Universidade Federal da Paraíba

Centro de Informática

Programa de Pós-Graduação em Informática

Uma Abordagem Heurística para o Problema de Roteamento *DIAL-A-RIDE*

Daniel Leite Viana Costa

João Pessoa, Paraíba, Brasil

22 de Março de 2012

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Uma Abordagem Heurística para o Problema de Roteamento *DIAL-A-RIDE*

Daniel Leite Viana Costa

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Computação Distribuída

Prof. Dr. Clauirton de Albuquerque Siebra
(Orientador)

João Pessoa, Paraíba, Brasil

C837u Costa, Daniel Leite Viana.
Uma abordagem heurística para o problema de roteamento *DIAL-A-RIDE* / Daniel Leite Viana Costa.- João Pessoa, 2012.
55f. : il.
Orientador: Clauirton de Albuquerque Siebra
Dissertação (Mestrado) – UFPB/CI
1. Informática. 2. Ciência da computação. 3. *Dial-a-Ride Problem*. 4. Metaheurística. 5. *Iterated Local Search*. 6.*Variable Neighborhood Search*.

UFPB/BC CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de **DANIEL LEITE VIANA COSTA**, candidato ao Título de Mestre em Informática na Área de Sistemas de Computação, realizada em 22 de março de 2012.

Aos vinte e dois dias do mês de março do ano dois mil e doze, às dez horas, no Auditório do CCEN - da Universidade Federal da Paraíba, reuniram-se os membros da Banca Examinadora constituída para examinar o candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa "*Computação Distribuída*", o Sr. **DANIEL LEITE VIANA COSTA**. A comissão examinadora foi composta pelos professores doutores: CLAUIRTON DE ALBUQUERQUE SIEBRA (PPGI-UFPB), Orientador e Presidente da Banca Examinadora, LUCIDIO DOS ANJOS FORMIGA CABRAL (DI-UFPB) e ANAND SUBRAMANIAN (DEP-UFPB), como examinadores internos e LUIZ SATORU OCHI (UFF), como examinador externo. Dando início aos trabalhos, o professor CLAUIRTON DE ALBUQUERQUE SIEBRA, cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado "**Uma Abordagem Heurística para o Problema de Roteamento DIAL-A-RIDE**". Concluída a exposição, o candidato foi argüido pela Banca Examinadora que emitiu o seguinte parecer: "aprovado". Assim sendo, deve a Universidade Federal da Paraíba expedir o respectivo diploma de Mestre em Informática na forma da lei e, para constar, eu, professora Tatiana Aires Tavares, Coordenadora deste Programa, servindo de secretária lavrei a presente ata que vai assinada por mim mesmo e pelos membros da Banca Examinadora. João Pessoa, 22 de março de 2012.

Tatiana Aires Tavares

Tatiana Aires Tavares

Prof. Dr. Clauirton de Albuquerque Siebra
Orientador (PPGI-UFPB)

Prof. Dr. Lucidio dos Anjos Formiga Cabral
Examinador Interno (DI-UFPB)

Prof. Dr. Anand Subramanian
Examinador Interno (DEP-UFPB)

Prof. Dr Luiz Satoru Ochi
Examinador Externo (UFF)

Resumo

Problemas de congestionamentos, falta de vagas em garagens e carros subutilizados fazem parte do cenário atual das grandes cidades. Neste trabalho é criado um módulo para criação de rotas eficiente para sistemas de caronas utilizando a abordagem *Dial-a-Ride Problem*. O DARP é um problema de roteamento pertencente a classe NP-Completo. Este tem como objetivo minimizar os custos operacionais, mas mantendo uma qualidade de serviço para o usuário. É apresentado um algoritmo que utiliza as metaheurística *Iterated Local Search* juntamente com a *Variable Neighborhood Search* para solucionar o DARP. Comparados com outros trabalhos relevantes na área, os resultados encontrados foram melhores no que se refere à distância percorrida e no tempo médio de viagem dos clientes.

