidade das vizinhanças

Vizinhança	**Complexidade**
Relocate-inter	$\mathcal{O}(n^5)$
Exchange-inter	$\mathcal{O}(n^4)$
*2-opt**	$\mathcal{O}(n^4)$
Exchange-vehicle	$\mathcal{O}(v^2n^2)$
Relocate-intra	$\mathcal{O}(n^5)$
Exchange-intra	$\mathcal{O}(n^4)$
Relocate-block	$\mathcal{O}(n^5)$
Exchange-block	$\mathcal{O}(n^6)$

4.7 Mecanismos de aceleração

Com o objetivo de melhorar a eficiência do algoritmo, são propostos mecanismos de aceleração. Esses mecanismos são estruturas de dados pertencentes às ADSs que fornecem informações sobre soluções ou vizinhanças já exploradas durante a execução

do algoritmo. Um outro mecanismo de aceleração diz repeito ao intervalo de posições viáveis para janela de tempo que um vértice pode ser inserido. A seguir são detalhados os mecanismos de aceleração propostos no trabalho.

- *Memoization Move Descriptor* (MMD) — O primeiro mecanismo é uma extensão daquele proposto por Zachariadis e Kiranoudis (2010), denominado *Static Move Descriptor*, que foi posteriormente aprimorado por Beek et al. (2018). A estrutura guarda, para cada movimento de busca local e para cada rota (ou par de rotas), as informações de quais vértices e para quais posições devem ir tal que forneçam o melhor movimento viável. Em outras palavras, uma vez que uma rota (ou par de rotas) é analisada por um movimento, armazenam-se os vértices e posições que geram uma solução viável de menor custo. Caso o movimento não gere nenhuma melhoria para aquela rota (ou pares de rotas), a informação também é salva. Dessa forma, antes da avaliação de um movimento, verifica-se se aquela rota já foi analisada em algum momento do algoritmo. Se sim, a MMD retorna a informação do melhor movimento previamente checado (ou que não existe melhoria possível). Caso contrário, a avaliação do movimento é feita normalmente e, ao final, armazena-se a informação do movimento para a rota (ou pares de rotas) que foi avaliado.

- *Memoization Solution Descriptor* (MSD) — A mesma ideia do mecanismo MMD pode ser estendida para salvar soluções já encontradas pelo ILS-SP_{DARP}. A estrutura permite salvar soluções ótimas locais que já foram visitadas pelo algoritmo em algum momento de sua execução, junto com a solução salva-se também a iteração corrente. Caso o ILS-SP_{DARP} encontre novamente uma solução já armazenada na MSD e cuja a iteração salva seja diferente da iteração corrente, a iteração é abortada. Isso permite que a heurística não fique realizando buscas a partir de soluções ótimas locais previamente visitadas e que, ou alguma perturbação já levou até outro ótimo local, ou o número máximo de perturbações não foram suficientes para escapar do ótimo local. O MSD também pode ser usado durante o procedimento de buscas locais, uma vez que seja encontrada uma solução e, sabe-se que ela é ótima local, não são necessárias outras checagens de

vizinhanças para tentar melhorá-la. Modificações dos Algoritmos 3 e 6 utilizando o mecanismo MSD são apresentados abaixo.

Algoritmo 8 ILS_{DARP} (com MSD)

```
1: Procedimento ILS_DARP (I_ILS, I_ρ, pool, pool_temp, s)
2: s* ← s
3: f* ← f(s)
4: iter ← 0
5: while iter ≤ I_ILS do
6:    s ← BL(s, iter, I_ILS)
7:
8:
9:
10:
11:   pool ← pool ∪ rotas de s
12:   pool_temp ← pool_temp ∪ rotas de s
13:   if f(s) < f* then
14:      s* ← s
15:      f* ← f(s)
16:      iter ← 0
17:   s ← perturbar(s*, I_ρ)
18:   iter ← iter + 1
19: return s*
```

```
1: Procedimento ILS^MSD_DARP (I_ILS, I_ρ, pool, pool_temp, s)
2: s* ← s
3: f* ← f(s)
4: iter ← 0
5: while iter ≤ I_ILS do
6:    s ← BL(s, iter, I_ILS)
7:    if s ∈ MSD and MSD(s) ≠ iter then
8:       return s
9:    else if s ∉ MSD then
10:      MSD ← (s, iter)
11:      pool ← pool ∪ rotas de s
12:      pool_temp ← pool_temp ∪ rotas de s
13:   if f(s) < f* then
14:      s* ← s
15:      f* ← f(s)
16:      iter ← 0
17:   s ← perturbar(s*, I_ρ)
18:   iter ← iter + 1
19: return s*
```

As modificações do Algoritmo 3 são apresentadas no Algoritmo 8 nas linhas 7–12. Se a solução retornada pela busca local já foi encontrada em uma iteração anterior à corrente (linha 7), o procedimento é abortado. Se a solução após a busca local é inédita (linha 9), inclui-se a informação da solução e a iteração corrente no MSD (linha 10). No Algoritmo 9 são apresentadas as modificações do Algoritmo 6, as checagens do MSD acontecem sempre após encontrar o melhor vizinho para uma solução (linhas 8, 19 e 32). Caso essa solução já esteja armazenada no MSD, significa que foi encontrado um ótimo local previamente visitado, portanto a busca local é interrompida.

- *Feasible Search Range* (FSR) — O último mecanismo aproveita-se das propriedades das janelas de tempo dos vértices. Quando dois vértices possuem janelas de tempo disjuntas entre si, pode-se afirmar que um dos vértices deve vir antes (ou depois) do outro caso pertençam à mesma rota. Essas informações sobre a disjunção de duas janelas de tempo entre todos os pares vértices podem ser ar-

Algoritmo 9 Busca local (com MSD)

```
1: Procedimento BL(s, iter, I_ILS)
2: Atualiza ADSs
3: Inicializa vizinhanças N_blocos
4: while N_blocos ≠ ∅ do
5:    Escolhe aleatoriamente uma vizinhança N_blocos^(i) ∈ N_blocos
6:    Remove N_blocos^(i) de N_blocos
7:    Encontre melhor vizinho s' ∈ N_blocos^(i)(s)
8:
9:
10:   if f(s') < f(s) then
11:      s ← s'
12:      Reinicializa vizinhanças N_blocos
13:      Atualiza ADSs
14:      Inicializa vizinhanças N_inter
15:      while N_inter ≠ ∅ do
16:         Escolhe aleatoriamente uma vizinhança N_inter^(j) ∈ N_inter
17:         Remove N_inter^(j) de N_inter
18:         Encontre melhor vizinho s' ∈ N_inter^(j)(s)
19:
20:
21:         if f(s') < f(s) then
22:            s ← s'
23:            Reinicializa vizinhanças N_inter
24:            Atualiza ADSs
25: if iter = I_ILS then
26:    f ← f(s)
27:    Inicializa vizinhanças N_intra
28:    while N_intra ≠ ∅ do
29:       Escolhe aleatoriamente uma vizinhança N_intra^(i) ∈ N_intra
30:       Remove N_intra^(i) de N_intra
31:       Encontre melhor vizinho s' ∈ N_intra^(i)(s)
32:
33:
34:       if f(s') < f(s) then
35:          s ← s'
36:          Reinicializa vizinhanças N_intra
37:          Atualiza ADSs
38:    if f(s) < f then
39:       goto linha 3
40: return s
```

```
1: Procedimento BL^MSD(s, iter, I_ILS)
2: Atualiza ADSs
3: Inicializa vizinhanças N_blocos
4: while N_blocos ≠ ∅ do
5:    Escolhe aleatoriamente uma vizinhança N_blocos^(i) ∈ N_blocos
6:    Remove N_blocos^(i) de N_blocos
7:    Encontre melhor vizinho s' ∈ N_blocos^(i)(s)
8:    if s' ∈ MSD then
9:       return s'
10:   if f(s') < f(s) then
11:      s ← s'
12:      Reinicializa vizinhanças N_blocos
13:      Atualiza ADSs
14:      Inicializa vizinhanças N_inter
15:      while N_inter ≠ ∅ do
16:         Escolhe aleatoriamente uma vizinhança N_inter^(j) ∈ N_inter
17:         Remove N_inter^(j) de N_inter
18:         Encontre melhor vizinho s' ∈ N_inter^(j)(s)
19:         if s ∈ MSD then
20:            return s'
21:         if f(s') < f(s) then
22:            s ← s'
23:            Reinicializa vizinhanças N_inter
24:            Atualiza ADSs
25: if iter = I_ILS then
26:    f ← f(s)
27:    Inicializa vizinhanças N_intra
28:    while N_intra ≠ ∅ do
29:       Escolhe aleatoriamente uma vizinhança N_intra^(i) ∈ N_intra
30:       Remove N_intra^(i) de N_intra
31:       Encontre melhor vizinho s' ∈ N_intra^(i)(s)
32:       if s' ∈ MSD then
33:          return s'
34:       if f(s') < f(s) then
35:          s ← s'
36:          Reinicializa vizinhanças N_intra
37:          Atualiza ADSs
38:    if f(s) < f then
39:       goto linha 3
40: return s
```

mazenadas durante o pré-processamento de dados. Utilizando essas informações, é possível identificar para cada rota qual é a primeira e a última posição viável para as restrições de janela de tempo de todos os vértices de entrada do problema. Assim, a FSR armazena para cada rota e para cada vértice qual a primeira e a última posição viável em janela de tempo que esse vértice poderá aparecer. A FSR é utilizada durante as avaliações de movimentos com o objetivo de fornecer um intervalo mais estreito em que um novo vértice poderá aparecer na rota.

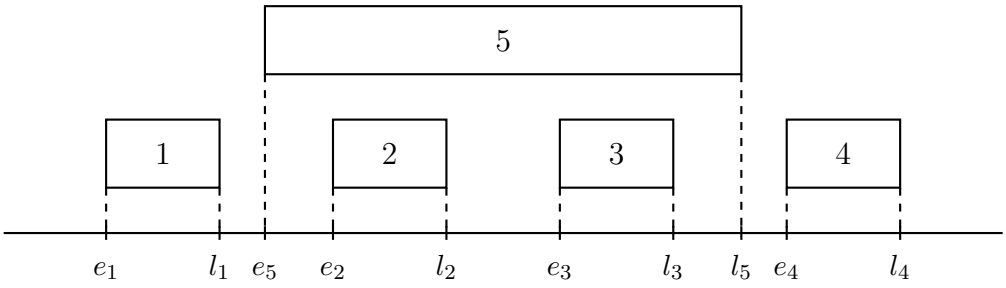

Figura 4.10: A janela de tempo de 5 é disjunta das janelas de tempo de 1 e 4, uma vez que $e_5 > l_1$ e $l_5 < e_4$. Porém, a janela de tempo de 5 não é disjunta das janelas de tempo dos vértices 2 e 3. Por isso, pode-se afirmar que 5 deve vir após 1 e antes de 4.

As implementações dos MMD e MSD foram feitas utilizando árvores binárias de busca. Para um melhor funcionamento dessa estrutura de dados cada elemento na MMD e MSD (rota, pares de rota ou solução) deve ser representado por uma chave. A chave deve garantir que cada elemento respectivo à sua estrutura seja único. As árvores binárias de busca permitem que operações de busca de elemento, checagem de existência de elemento e inserção de elemento tenham complexidade de tempo de $\mathcal{O}(\log \tau)$, onde τ é a quantidade de elementos salvos na árvore.

A estrutura que identifica a precedência entre dois vértices com janelas de tempo disjuntas é preenchida durante o pré-processamento dos dados e tem complexidade de tempo de $\mathcal{O}(n^2)$. Para cada nova rota é necessária uma atualização sobre a FSR, essa atualização tem complexidade de tempo de $\mathcal{O}(n^2)$. Durante a avaliação de um movimento, o intervalo fornecido pela FSR de um vértice em uma rota é recuperado em $\mathcal{O}(1)$.

4.8 Perturbação

Dois tipos de perturbações foram implementadas para o Algoritmo 3. A decisão de qual tipo de perturbação será realizada é feita de forma aleatória e com mesma probabilidade. No primeiro método escolhe-se aleatoriamente uma rota π. Em seguida, remove-se um par de vértices coleta/entrega escolhido de forma aleatória. Esse par é inserido em posições aleatórias em uma rota extra π_e. No segundo método, escolhe-se aleatoriamente uma rota π e uma requisição pertencente à π. Em seguida, remove-se o menor bloco de soma zero que contém essa requisição. O bloco é inserido em uma posição aleatória na rota extra π_e. A rota π_e possui uma penalidade proporcional ao seu custo, permitindo que as restrições de carga e janela de tempo sejam violadas. São realizadas uma quantidade aleatória entre 1 e I_ρ operações de perturbação sobre a solução. Ao final, a rota extra é inserida na solução corrente, portanto, as buscas locais são realizadas também em π_e, que naturalmente ficará vazia por causa do seu custo penalizado.

Capítulo 5

Experimentos computacionais

Este capítulo descreve os experimentos computacionais realizados para algoritmo proposto, analisando os componentes que o compõe e comparando os resultados do método com outros trabalhos relacionados. Todos os experimentos foram executados em um computador com processador Intel(R) Core(TM) i5-9600KF CPU 3.70 GHz e 8 GB de memória RAM. A máquina utiliza o sistema operacional Linux Ubuntu 18.04. Os métodos exatos foram implementados com o auxílio do resolvedor CPLEX 12.7.

Os experimentos foram executados sobre um conjunto de instâncias da literatura dos problemas HDARP e MDHDARP propostas por Parragh (2011) e Braekers et al. (2014). Essas instâncias são agrupadas em três conjuntos, U que utiliza apenas um único recurso de demanda/capacidade, E e I que utilizam até quatro recursos de demanda/capacidade. A diferença entre os grupos E e I é que no primeiro todos os veículos tem a mesma capacidade, já o segundo, os veículos podem ter capacidades diferentes. As instâncias variam de 2 até 8 veículos e de 16 até 96 requisições, onde todos os vértices do grafo são localizados em um plano com limites $[-10, 10]^2$. Para o HDARP, o depósito é localizado no centro do quadrado, enquanto que para o MDHDARP os depósitos são divididos em 4 pontos com as seguintes coordenadas: $[-5, -5]$, $[5, 5]$, $[-5, 5]$ e $[5, -5]$, onde o primeiro e o quinto veículo pertencem ao primeiro depósito, o segundo e o sexto veículo ao segundo depósito, e assim por diante.

Um subconjunto das instâncias foi selecionado para os testes computacionais de

calibração de parâmetros e de análise de procedimentos do algoritmo proposto. Os critérios utilizados para a escolha desse subconjunto foram, instâncias com mais que dois veículos e com um número de requisições maior ou igual que dez vezes a quantidade de veículos. O motivo da escolha desses critérios se deu ao fato de que, para as demais instâncias, soluções de alta qualidade são encontradas facilmente por outros trabalhos da área (Parragh, 2011; Braekers et al., 2014; Masmoudi et al., 2017). O subconjunto contém 72 instâncias das 144, onde metade está relacionadas ao HDARP e a outra metade ao relacionadas ao MDHDARP.

5.1 Impacto das vizinhanças de blocos

As vizinhanças pertencentes ao conjunto $\mathcal{N}_{blocos}$ foram propostas pela primeira vez para o problema do MDHDARP. Com o objetivo de descobrir seu impacto na busca local, realizaram-se experimentos computacionais de todas as combinações de configurações possíveis desse conjunto de vizinhanças. Para cada combinação, foram feitas 10 execuções do algoritmo proposto sem os procedimentos de SP e com os parâmetros $I_R = 1$ e $I_{ILS} = 0$. Isso significa que, em cada uma das execuções apenas uma busca local sobre solução gerada pelo procedimento construtivo foi aplicada, sem passar por nenhuma operação de perturbação. Os resultados desses testes são apresentados na Tabela 5.1, a primeira coluna indica as combinações de vizinhanças aplicada. Em seguida, o *gap* médio agregado das instâncias testadas em relação aos melhores limites inferiores (do inglês, *Lower Bound* – LB) disponíveis na literatura. A última coluna representa o tempo médio agregado das instâncias.

Tabela 5.1: Impacto da vizinhança $\mathcal{N}_{blocos}$

Combinação	**Gap médio** (%)	**Tempo (s)**
Sem $\mathcal{N}_{block}$	7,46%	0,026
$\mathcal{N}^{(1)}_{block}$	7,32%	0,031
$\mathcal{N}^{(2)}_{block}$	6,61%	0,033
$\mathcal{N}^{(1)}_{block} + \mathcal{N}^{(2)}_{block}$	**6,48%**	0,033

Os testes apresentados na Tabela 5.1 demostraram que as vizinhanças possuem um impacto na diminuição do *gap* médio. Quando utilizadas individualmente, a vizinhança

$\mathcal{N}^{(2)}_{blocos}$ obteve um menor *gap* médio e maior custo de tempo do que $\mathcal{N}^{(1)}_{blocos}$, o motivo para isso pode ser atribuído ao fato de que $\mathcal{N}^{(2)}_{blocos}$ é uma vizinhança com mais possibilidades para exploração do que $\mathcal{N}^{(1)}_{blocos}$, conforme apresentado na Tabela 4.1. A combinação que forneceu o menor *gap* foi utilizando as duas vizinhanças, além disso, o tempo de execução foi similar aos testados com as vizinhanças individualmente.