Palavras-chave: *Dial-a-Ride Problem*, Metaheurística, *Iterated Local Search*, *Variable Neighborhood Search*

Abstract

Problems of traffic jam, lack of vacancies in garages and cars underutilized are part of the current scenario of big cities. In this work is created a module for creating efficient routes for a system using the approach Dial-a-Ride Problem. The DARP is a vehicle routing problem that belongs to NP-complete class. It aims is to minimize operating costs while maintaining quality of service to the client. It is presented an algorithm that uses the metaheuristics Iterated Local Search with the Variable Neighborhood Search to solve the DARP. Compared to related work in the area, the results were better regarding to distance traveled and average travel time of customers.

Keywords: *Dial-a-Ride Problem*, Metaheuristic, *Iterated Local Search*, *Variable Neighborhood Search*

Agradecimentos

Agradeço a todos que contribuiram para a construção deste trabalho e que me apoiaram nessa jornada. Aos meus orientadores Dr. Clauirton de Albuquerque Siebra e Dr. Lucídio dos Anjos Formiga Cabral agradeço pela dedicação e confiança durante este percurso. Agradeço também a Gilberto Farias e Carlos Peixoto pelas dúvidas tiradas durante o caminho.
Agradeço aos amigos e em especial a Rosane Miranda, pela companhia e paciência.
À minha família meu eterno obrigado.
Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes) pela bolsa concedida durante os dois anos do mestrado.

Sumário

1 Introdução **1**

1.1 Motivação 1

1.2 Visão Geral do DARP 3

1.3 Objetivo Geral 4

1.4 Objetivos Específicos 4

1.5 Escopo da Dissertação 4

1.6 Estrutura da Dissertação 5

2 Fundamentação Teórica **6**

2.1 Problema *Dial-a-Ride* 6

2.1.1 DARP para veículos únicos 6

2.1.2 DARP com múltiplos veículos 7

2.2 Modelo Matemático 13

2.2.1 Formulação geral do Problema *Dial-a-Ride* 13

2.2.2 Modelo propospo por Mauri e Lorena 15

2.3 Metaheurísticas 17

2.3.1 *Iterated Local Search* 18

3 Material e Métodos **22**

3.1 Métodos Propostos 22

3.2 Algoritmo Desenvolvido 22

3.2.1 Pré-Processamento 22

3.2.2 Solução Inicial 23

3.2.3 Heurística de Programação 23

3.2.4 Estruturas de Vizinhança 26
3.2.5 Perturbação 33
3.3 Instâncias Utilizadas 34

4 Resultados Computacionais **36**
4.1 Tabelas de Resultados 36

5 Conclusões **43**

Capítulo 1

Introdução

1.1 Motivação

Os grandes centros urbanos enfrentam graves problemas devido a crescente frota de veículos. Relatos de engarrafamentos, ar poluído, motoristas estressados, são comuns em cidades com grande número de carros. Pessoas que moram perto de vias muito movimentadas também sofrem com o barulho provocado pelos engarrafamentos e/ou com o grande tráfego dos automóveis.

O tráfego tem implicações negativas na qualidade de vida e bem-estar da população, uma vez que gera incômodos e interfere nas atividades básicas e rotineiras [23]. A pesquisa realizada por Zerbini et al. [39], evidenciou que a maioria dos motoristas entrevistados apontou o trânsito como grande fator desencadeante de estresse no dia a dia, o que contribui para um baixo rendimento no trabalho e piora de qualidade de vida.

Segundo dados divulgados pelo Departamento Nacional de Trânsito (Denatran) [9] no Brasil há 64.817.974 veículos (dados de 2011). O estado recordista é São Paulo (20.537.980), seguido por Minas Gerais (7.005.640) e Paraná (5.160.354). A Companhia de Engenharia de Tráfego (CET) [26] do estado de São Paulo registrou, no mês de Março de 2011, 143km de engarrafamento.