5.2 Impacto dos mecanismos de perturbação

Foram realizados testes computacionais em três versões diferentes de procedimentos de perturbação. A primeira versão (ρ_1) consiste em escolher aleatoriamente uma rota π, em seguida, escolher aleatoriamente uma requisição de π e inseri-la em uma rota extra π_e com custo penalizado. Para a segunda versão (ρ_2), assim como na primeira, escolhe-se aleatoriamente uma rota π e uma requisição de π, em seguida, remove-se o menor bloco de soma zero que contém essa requisição. O bloco escolhido é inserido em uma rota extra π_e com custo penalizado. Por último, a terceira versão ($\rho_1 + \rho_2$) combina as duas abordagem anteriores, a cada execução, escolhe-se aleatoriamente qual das duas versões será utilizada.

Os experimentos levaram em consideração os três métodos de perturbação e a variação do parâmetro I_ρ de 2 até 7. Onde, o parâmetro I_ρ indica que uma quantidade aleatória entre 1 e I_ρ operações de perturbação será aplicada sobre uma solução. Assim como os experimentos da Seção 5.1, foram realizadas 10 execuções do algoritmo proposto sem os procedimentos de SP e com a configuração $I_R = 1$. Entretanto, o valor definido para I_{ILS} foi de $2 \times n$, ou seja, proporcional à quantidade de requisições da instância. Os resultados obtidos foram reportados no gráfico da Figura 5.1. Cada configuração de teste representa um ponto do gráfico, conforme explicado na legenda da figura. Os eixos x e y indicam respectivamente o tempo médio em segundos e o *gap* médio em relação aos melhores LBs disponíveis na literatura.

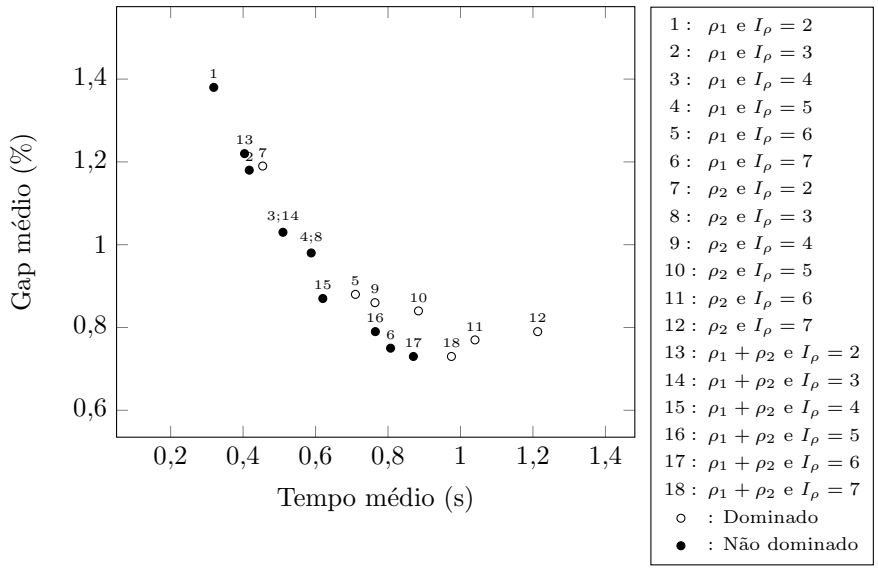

Figura 5.1: Impacto dos procedimentos de perturbação

O gráfico mostra que os pontos 5 e 7 associados à perturbação ρ_1, os pontos 9, 10, 11 e 12 associados à ρ_2, e o ponto 18 associado à $\rho_1 + \rho_2$ são dominados por outras configurações. Desconsiderando os pontos dominados, os menores valores de *gap* médio foram encontrados pelos pontos 6, 16 e 17, porém, eles apresentaram um valor de tempo próximos à $0,8$ segundos. Enquanto que os pontos 1, 2, 3, 13 e 14 encontraram os menores valores em relação ao tempo, mas um *gap* médio acima de 1%. A configuração indicada pelo ponto 15, representando $\rho_1 + \rho_2$ e $I_\rho = 4$, não foi dominado por nenhum outro ponto e demostrou um bom balanceamento entre *gap* e tempo. Por isso, escolheu-se os valores representados por 15 como parâmetros para os próximos testes do algoritmo.

5.3 Calibração dos parâmetros I_R e I_{ILS}

Após a calibração dos mecanismos de perturbação, os experimentos seguintes tiveram como objetivo encontrar a melhor combinação de configuração para os valores de I_R e I_{ILS} para o algoritmo proposto. Mais uma vez, utilizou-se o algoritmo sem execuções do SP, os valores de I_R variaram entre 15, 20 e 25, enquanto os valores de I_{ILS} variaram entre $max(2 \times n, 100)$, $max(2 \times n, 150)$ e $max(2 \times n, 200)$. Foram propostos valores adaptativos ao tamanho da instância para o parâmetro I_{ILS}, porém, com um

limite inferior. Uma vez que podem ser atribuídos valores muito baixos para esse parâmetro em instâncias de pequeno porte. Utilizando o mesmo subconjunto de instâncias dos experimentos anteriores, foram realizadas 10 execuções para cada combinação de valores dos parâmetros I_R e I_{ILS} . O gráfico dos resultados obtidos são apresentados na Figura 5.2, onde cada ponto representa uma combinação dos parâmetros. O eixo x indica o tempo médio em segundos e o eixo y representa o *gap* médio encontrado no experimento em relação aos melhores LBs do HDARP e do MDHDARP.

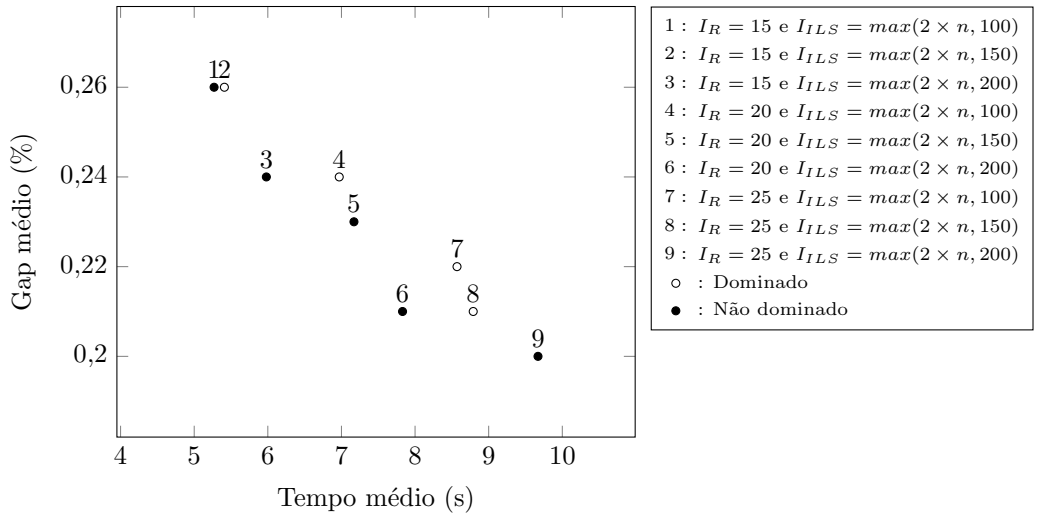

Figura 5.2: Calibração dos parâmetros I_R e I_{ILS}

Os pontos 2, 4, 7 e 8 foram todos dominados por outros pontos do gráfico. Nota-se que nenhum desses pontos representa configurações com $I_{ILS} = max(2 \times n, 200)$. Os pontos 1, 3 e 5 apresentaram os menores tempos médio dentre os pontos não dominados. Porém, os percentuais dos *gaps* médios desses pontos foram todos maiores do que $0,22\%$. Os pontos não dominados com menores *gaps* médios foram 6 e 9, sendo o ponto 6 o que demonstrou menor tempo. Por isso, a combinação representada por esse ponto, $I_R = 20$ e $I_{ILS} = max(2 \times n, 200)$, foi escolhida para o algoritmo proposto.

5.4 Análise do consumo de tempo da busca local

Assumindo os parâmetros definidos nos experimentos anteriores, foram realizados testes com o objetivo de medir o consumo de tempo de cada componente da busca local. Para isso, foram utilizadas todas as 144 instâncias do HDARP e MDHDARP. A Figura

5.3 apresenta os percentuais das médias do consumo de tempo de cada componente da busca local.

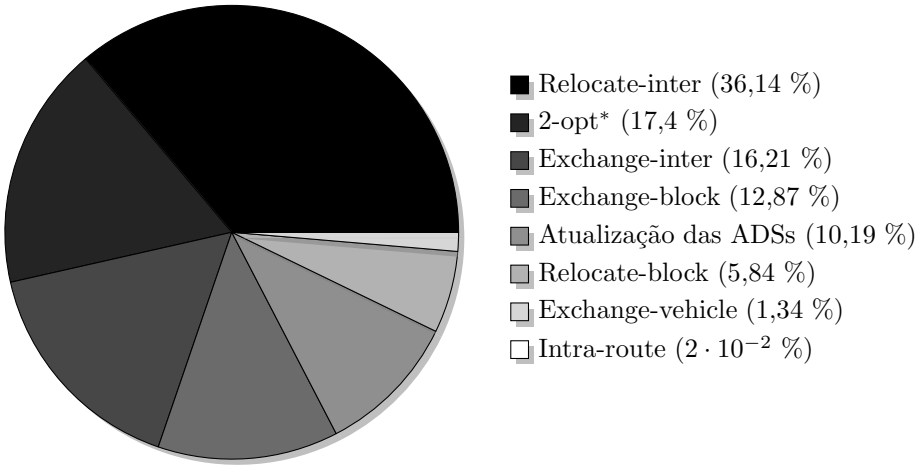

Figura 5.3: Percentual do consumo de tempo de cada componente da busca local

O gráfico indica que *Relocate-inter* e *2-opt**, em média, consumiram mais da metade do tempo da busca local. É importante destacar que o tempo necessário para atualização das ADSs é maior do que outros componentes, mais especificamente, *Relocate-block*, *Exchange-vehicle* e das vizinhanças intra-rota. Por último, *Exchange-inter* e *Exchange-block* juntos, em média, consumiram aproximadamente 30% do tempo total.

A análise aponta que apesar do *Exchange-block* ser a vizinhança com maior espaço de busca ($\mathcal{O}(n^4)$), ela não consume a maior parte do tempo na prática. Isso é explicado pela forma em que a busca local foi implementada, resultando em mais execuções da vizinhança $\mathcal{N}_{inter}$ do que da vizinhança $\mathcal{N}_{block}$.

5.5 Impacto dos mecanismos de aceleração

Após a calibração dos parâmetros, foram feitos testes para investigar o impacto dos mecanismos de aceleração no tempo computacional. A versão do algoritmo para esses experimentos utilizou os parâmetros definidos anteriormente e, como nos cenários anteriores, não foram aplicados os procedimentos de SP. Optou-se em utilizar todas as 144 instâncias do HDARP e MDHDARP (Parragh, 2011; Braekers et al., 2014), pois o objetivo desses testes foi de avaliar os ganhos de tempo em instâncias de todas as dimensões. Realizaram-se 10 execuções de cada instância para cada possibilidade

de configuração dos três mecanismos de aceleração (FSR, MMD e MSD). Ao final, agruparam-se as instâncias de mesmo tamanho, independente do grupo que pertence (U, E ou I) ou do problema associado (HDARP ou MDHDARP), e calcularam-se os tempos médios. Os resultados são apresentados no gráfico da Figura 5.4, cada linha representa uma configuração da utilização dos mecanismos de aceleração. Os eixos x e y indicam respectivamente o tamanho da instância e o tempo médio computacional que o algoritmo necessitou para resolve-la.

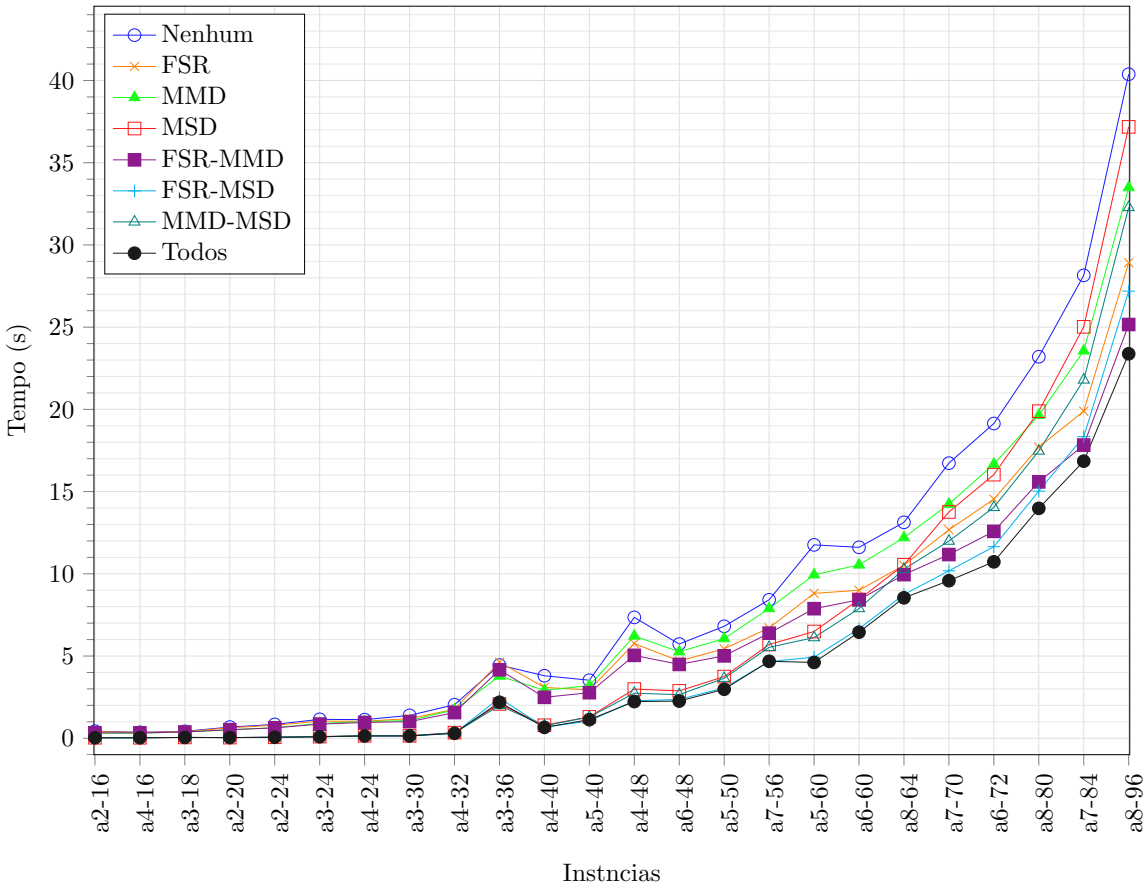

Figura 5.4: Impacto dos mecanismos de aceleração

Os resultados demostraram que os mecanismos de aceleração obtiveram ganhos de tempo durante as execuções. Observando cada mecanismo individualmente, nota-se que o MSD apresentou ganhos maiores nas instâncias de pequeno porte, mas esse ganho diminuem à medida que as instâncias crescem. O motivo para isso é que o MSD se aproveita da repetição de soluções e, instâncias de grande porte possuem uma maior variedade de soluções, por isso, seu impacto é menor em instâncias com essas caracte-

rísticas. As configurações individuais do MMD e FSR demostraram ganhos de tempo proporcionais ao passo que as instâncias aumentaram, sendo o FSR o mecanismo que mais gerou ganhos nas grandes instâncias quando comparado com os outros mecanismos. As melhorias de tempo se mantiveram quando combinaram os mecanismos, onde a configuração com todos ativados foi a que mais gerou ganhos de tempo na maioria das instâncias. No último grupo de instâncias, a versão do algoritmo sem nenhum mecanismo foi 1.73 vezes mais lenta do que a versão com todos os mecanismos.

5.6 Impacto do SP

Para medir o impacto do SP no algoritmo proposto, utilizou-se o mesmo subconjunto de instâncias dos experimentos apresentados nas Seções 5.1, 5.2 e 5.3. Três versões do algoritmo foram analisadas, uma sem SP com os parâmetros definidos nos experimentos anteriores, uma segunda versão sem o SP e com $I_R = 40$, e uma terceira com as mesmas configurações da primeira versão e com o procedimento de SP. A motivação dos testes com a segunda versão foi de que os tempos computacionais são mais similares com a versão com SP. Para cada uma das abordagens, o algoritmo foi executado 10 vezes. A Tabela 5.2 apresenta os resultados obtidos, cada coluna da tabela indica respectivamente: a instância do experimento, a quantidade de soluções iguais à melhor solução reportada na literatura (do inglês, *Best Known Solution* – BKS), a quantidade de soluções melhores do que a BKS daquela instância (do inglês, *Improvements* – *Imp*), os *gaps* médios em relação aos LBs e a média dos tempos em segundos.

A tabela mostra que os resultados com SP foram mais robustos do que as versões sem esse procedimento. Mesmo aumentando o parâmetro I_R para que sejam gastos mais tempos de execução no algoritmo sem o SP, a versão com o procedimento exato obteve consistentemente menores percentuais dos *gaps*. Além disso, a quantidade de BKSs encontradas pelo ILS-SP_{DARP} foi igual ou maior do que o ILS_{DARP} nos dois problemas. Por último, observa-se que o algoritmo com SP encontrou mais vezes soluções melhores do que as BKSs. Como resultado final, 7 novas melhores soluções foram encontradas no algoritmo com SP enquanto que os algoritmos sem SP encontraram no máximo 5.