Um outro problema provocado pelo excesso de automotores nas cidades é a falta de estacionamentos que, segundo Scaringella [30], vem sendo tratado como se não fosse um problema grave no trânsito urbano. Essa questão tem tido pouco espaço nos debates sobre soluções de mobilidade.

O aquecimento global soma-se a conta dos prejuízos causados ou ampliados pelo aumento da queima de combustíveis pela frota. Isto ocorre devido a liberação de gases como o Gás Carbônico (CO_2) que ajuda no aumento do efeito estufa.

Estimativa do 1º Inventário Nacional de Emissões Atmosféricas por Veículos Automotores Rodoviários, divulgado pelo Ministério do Meio Ambiente (MMA) [10], estimou que no ano de 2009, tenham sido emitidas quase 170 milhões de toneladas de CO_2. Este fato é destacado pelo relatório de Gestão dos Recursos Naturais elaborado pelo Ministério do Meio Ambiente e parceiros, o qual afirma que:"os veículos automotores produzem mais poluição atmosférica do que qualquer outra atividade humana isolada " [7].

Por fim temos o impacto econômico gerado pela queima de combustíveis que, segundo o levantamento da Fundação Getulio Vargas, mostra que somente a cidade de São Paulo perde anualmente 33,5 bilhões de reais por causa dos congestionamentos [37].

Os administradores públicos desses grandes aglomerados urbanos tem enfrentado o problema do crescimento da frota aumentando a malha rodoviária. Essa solução não se mostra eficaz, pois além de não acompanhar a quantidade de veículos que chegam às ruas, o espaço necessário para a construção de novas pistas em sua grande maioria é escasso [30].

Os dados do Censo IBGE (Instituto Brasileiro de Geografia e Estatística) 2010 [8], apontam que a população do Brasil é de 190,732 milhões, considerando o número total de veículos (64.817.974) verifica-se que há cerca de 2,94 habitantes por carro. Percebe-se portanto, um sub-aproveitado da frota brasileira, tendo em vista que cada veículo tem capacidade média para cinco pessoas.

Como alternativa para minimizar o baixo aproveitamento dos veículos e os problemas gerados pela grande frota, temos os sistemas de caronas apresentados em [2], [1], [4], [3]. Ao longo dos anos surgiram vários modelos de compartilhamento de veículos ao redor do mundo. Segundo Matthew Barth e Susan Shaheen [5], são exemplos clássicos: carros pela vizinhança, o modelo clássico de carro para estação e o compartilhamento de veículos multinodal.

Uma tendência observada com a evolução da carona é o modelo híbrido, que engloba algumas características dos formatos citados acima. Esse tem como objetivo buscar uma maior maleabilidade para atender as demandas dos usuários. Fazem parte dessa evolução os agendamentos flexíveis de caronas, o que torna mais atrativo para os usuários e proporciona

um sistema mais efetivo.

A proposta de modelo de caronas deve levar em consideração problemas culturais e a falta de incentivos para que pessoas disponibilizem uma vaga no seu carro. Estas podem temer pela sua segurança, pela subtração ou danos aos seus bens, ao cederem espaço em seu veículo. Uma possível alternativa para amenizar tal problema seria restringir o nicho de usuários aplicando um sistema inteligente.

Com a utilização eficiente da carona poderemos empregar melhor os recursos da frota urbana, minimizando as questões socioeconômicas e ambientais abordadas anteriormente [32]. Diante do exposto, o presente trabalho busca a melhoria do uso dos veículos que saem às ruas com sua capacidade sub-aproveitada. Para isso é apresentado um módulo de geração de rotas para sistema de caronas, que otimize a utilização do poder de deslocamento do automotor.

Este trabalho adota a abordagem *Dial-a-Ride Problem* (DARP) como solução para o agendamento de caronas. Esta consiste em desenvolver rotas e escalas de veículos para transportar usuários, os quais informam os pontos de embarque e desembarque, que estão entre a origem e destino do percurso de um veículo [15]. O DARP tem como principal objetivo processar e criar um conjunto de rotas buscando minimizar o custo operacional de transporte e maximizar o número de usuários atendidos, obedecendo um conjunto de restrições [21].