Tabela 5.2: Impacto do procedimento SP

	ILS$_{\text{DARP}}$ ($I_R = 20$)				**ILS$_{\text{DARP}}$** ($I_R = 40$)				**ILS-SP$_{\text{DARP}}$** ($I_R = 20$)			
	#BKS	**#Imp**	**Gap (%)**	**Tempo (s)**	**#BKS**	**#Imp**	**Gap (%)**	**Tempo (s)**	**#BKS**	**#Imp**	**Gap (%)**	**Tempo (s)**
HDARP												
U	12	0	0,066	7,77	12	0	0,030	14,86	12	0	0,004	12,13
E	11	0	0,031	7,42	11	0	0,024	14,07	12	0	0,005	11,96
I	11	1	0,128	7,67	12	0	0,085	14,74	11	1	0,056	11,54
Total HDARP	34	1	0,075	7,62	35	0	0,046	14,56	35	1	0,022	11,87
MDHDARP												
U	8	0	0,295	8,34	9	0	0,254	16,28	11	1	0,167	14,14
E	9	1	0,158	8,10	10	1	0,139	15,71	11	1	0,061	14,68
I	7	3	0,602	7,70	8	4	0,554	15,36	8	4	0,455	12,07
Total MDHDARP	24	4	0,352	8,05	27	5	0,316	15,78	30	6	0,228	13,63
Total / Média	58	5	0,213	7,83	62	5	0,181	15,17	65	7	0,125	12,75

5.7 Comparação com outros algoritmos

Os resultados dos experimentos computacionais aplicados nas instâncias do HDARP e MDHDARP são detalhados nas próximas seções. As instâncias utilizadas para os testes foram propostas por Parragh e Schmid (2013) e Braekers et al. (2014) para o HDARP e MDHDARP.

O código fonte da implementação da meta-heurística DA para o HDARP e o MDHDARP (Braekers et al., 2014) foi fornecido por um dos autores. Assim, realizaram-se execuções na mesma máquina que o algoritmo proposto, isso possibilitou uma comparação mais justa entre os dois métodos. Dessa forma, para cada experimento comparou-se os resultados do algoritmo proposto no presente trabalho com os apresentados em Braekers et al. (2014). Além disso, acrescentaram-se as média de tempo em 10 execuções que o DA necessitou para resolver as instâncias na mesma máquina em que os experimentos realizados para o ILS-SP$_{\text{DARP}}$.

Por último, novos LBs obtidos após execução do algoritmo B&C (Braekers et al., 2014) foram disponibilizados pelo primeiro autor do referido trabalho. Notou-se que alguns resultados apresentados em Masmoudi et al. (2016) e Masmoudi et al. (2017) para o HDARP e MDHDARP constavam com valores menores do que os novos LBs. A provável causa dessa inconsistência pode estar relacionada à imprecisão numérica. Como os valores de tempo e custo do problema são baseados em distância euclidiana, existe a possibilidade desses cálculos resultarem em números com muitas casas decimais que são sujeitos à problemas de precisão. Por isso, os resultados apresentados em Mas-

moudi et al. (2016) e Masmoudi et al. (2017) foram desconsiderados das comparações deste trabalho.

5.7.1 HDARP

A Tabela 5.3 apresenta os resultados dos experimentos para o HDARP. Para cada instância, são apresentados os valores das BKS encontradas na literatura relacionada ao problema e os valores de LB do algoritmo B&C de Braekers et al. (2014). As próximas colunas fazem referência aos resultados obtidos pelo DA no trabalho Braekers et al. (2014), indicando respectivamente: a melhor solução encontrada, a média das soluções, o percentual dos *gaps* em relação aos LBs, as médias dos tempos para resolver cada problema e a média dos tempos que o algoritmo necessitou na mesma máquina dos experimentos deste trabalho. As últimas colunas apresentam os dados obtidos após 10 execuções do ILS-SP_{DARP}, cada coluna indica respectivamente: a melhor solução encontrada, a média dos resultados, o *gap* da média dos resultados em relação aos LBs e a média dos tempos para resolver cada instância.

Tabela 5.3: Resultados do HDARP

Instância	BKS	LB	DA					ILS-SP$_{DARP}$			
			Melhor	Média	Gap (%)	Tempo (s)	Tempo* (s)	Melhor	Média	Gap (%)	Tempo (s)
U											
a2-16	294,25	294,25	294,25	294,25	0,00	8,60	7,80	294,25	294,25	0,00	0,44
a2-20	344,83	344,83	344,83	344,83	0,00	20,20	19,80	344,83	344,83	0,00	0,62
a2-24	431,12	431,12	431,12	431,12	0,00	17,40	17,65	431,12	431,12	0,00	0,74
a3-18	300,48	300,48	300,48	300,48	0,00	9,40	7,30	300,48	300,48	0,00	0,39
a3-24	344,83	344,83	344,83	344,83	0,00	16,60	14,40	344,83	344,83	0,00	0,92
a3-30	494,85	494,85	494,85	494,85	0,00	18,80	18,10	494,85	494,85	0,00	1,18
a3-36	583,19	583,19	583,19	583,19	0,00	28,40	28,10	583,19	583,19	0,00	1,74
a4-16	282,68	282,68	282,68	282,68	0,00	9,80	6,90	282,68	282,68	0,00	0,40
a4-24	375,02	375,02	375,02	375,02	0,00	13,00	10,50	375,02	375,02	0,00	0,97
a4-32	485,50	485,50	485,50	485,50	0,00	25,60	21,70	485,50	485,50	0,00	1,53
a4-40	557,69	557,69	557,69	557,69	0,00	26,40	22,30	557,69	557,69	0,00	2,19
a4-48	668,82	668,82	668,82	668,82	0,00	35,40	33,50	668,82	668,82	0,00	4,63
a5-40	498,41	498,41	498,41	498,41	0,00	21,80	19,20	498,41	498,41	0,00	2,45
a5-50	686,62	686,62	686,62	686,62	0,00	30,60	29,60	686,62	686,62	0,00	5,63
a5-60	808,42	808,42	808,42	808,42	0,00	43,80	44,60	808,42	808,42	0,00	7,61
a6-48	604,12	604,12	604,12	604,12	0,00	30,60	27,00	604,12	604,12	0,00	4,07
a6-60	819,25	819,25	819,25	819,25	0,00	37,00	36,30	819,25	819,25	0,00	9,64
a6-72	916,05	916,05	916,05	916,05	0,00	53,60	54,10	916,05	916,24	0,02	17,32
a7-56	724,04	724,04	724,04	724,04	0,00	32,40	30,50	724,04	724,04	0,00	6,62
a7-70	889,12	889,12	889,12	889,30	0,02	43,60	42,90	889,12	889,12	0,00	13,51
a7-84	1033,37	1033,37	1033,37	1033,38	0,00	58,20	58,00	1033,37	1033,66	0,03	23,68
a8-64	747,46	747,46	747,46	747,46	0,00	37,80	36,20	747,46	747,46	0,00	13,40
a8-80	945,73	945,73	945,73	945,73	0,00	57,60	57,40	945,73	945,73	0,00	22,70
a8-96	1229,66	1229,66	1229,66	1232,03	0,19	66,40	65,60	1229,66	1229,66	0,00	35,67
Média	627,73	627,73	627,73	627,84	0,009	30,96	29,56	627,73	627,75	0,002	7,42
E											
a2-16	331,16	331,16	331,16	331,16	0,00	9,40	7,80	331,16	331,16	0,00	0,44
a2-20	347,03	347,03	347,03	347,03	0,00	19,60	19,20	347,03	347,03	0,00	0,63
a2-24	450,25	450,25	450,25	450,25	0,00	15,80	15,30	450,25	450,25	0,00	0,76
a3-18	300,63	300,63	300,63	300,63	0,00	9,60	7,20	300,63	300,63	0,00	0,44
a3-24	344,91	344,91	344,91	344,91	0,00	14,60	12,20	344,91	344,91	0,00	0,91
a3-30	500,58	500,58	500,58	500,58	0,00	17,00	16,00	500,58	500,58	0,00	1,15
a3-36	583,19	583,19	583,19	583,19	0,00	23,60	24,20	583,19	583,19	0,00	1,53
a4-16	285,99	285,99	285,99	285,99	0,00	8,20	5,90	285,99	285,99	0,00	0,41
a4-24	383,84	383,84	383,84	383,84	0,00	12,20	9,90	383,84	383,84	0,00	0,94
a4-32	500,24	500,24	500,24	500,24	0,00	22,80	20,70	500,24	500,24	0,00	1,47
a4-40	580,42	580,42	580,42	580,42	0,00	24,20	22,30	580,42	580,42	0,00	2,23
a4-48	670,52	670,52	670,52	670,52	0,00	33,60	32,70	670,52	670,52	0,00	4,41
a5-40	500,06	500,06	500,06	500,06	0,00	21,80	18,70	500,06	500,06	0,00	2,46
a5-50	693,77	693,77	693,77	693,77	0,00	28,00	26,30	693,77	693,77	0,00	4,52

Continua na página seguinte

Tabela 5.3: Resultados do HDARP (continuação)

Instância	BKS	LB	DA					ILS-SP$_{DARP}$			
			Melhor	Média	Gap (%)	Tempo (s)	Tempo* (s)	Melhor	Média	Gap (%)	Tempo (s)
E											
a5-60	828,90	828,90	828,90	829,28	0,05	40,40	38,90	828,90	828,90	0,00	7,03
a6-48	614,36	614,36	614,36	614,36	0,00	29,40	25,80	614,36	614,36	0,00	4,13
a6-60	847,58	847,58	847,58	848,48	0,11	41,60	38,60	847,58	847,58	0,00	11,27
a6-72	949,17	949,17	949,17	949,17	0,00	55,40	54,60	949,17	949,35	0,02	16,47
a7-56	740,63	740,63	740,63	740,63	0,00	32,80	29,70	740,63	740,63	0,00	5,99
a7-70	946,32	946,32	946,32	946,32	0,00	46,60	41,80	946,32	946,32	0,00	13,50
a7-84	1092,90	1092,90	1092,90	1092,93	0,00	57,20	56,20	1092,90	1093,07	0,02	26,23
a8-64	762,81	762,81	762,81	762,81	0,00	39,00	34,90	762,81	762,81	0,00	10,64
a8-80	982,71	982,71	982,71	982,71	0,00	57,40	57,30	982,71	982,71	0,00	17,12
a8-96	1265,36[1]	1265,36	1265,54	1266,78	0,11	63,00	63,80	1265,36	1265,74	0,03	38,03
Média	645,98	645,97	645,98	646,09	0,011	30,13	28,33	645,97	646,00	0,003	7,20
I											
a2-16	294,25	294,25	294,25	294,25	0,00	7,20	6,60	294,25	294,25	0,00	0,37
a2-20	355,74	355,74	355,74	355,74	0,00	17,40	17,50	355,74	355,74	0,00	0,47
a2-24	431,12	431,12	431,12	431,12	0,00	12,60	12,90	431,12	431,12	0,00	0,44
a3-18	302,17	302,17	302,17	302,17	0,00	8,40	6,60	302,17	302,17	0,00	0,35
a3-24	344,83	344,83	344,83	344,83	0,00	13,40	12,00	344,83	344,83	0,00	0,71
a3-30	494,85	494,85	494,85	494,85	0,00	14,80	14,00	494,85	494,85	0,00	1,03
a3-36	618,15	618,15	618,15	618,59	0,07	22,60	23,40	618,15	618,15	0,00	3,80
a4-16	299,05	299,05	299,05	299,05	0,00	7,20	5,30	299,05	299,05	0,00	0,27
a4-24	375,02	375,02	375,02	375,02	0,00	12,00	10,30	375,02	375,02	0,00	0,93
a4-32	486,93	486,93	486,93	487,50	0,12	21,00	19,60	486,93	486,93	0,00	1,42
a4-40	557,69	557,69	557,69	557,69	0,00	23,80	22,50	557,69	557,69	0,00	2,06
a4-48	670,72	670,72	670,72	670,72	0,00	30,00	29,40	670,72	670,72	0,00	4,87
a5-40	507,18	507,18	507,18	507,18	0,00	19,20	17,70	507,18	507,18	0,00	2,46
a5-50	690,99	690,99	690,99	690,99	0,00	27,80	27,10	690,99	690,99	0,00	5,37
a5-60	816,15	816,15	816,15	816,81	0,08	39,00	39,50	816,15	816,15	0,00	6,84
a6-48	604,12	604,12	604,12	604,12	0,00	29,00	26,90	604,12	604,12	0,00	3,87
a6-60	829,23	829,23	829,23	829,26	0,00	34,20	35,00	829,23	829,23	0,00	9,99
a6-72	938,46	938,46[2]	938,46	939,32	0,09	48,60	48,30	938,46	938,46	0,00	14,94
a7-56	727,20	727,20	727,2	727,20	0,00	29,20	28,60	727,20	727,20	0,00	6,94
a7-70	925,39	923,19[2]	925,39	927,60	0,48	39,20	38,60	925,38	925,64	0,27	13,53
a7-84	1037,04	1037,04[2]	1037,04	1037,88	0,08	54,00	54,70	1037,04	1037,04	0,00	23,01
a8-64	748,04	748,04	748,04	748,04	0,00	37,20	34,10	748,04	748,04	0,00	13,50
a8-80	969,87	968,84[2]	969,87	970,83	0,21	50,20	47,90	**968,84**	969,60	0,08	20,75
a8-96	1237,89	1233,86[2]	1237,89	1237,96	0,33	61,80	60,00	1237,89	1237,97	0,33	32,30
Média	635,92	635,62	635,92	636,20	0,061	27,49	26,60	635,88	635,92	0,028	7,09
Média Total	636,54	636,44	636,54	636,71	0,03	29,53	28,17	636,53	636,56	0,01	7,24

[1] *Resultado apresentado em Masmoudi et al. (2018)*

[2] *Novos LBs fornecidos pelo algoritmo de B&C proposto por Braekers et al. (2014)*

Os resultados apresentados demonstram que o ILS-SP$_{\text{DARP}}$ obteve um desempenho geral melhor do que o DA. O algoritmo proposto foi capaz de alcançar todas as BKSs disponibilizadas na literatura, além de uma nova descoberta, a instância a-80 do grupo *I*. Apesar da abordagem híbrida não obter dominância em todos os *gaps* individualmente, nota-se que os percentuais das médias dos *gaps* agregados por grupos de instância foram menores que o DA. Por consequência, a média dos *gaps* de todas as instâncias do ILS-SP$_{\text{DARP}}$ também foi menor que a heurística de Braekers et al. (2014). Por último, as médias dos tempos necessários para resolver as instâncias foram menores quando comparados aos resultados do DA, mesmo quando essa última foi executada na mesma máquina que o algoritmo proposto. É importante destacar os valores dos tempos necessários para resolver instâncias de pequeno porte, o motivo desse desempenho está relacionado ao mecanismo de aceleração MSD. Na média total, o ILS-SP$_{\text{DARP}}$ encontrou soluções iguais ou melhores às BKSs com maior frequência, além de menores valores para os *gaps* e para o tempo computacional quando comparados ao DA.

5.7.2 MDHDARP

Seguindo um padrão similar aos experimentos anteriores, a Tabela 5.4 ilustra os resultados para o MDHDARP. As primeiras colunas indicam: o identificador de cada instância, as BKS disponíveis na literatura e os valores de LBs encontrados pelo algoritmo exato de Braekers et al. (2014). Assim como na Tabela 5.3, as colunas seguintes apresentam informações do algoritmo DA reportadas em Braekers et al. (2014), sendo elas: a melhor solução encontrada, a média das soluções, o percentual dos *gaps* em relação aos LBs e a média dos tempos computacionais. Além disso, acrescentou-se uma coluna que indica a média dos tempos em 10 execuções que o DA necessitou na mesma máquina que os experimentos realizados para o presente trabalho. As últimas colunas exibem os resultados do ILS-SP$_{\text{DARP}}$ após 10 execuções em cada instância. As colunas dizem respeito à melhor solução encontrada, à média das soluções obtidas, ao Gap entre as médias e os LBs e à média dos tempos computacionais em cada instância.