1.2 Visão Geral do DARP

O *Dial-a-Ride Problem* é uma variante de alguns modelos de problemas de roteamento de veículos, como o *Pick-up and Delivery Vehicle Routing Problem* (PDVRP) e o *Vehicle Routing Problema with Time Windows* (VRPTW). O que diferencia o DARP da maioria dos problemas é a perspectiva humana. Por se tratar de transporte de pessoas, busca-se o equilíbrio entre a redução da inconveniência dos passageiros e minimização dos custos operacionais [15].

O DARP é modelado como um grafo completo $G = (V, A)$, onde V é o conjunto de vértices e A é o conjunto de arcos. Para cada arco(i, j) é atribuído um custo de viagem $c_{i,j}$ e atribuído um tempo de viagem $t_{i,j}$, ambos positivos. O total de clientes é representado por n,

para cada cliente existe um pedido de embarque e outro de desembarque, representados pelo par de vértices $\{i, n+1\}$. Os clientes são servidos por m veículos, onde cada k veículo possui uma determinada capacidade Q^k. Existe um depósito de origem representado pelo vértice 0 e a garagem destino é representada por $2n + 1$. Cada rota deverá completar sua trajetória antes do tempo máximo de duração da rota. Cada vértice de embarque terá um determinado número de passageiros ($q_i > 0$) e no respectivo vértice de desembarque teremos a carga $q_{i+n} = -q_i$. Cada vértice tem uma janela de tempo atribuída, onde esta determina o horário mínimo para atendimento local. Diante do exposto o DARP é utilizado para alcançar os objetivos descritos a seguir.

1.3 Objetivo Geral

Desenvolver um módulo para criação de rotas para um sistema de caronas utilizando a abordagem do DARP, cujo problema será resolvido por um algoritmo aproximativo baseado nas metaheurísticas *Iterated Local Search* (ILS) e *Variable Neighborhood Search* (VNS).

1.4 Objetivos Específicos

- Desenvolver um módulo para criação de rotas.
- Aplicar metaheurísticas ILS/VNS para a solução do DARP.
- Testar o algoritmo desenvolvido em instâncias fornecidas pela Montreal Transit Commission (MTC).
- Comparar os resultados obtidos com os encontrados na literatura até o presente momento.

1.5 Escopo da Dissertação

O módulo desenvolvido para o sistema de caronas visa dar uma maior escalabilidade nos agendamentos/atendimentos das requisições de carona, atribuindo um caráter mais dinâmico ao processo de construção de rotas. Deste modo, este trabalho não foca em desenvolver

soluções híbridas, que combinem o uso de metaheurística com a exploração de vizinhanças maiores via busca local exata.

1.6 Estrutura da Dissertação

O restante desta dissertação está estruturada da seguinte maneira; Capítulo 2, motivação, objetivos e fundamentação teórica, revisão da literatura, modelagem matemática do DARP modelo proposto por Mauri e Lorena [22] e metaheurísticas aplicadas.

Para o Capítulo 3 as seções de tecnologias utilizadas, algoritmo, estruturas de vizinhança e métodos de perturbação desenvolvidos, instâncias utilizadas para a avaliação da eficiência do algoritmo em questão.

No Capítulo 4 reporta-se os resultados computacionais e a comparação com os demais trabalhos correlacionados.

Capítulo 5 a conclusão desta dissertação e propostas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Problema *Dial-a-Ride*

Ao logo do tempo foram criadas algumas abordagens para o problema *Dial-a-Ride*. Entre essas estão problemas considerando requisições dinâmicas ou estáticas, veículos únicos ou múltiplos, frota homogênea ou heterogênea, garagens únicas ou várias. Esta seção apresenta um breve histórico das principais abordagens feitas até hoje.