Tabela 5.4: Resultados do MDHDARP

Instância	BKS	LB	DA					ILS-SP$_{DARP}$			
			Melhor	Média	Gap (%)	Tempo (s)	Tempo* (s)	Melhor	Média	Gap (%)	Tempo (s)
U											
a2-16	284,18	284,18	284,18	284,18	0,00	5,40	4,63	284,18	284,18	0,00	0,37
a2-20	343,43	343,43	343,43	343,43	0,00	16,00	14,77	343,43	343,43	0,00	0,58
a2-24	427,17	427,17	427,17	427,17	0,00	15,80	15,71	427,17	427,17	0,00	0,77
a3-18	289,67	289,67	289,67	289,67	0,00	8,00	5,95	289,67	289,67	0,00	0,39
a3-24	348,30	348,30	348,30	348,30	0,00	14,60	12,36	348,30	348,30	0,00	0,86
a3-30	469,16	469,16	469,16	469,16	0,00	14,40	13,08	469,16	469,16	0,00	0,99
a3-36	592,42	592,42	592,42	592,42	0,00	24,60	23,87	592,42	592,42	0,00	1,79
a4-16	262,44	262,44	262,44	262,44	0,00	7,20	5,17	262,44	262,44	0,00	0,28
a4-24	355,72	355,72	355,72	355,72	0,00	10,80	8,58	355,72	355,72	0,00	0,69
a4-32	461,65	461,65	461,65	461,65	0,00	17,80	15,58	461,65	461,65	0,00	1,55
a4-40	540,34	540,34	540,34	540,34	0,00	20,00	18,21	540,34	540,34	0,00	2,51
a4-48	631,75	631,75	631,75	632,31	0,09	25,80	24,96	631,75	631,75	0,00	4,90
a5-40	482,19	482,19	482,19	482,19	0,00	19,80	17,62	482,19	482,19	0,00	3,01
a5-50	664,54	664,54	664,54	665,17	0,09	28,60	26,61	664,54	664,54	0,00	6,70
a5-60	789,87	789,87	789,87	789,87	0,00	37,80	35,87	789,87	789,90	0,00	10,23
a6-48	586,08	586,08	586,08	586,08	0,00	25,20	22,31	586,08	586,08	0,00	5,12
a6-60	776,63	776,63	776,63	776,65	0,00	32,20	30,67	776,63	776,68	0,01	10,99
a6-72	883,78	883,78	883,78	883,78	0,00	49,20	48,22	883,78	883,78	0,00	18,91
a7-56	680,08	680,08	680,08	682,14	0,30	28,40	25,18	680,08	680,08	0,00	8,34
a7-70	854,22	854,22	855,76	857,67	0,40	40,40	38,46	854,22	855,04	0,10	15,45
a7-84	1007,33	1007,33	1007,33	1009,92	0,26	51,80	51,73	1007,33	1007,91	0,06	28,10
a8-64	713,11	713,11	713,11	713,11	0,00	35,60	32,94	713,11	713,11	0,00	13,14
a8-80	889,29[1]	885,45	890,69	892,79	0,83	47,60	46,09	**889,10**	889,67	0,48	26,14
a8-96	1185,45[1]	1170,91	1187,26	1189,74	1,61	61,00	60,19	1185,45	1186,87	1,36	42,99
Média	604,95	604,18	605,15	605,66	0,149	26,58	24,95	604,94	605,09	0,083	8,53
E											
a2-16	327,67	327,67	327,67	327,67	0,00	6,40	5,22	327,67	327,67	0,00	0,43
a2-20	345,59	345,59	345,59	345,59	0,00	17,80	14,01	345,59	345,59	0,00	0,55
a2-24	445,88	445,88	445,88	445,88	0,00	15,20	14,74	445,88	445,88	0,00	0,72
a3-18	289,67	289,67	289,67	289,67	0,00	7,80	5,95	289,67	289,67	0,00	0,39
a3-24	348,61	348,61	348,61	348,61	0,00	12,40	10,49	348,61	348,61	0,00	0,86
a3-30	471,43	471,43	471,43	471,43	0,00	12,80	11,80	471,43	471,43	0,00	1,05
a3-36	593,84	593,84	593,84	593,84	0,00	20,80	20,73	593,84	593,84	0,00	1,71
a4-16	262,44	262,44	262,44	262,44	0,00	6,00	4,30	262,44	262,44	0,00	0,26
a4-24	364,54	364,54	365,54	365,54	0,27	10,40	8,42	364,54	364,54	0,00	0,77
a4-32	476,59	476,59	476,59	476,59	0,00	16,20	14,29	476,59	476,59	0,00	1,48
a4-40	562,86	562,86	562,86	562,86	0,00	19,40	17,62	562,86	562,86	0,00	2,25
a4-48	633,49	633,49	633,49	633,49	0,00	25,00	23,97	633,49	633,49	0,00	4,24
a5-40	483,84	483,84	483,84	483,84	0,00	19,40	17,37	483,84	483,84	0,00	2,93
a5-50	674,19	674,19	674,19	674,71	0,08	26,80	24,93	674,19	674,19	0,00	5,98

Continua na página seguinte

Tabela 5.4: Resultados do MDHDARP (continuação)

Instância	BKS	LB	DA					ILS-SP$_{DARP}$			
			Melhor	Média	Gap (%)	Tempo (s)	Tempo* (s)	Melhor	Média	Gap (%)	Tempo (s)
E											
a5-60	813,96	813,96	813,96	814,00	0,00	34,60	33,91	813,96	813,96	0,00	8,32
a6-48	599,76	599,76	599,76	599,91	0,03	24,20	21,62	599,76	599,76	0,00	5,38
a6-60	802,49	802,49	802,49	803,59	0,14	34,40	32,33	802,49	802,49	0,00	10,44
a6-72	915,03	915,03	915,03	915,53	0,05	50,00	48,81	915,03	915,26	0,03	18,61
a7-56	703,62	703,62	703,62	703,62	0,00	28,20	25,49	703,62	703,62	0,00	8,20
a7-70	910,91	910,91	911,06	913,51	0,29	42,00	39,02	910,91	910,91	0,00	16,15
a7-84	1059,12	1059,12	1060,21	1062,97	0,36	51,80	51,27	1059,12	1061,46	0,22	44,21
a8-64	731,11	731,11	731,11	733,01	0,26	44,00	32,45	731,11	731,11	0,00	14,49
a8-80	925,72	925,72	927,89	930,95	0,56	71,20	45,19	925,72	925,98	0,03	22,66
a8-96	1222,43	1215,38	1222,43	1225,02	0,79	94,40	57,96	**1219,40**	1220,96	0,46	40,60
Média	623,53	623,24	623,72	624,34	0,118	28,80	24,25	623,41	623,59	0,031	8,86
I											
a2-16	284,18	284,18	284,18	284,18	0,00	7,60	3,96	284,18	284,18	0,00	0,30
a2-20	358,88	358,88	358,88	358,88	0,00	19,60	11,43	358,88	358,88	0,00	0,48
a2-24	439,29	439,29	439,29	439,29	0,00	13,80	11,98	439,29	439,29	0,00	0,45
a3-18	292,41	292,41	292,41	292,41	0,00	7,80	5,09	292,41	292,41	0,00	0,27
a3-24	348,54	348,54	348,54	348,54	0,00	13,20	9,73	348,54	348,54	0,00	0,73
a3-30	486,04	486,04	486,04	486,04	0,00	14,80	12,09	486,04	486,04	0,00	0,89
a3-36	626,96	626,96	626,96	627,73	0,12	23,60	20,81	626,96	626,96	0,00	10,58
a4-16	285,40	285,40	285,40	285,40	0,00	6,00	3,88	285,40	285,40	0,00	0,33
a4-24	357,51	357,51	357,51	357,51	0,00	12,00	8,42	357,51	357,51	0,00	0,70
a4-32	471,54	471,54	471,54	471,54	0,00	16,60	13,26	471,54	471,54	0,00	1,32
a4-40	542,56	542,56	542,56	542,56	0,00	21,40	17,04	542,56	542,56	0,00	2,13
a4-48	637,58	637,58	637,58	637,58	0,00	26,80	23,25	637,58	637,58	0,00	4,48
a5-40	496,36	496,36	496,36	496,36	0,00	18,80	14,73	496,36	496,36	0,00	3,13
a5-50	669,30	669,30	669,30	669,67	0,06	27,40	22,79	669,30	669,30	0,00	4,83
a5-60	800,10	800,10	800,10	802,42	0,29	37,60	32,65	800,10	800,10	0,00	8,47
a6-48	586,08	586,08	586,08	586,08	0,00	27,20	21,60	586,08	586,08	0,00	4,95
a6-60	789,40	789,40	789,40	789,80	0,05	32,60	27,37	789,40	789,41	0,00	8,38
a6-72	910,24	910,24	910,24	910,24	0,00	47,60	41,81	910,24	910,63	0,04	12,42
a7-56	688,51	688,51	688,51	688,51	0,00	27,80	22,63	688,51	688,51	0,00	8,38
a7-70	887,53	878,91[2]	887,53	889,75	1,23	39,40	32,81	**885,89**	885,89	0,79	12,90
a7-84	1014,12	1012,24[2]	1014,12	1016,67	0,44	56,40	48,64	**1012,24**	1013,40	0,11	28,55
a8-64	713,11	713,11	713,11	715,56	0,34	38,20	31,47	713,11	713,11	0,00	11,43
a8-80	925,56	904,73	925,56	926,63	2,42	48,00	40,02	**923,87**	925,10	2,25	19,59
a8-96	1196,35	1169,75	1196,79	1204,49	2,97	63,80	55,35	**1195,85**	1196,22	2,26	31,66
Média	616,90	614,57	617,00	617,83	0,330	27,00	22,20	616,74	616,87	0,228	7,39
Média Total	615,13	614,00	615,29	615,94	0,20	27,46	23,80	615,03	615,18	0,11	8,26

[1] *Resultados apresentado em Molenbruch et al. (2017a)*

[2] *Novos LBs fornecidos pelo algoritmo de B&C proposto por Braekers et al. (2014)*

Mais uma vez, os resultados obtidos pelo ILS-SP$_{\text{DARP}}$ apresentaram um desempenho geral melhor do que a meta-heurística proposta por Braekers et al. (2014). Todas as BKSs foram encontradas pelo algoritmo deste trabalho, além de seis novas descobertas, são elas: a8-80 do grupo U, a8-96 do grupo E e, a7-70, a7-84, a8-80 e a8-96 do grupo I. Assim como nos experimentos do HDARP, o algoritmo proposto não obteve dominância em todos os *gaps* individualmente, mas foi superior quando se agregaram as médias dos *gaps* por grupos de instância. Por fim, a média dos tempos do algoritmo proposto foi inferior ao DA, mesmo quando a comparação é realizada em execuções na mesma máquina. Comparando as médias totais entre ILS-SP$_{\text{DARP}}$ e o DA, os resultados obtidos demonstram que a abordagem proposta encontrou mais vezes soluções iguais ou melhores à's BKSs, além de menores percentuais de *gaps* com menores tempos computacionais.