2.1.1 DARP para veículos únicos

Em 1980 Psaraftis [28] formulou e solucionou o DARP utilizando Programação Dinâmica, onde sua função objetivo é a minimização da soma ponderada do tempo de conclusão do percurso e a insatisfação dos clientes, considerando tanto a abordagem dinâmica quanto a estática do problema. Essa insatisfação é expressa como uma ponderação do tempo de espera antes de embarcar no veículo e no tempo de viagem e não considera janela de tempo especificada pelo usuário. Já em 1983, Psaraftis [27] atualizou o algoritmo para que o mesmo atendesse janelas de tempo especificadas pelos clientes no embarque e no desembarque. Porém o algoritmo ficou limitado a solucionar otimamente apenas pequenas instâncias.

Sexton em 1979 (*apud* Cordeau e Laporte [14]) e mais tarde Sexton juntamente com Bondin em 1985 [31] abordaram o DARP com veículo único como um passo para o DARP utilizando múltiplos veículos onde primeiramente faz-se *clusters* dos usuários. Nesse algoritmo, busca-se resolver o problema de roteamento através da inserção de heurística e resol-

vendo a programação associada ao problema. Em 1986, Desrosiers (*apud* Hosny e Munford [16]) formulou o DARP para veículo único utilizando programação dinâmica. Essa formulação incluiu janelas de tempo, capacidade do veículo e restrições de precedência. Neste trabalho foi possível obter soluções para instâncias com mais de 40 usuários.

2.1.2 DARP com múltiplos veículos

Um dos primeiros autores a propor heurísticas para o modelo estático do DARP para múltiplos veículos foi Jaw et al. [18] em 1986. Nesse modelo os autores impuseram janelas de tempo nos embarques e nos desembarques, o tempo máximo de viagem foi expresso através de função linear do tempo de viagem e imposta para cada usuário. A heurística seleciona os clientes em ordem do embarque mais viável e gradualmente os insere nas rotas dos veículos buscando aumentar o mínimo a função objetivo. Foi testada para uma instância com até 2600 clientes e 20 veículos, onde estas representavam mais de um dia de operação.

Bodin e Sexton em 1986 (*apud* Cordeau e Laporte [14]) foram os primeiros a construírem *clusters* para agrupar usuários que estavam próximos em uma combinação espacial e temporal antes de aplicar para cada *cluster* um algoritmo para único veículo e fazer *swaps* entre os clusters. Ioachim et al. [17] mostrou que existe vantagem em termos de qualidade na solução utilizam técnicas de otimização na etapa de construção desses clusters.

Toth e Vigo em 1996 [35], permitiram ao usuário especificar as requisições de embarque e desembarque com janelas de tempo. Foi imposto um limite com relação ao tempo de viagem proporcional a distância direta. Seu objetivo era minimizar o custo total do serviço. A frota que serve os usuários é composta de carros especiais e miniônibus, onde esporadicamente poderia utilizar taxis, porém com penalidades aplicadas a seu uso. Uma heurística de inserção paralela é proposta onde o problema é dividido em duas categorias. A primeira, efetua movimentos troca intra-rotas, onde tenta-se melhorar uma determinada rota pela alteração na sequencia nas quais as requisições são atendidas. A segunda categoria efetua movimentos troca inter-rotas, que consiste na troca de requisições entre duas rotas. Esse método foi testado em um conjunto com cerca de 1500 solicitações com instâncias reais da cidade de Bologna. Toth e Vigo [36] aprimoraram o método, incluindo um método de busca local, derivado da Busca Tabu (BT). Testado para 276 e 312 requisições, comparado com os resultados anteriores conseguiu alcançar melhorias significativas com as modificações pro-

postas.

Znamensky e Cunha [40] adaptaram a heurística de Inserção paralela, proposta por Madsen et al. [24], e aplicaram melhorias ao método de rotas, semelhantes as de Toth e Vigo [35]. Esse trabalho foi aplicado ao Serviço de Transporte de Deficientes da cidade de São Paulo, através do sistema ATENDE, onde utiliza-se vans ou peruas para atendimento de idosos e portadores de necessidades especiais. As instâncias correspondem a um dia de requisições, totalizando 349 pedidos de clientes e 84 veículos disponíveis distribuídos em 47 garagens espalhadas pelo município.