Capítulo 6

Conclusão

Este trabalho propôs um algoritmo híbrido que combina a meta-heurística ILS com procedimentos exatos para resolver duas variantes do DARP. Apresentaram-se novos mecanismos de aceleração capazes de diminuir o tempo computacional do algoritmo conforme foram ilustrados nos testes. Outros experimentos computacionais foram realizados para verificar o desempenho de cada componente do algoritmo, incluindo o impacto das vizinhanças de bloco e o impacto do SP. Além disso, comparou-se o algoritmo proposto com outras meta-heurísticas da literatura.

Para os experimentos do HDARP, ILS-SP_{DARP} foi capaz de encontrar todas as BKSs disponibilizadas na literatura e ainda uma nova melhor solução descoberta. Além disso, o algoritmo obteve em média menores percentuais de *gaps* e menores tempos computacionais quando comparado com outros trabalhos (Braekers et al., 2014).

Nos experimentos relacionados ao MDHDARP, o algoritmo proposto também obteve um resultado geral melhor do que outras meta-heurísticas do estado da arte. O ILS-SP_{DARP} encontrou todas as BKSs do conjunto de instâncias que foi submetido e ainda seis novas melhores soluções. Além disso, o algoritmo mostrou superioridade na média das qualidades da solução e no custo de tempo computacional quando comparado com outros algoritmos (Braekers et al., 2014).

Para trabalhos futuros, pretende-se utilizar a checagem eficiente para violação da restrição de tempo máximo de viagem proposta por Gschwind e Drexl (2019). Estima-

se que o desempenho de tempo computacional seria reduzido com esses mecanismos, uma vez que as checagens tradicionais para violação de tempo máximo de viagem tem complexidade de $\mathcal{O}(n^2)$, onde n é o tamanho da rota.

Um outra possibilidade seria utilizar uma nova estrutura de *pool* como parâmetro de entrada para o SP. Essa outra estrutura guardaria blocos de soma zero em vez de rotas completas. Assim, o SP procuraria extrair a melhor combinação de blocos desse *pool* que fossem capazes de gerar soluções com custo mínimo.

Por último, outra sugestão seria adaptar o algoritmo para avaliar o seu desempenho em problemas similares ao DARP, como por exemplo o PDP. Caso obtivesse um bom desempenho nessa classe de problemas, poderia também realizar experimentos em variantes do PDP ou até mesmo em variantes que combinam coleta e entrega de pessoas com coleta e entrega de encomendas, como por exemplo o *share-a-ride problem* (Li et al., 2014).

REFERÊNCIAS BIBLIOGRÁFICAS

ATTANASIO, A.; CORDEAU, J.-F.; GHIANI, G.; LAPORTE, G. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, v. 30, n. 3, p. 377–387, 2004.

BEEK, O.; RAA, B.; DULLAERT, W.; VIGO, D. An efficient implementation of a static move descriptor-based local search heuristic. *Computers & Operations Research*, v. 94, p. 1–10, 2018.

BOUSSAÏD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. *Information sciences*, v. 237, p. 82–117, 2013.

BRAEKERS, K.; CARIS, A.; JANSSENS, G. K. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, v. 67, p. 166–186, 2014.

BULHÕES, T.; SUBRAMANIAN, A.; ERDOĞAN, G.; LAPORTE, G. The static bike relocation problem with multiple vehicles and visits. *European Journal of Operational Research*, v. 264, n. 2, p. 508–523, 2018.

CORDEAU, J.-F. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, v. 54, n. 3, p. 573–586, 2006.

CORDEAU, J.-F.; LAPORTE, G. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, v. 1, n. 2, p. 89–101, 2003a.

CORDEAU, J.-F.; LAPORTE, G. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, v. 37, n. 6, p. 579–594, 2003b.

DESROCHERS, M.; LAPORTE, G. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, v. 10, n. 1, p. 27–36, 1991.

GSCHWIND, T.; DREXL, M. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science*, v. 53, n. 2, p. 480–491, 2019.

GSCHWIND, T.; IRNICH, S. Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science*, v. 49, n. 2, p. 335–354, 2015.

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. *European journal of operational research*, v. 130, n. 3, p. 449–467, 2001.

HENAO, A. *Impacts of Ridesourcing-Lyft and Uber-on Transportation Including VMT, Mode Replacement, Parking, and Travel Behavior*. University of Colorado at Denver, 2017.

HO, S. C.; SZETO, W.; KUO, Y.-H.; LEUNG, J. M.; PETERING, M.; TOU, T. W. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, v. 111, p. 395–421, 2018.

JAIN, S.; VAN HENTENRYCK, P. Large neighborhood search for dial-a-ride problems. *International Conference on Principles and Practice of Constraint Programming*, p. 400–413. Springer, 2011.

LI, B.; KRUSHINSKY, D.; REIJERS, H. A.; VAN WOENSEL, T. The share-a-ride problem: People and parcels sharing taxis. *European Journal of Operational Research*, v. 238, n. 1, p. 31–40, 2014.

LOURENCO, H. R.; MARTIN, O. C.; STUETZLE, T. Handbook of metaheuristics, chapter iterated local search, 2002.

MADSEN, O. B.; RAVN, H. F.; RYGAARD, J. M. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of operations Research*, v. 60, n. 1, p. 193–208, 1995.

MASMOUDI, M. A.; BRAEKERS, K.; MASMOUDI, M.; DAMMAK, A. A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Computers & operations research*, v. 81, p. 1–13, 2017.

MASMOUDI, M. A.; HOSNY, M.; BRAEKERS, K.; DAMMAK, A. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, v. 96, p. 60–80, 2016.

MASMOUDI, M. A.; HOSNY, M.; DEMIR, E.; GENIKOMSAKIS, K. N.; CHEIKHROUHOU, N. The dial-a-ride problem with electric vehicles and battery swapping stations. *Transportation research part E: logistics and transportation review*, v. 118, p. 392–420, 2018.

MASMOUDI, M. A.; HOSNY, M.; DEMIR, E.; PESCH, E. Hybrid adaptive large neighborhood search algorithm for the mixed fleet heterogeneous dial-a-ride problem. *Journal of Heuristics*, v. 26, n. 1, p. 83–118, 2020.

MOLENBRUCH, Y.; BRAEKERS, K.; CARIS, A. Benefits of horizontal cooperation in dial-a-ride services. *Transportation Research Part E: Logistics and Transportation Review*, v. 107, p. 97–119, 2017a.

MOLENBRUCH, Y.; BRAEKERS, K.; CARIS, A. Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, v. 259, n. 1-2, p. 295–325, 2017b.

NEUMANN, F.; WITT, C. Combinatorial optimization and computational complexity. *Bioinspired Computation in Combinatorial Optimization*, p. 9–19. Springer, 2010.

PARRAGH, S. N. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, v. 19, n. 5, p. 912–930, 2011.

PARRAGH, S. N.; DOERNER, K. F.; HARTL, R. F. Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, v. 37, n. 6, p. 1129–1138, 2010.

PARRAGH, S. N.; SCHMID, V. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, v. 40, n. 1, p. 490–497, 2013.

PSARAFTIS, H. N. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation science*, v. 17, n. 3, p. 351–357, 1983.

ROPKE, S.; CORDEAU, J.-F.; LAPORTE, G. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks: An International Journal*, v. 49, n. 4, p. 258–272, 2007.

SAVELSBERGH, M. W. Local search in routing problems with time windows. *Annals of Operations research*, v. 4, n. 1, p. 285–305, 1985.

SUBRAMANIAN, A.; DRUMMOND, L. M. D. A.; BENTES, C.; OCHI, L. S.; FARIAS, R. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, v. 37, n. 11, p. 1899–1911, 2010.

SUBRAMANIAN, A.; UCHOA, E.; OCHI, L. S. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, v. 40, n. 10, p. 2519–2531, 2013.

TALBI, E.-G. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

TOTH, P.; VIGO, D. *The vehicle routing problem*. SIAM, 2002.

VIDAL, T.; CRAINIC, T. G.; GENDREAU, M.; PRINS, C. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research*, v. 40, n. 1, p. 475–489, 2013.

WILSON, N.; WEISSBERG, H. Advanced dial-a-ride algorithms research project final report, r 76-20, 1976.

WILSON, N. H.; SUSSMAN, J. M.; WONG, H.-K.; HIGONNET, T. *Scheduling algorithms for a dial-a-ride system.* Massachusetts Institute of Technology. Urban Systems Laboratory, 1971.

ZACHARIADIS, E. E.; KIRANOUDIS, C. T. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & operations research*, v. 37, n. 12, p. 2089–2105, 2010.