Cordeau e Laporte [13] desenvolveram um algoritmo multi-veículos para o DARP estático aplicando uma heurística Busca Tabu. O usuário especifica uma janela para as requisições (embarque e desembarque) e a frota de veículos pertence a um único depósito. Restrições são aplicadas a capacidade do veículo, duração do percurso e tempo máximo de viagem dos usuários. A função objetivo busca construir um conjunto de rotas de menor custo capaz de acomodar todas as requisições. Os autores utilizam três métodos heurísticos, chamados de P1, P2 e P3, que foram aplicados com a metaheurística da Busca Tabu para a resolução do problema. P1 procura minimizar apenas as violações nas janelas de tempo, já P2 além de fazer a minimização do P1, visa minimizar o tempo de duração das rotas. O método P3 busca minimizar os aspectos do P1 e P2, além de ter o escopo de reduzir o tempo que o usuário permanece dentro do veículo. Apesar dos métodos P1 e P2 serem mais simples e, consequentemente mais rápidos que P3, é o terceiro método que gera melhores soluções. As instâncias utilizadas são reais e foram fornecidas pela *Montreal Transit Commission* (MTC), além de seis conjuntos de dados também reais fornecidos pela transportadora Danish.

Jorgensen et al. [19] abordaram o DARP utilizando um Algoritmo Genético (AG). O modelo utilizado foi a versão estática, com múltiplos veículos, frota heterogênea e múltiplas garagens. A abordagem para resolução do problema foi "primeiro agrupar" e "depois rotear". Um AG agrupa os clientes para os veículos, determinando quais os clientes serão alocados em quais veículos. As etapas de roteamento determinam a sequência de atendimento dos clientes; a de programação, determina os horários para cada veículo independentemente por meio de uma heurística específica.

Cordeau [12] utiliza um algoritmo *Branch-and-Cut* para solucionar o DARP. Nesta abordagem o problema é tratado na forma estática, com múltiplos veículos, frota homogênea

e com garagem única. A resolução do problema é dada por uma abordagem exata, o que garante obter a solução ótima para o problema. As instâncias utilizadas foram geradas aleatoriamente, tendo no máximo 32 requisições.

Parragh et al. [25] utiliza um algoritmo baseado na metaheurística *Variable Neigborhood Search* (VNS). Neste trabalho são utilizados três classes de estruturas de vizinhança: operação *swap* simples, outra baseada na idéia do *ejection chain* e a terceira explora a existência de arcos onde a carga do veículo é zero. Neste estudo foram reportados novos 16 melhores resultados das 20 instâncias do *benchmark* proposto por Cordeau e Laporte [13]

Mauri e Lorena [22] abordaram o problema utilizando o *Simulated Anneling* e propuseram uma versão multiobjetivo, na qual busca minimizar tanto os custos operacionais, quanto à insatisfação dos clientes. Esse modelo diferencia-se do proposto por Cordeau [12] na consideração dos requisitos essenciais. Nessa abordagem há uma relaxação de algumas restrições, passando a compor a função objetivo, tornando mais suave a obtenção de uma solução, porém permite encontrar soluções inválidas. Essa abordagem do problema é para o DARP na versão estática, múltiplos veículos, frota heterogênea e garagens múltiplas, onde cada veículo inicia e termina em garagens específicas. Foram utilizadas as instâncias executadas por Cordeau [13] que são utilizadas em vários trabalhos relevantes para a resolução do DARP. Estas 20 instâncias são combinadas entre 24 a 144 requisições de transporte, com 48 a 288 pontos e 3 a 13 veículos. Esse trabalho conseguiu reduzir o tempo de processamento para obter as soluções em 95,45% comparado com os resultados encontrados por Cordeau [13], e 95,37% comparados com os resultados obtidos por Jorgensen et al. [19].

Kaiser [20] utilizou a abordagem criada por Mauri e Lorena [22], aplicando além do *Simulated Anneling* uma metaheurística híbrida, o *Cluster Search* proposta por Chaves [11]. Esse trabalho também foi direcionado ao DARP na forma estática, múltiplos veículos, frota heterogênea e garagens múltiplas, onde cada veículo inicia e termina em garagens específicas. As instâncias utilizadas foram as utilizadas por Cordeau [13]. Kaiser [20] quando comparado a Mauri e Lorena [22] conseguiu reduzir em 16,65% em relação ao tempo de processamento para obter as soluções, 96,21% se comparado ao trabalho de Cordeau [13] e 96,14% com o trabalho de Jorgensen et al. [19].

Os trabalhos citados acima apresentam diferentes métodos e abordagens para resolução do DARP. Algumas dessas variações dizem respeito a frota ter veículo único ou múltiplos

e ser homogênea ou heterogênea, possuir garagem única ou garagens múltiplas, além das diferentes formulações da função objetivo a ser otimizada.

Métodos exatos só foram capazes de resolver problemas com um cenário reduzido, o que distoa da demanda real, que facilmente pode chegar a centenas de clientes [21]. Essa limitação não é observada nos métodos heurísticos, capazes de obter soluções de boa qualidade com mais rapidez com problemas maiores.

Várias instâncias utilizadas nos trabalhos citados acima são obtidas através de empresas, as quais muitas vezes não disponibilizam para os demais pesquisadores. A abordagem utilizada é baseada nos trabalhos mais recentes. Os resultados dos trabalhos de Cordeau [12], Jorgensen et al. [19], Mauri e Lorena [22], Kaiser [20] e Parragh [25] serão utilizados para comparar a eficiência do algoritmo desenvolvido nesta pesquisa (ver apêndice A).

Foram utilizadas afim de comparar eficiência do algoritmo desenvolvido com os trabalhos Cordeau [12], Jorgensen et al. [19], Mauri e Lorena [22], e Kaiser [20]. A seguir temos a Tabela 2.1 com os trabalhos presentes na literaruta, suas propiedades e o tamanho dos problemas resolvidos, onde n é a quantidade de requisições.

Autores	Tipo	Objetivo	Janelas de Tempo	Restrições Utilizadas	Algoritmo	Tamanho dos problemas resolvidos
Psaraftis - 1980	Veículo Único, estático e dinamico	Minimizar a combinação de serviço	Não possui	Capacidade dos veículos, número máximo de troca de posições	Algoritmo Exato utilizando programação dinâmica	$n \leq 9$
Psaraftis - 1983	Veículo Único, estático	Minimizar a duração das rotas	Nos pontos de embarque e desembarque	Capacidade dos veículos, numero máximo de troca de posições	Algoritmo Exato utilizando programação dinâmica	$n \leq 9$
Sexton e Bodin - 1985	Veículo Único, estático	Minimizar a soma das diferenças entre o tempo de atual e o tempo desejado, e diferenças entre o tempo de viagem atual e o desejado	Limites superiores nos tempos de embarque e desembarque	Capacidade dos veículos	Heurística. Itera entre as fases de roteamento e planejamento.	$7 \leq n \leq 20$
Desrosiers et al. - 1986	Multi-veículos, estático	Minimizar a duração das rotas	Nos pontos de embarque e desembarque	Capacidade dos veículos	Algoritmo Exato utilizando programação dinâmica	$n \leq 40$
Jaw et al. - 1986	Multi-veículos, estático	Minimizar um combinação não-linear de vários tipos de desutilidades para avaliar a qualidade das soluções	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo atual de viagem não poder ser excedido de uma dada porcentagem	Heurística de inserção	$n = 250$ e $n = 2617$
Bodin and Sexton - 1986	Multi-veículos, estático	Minimizar a soma das diferenças entre o tempo de atual e o tempo desejado, e diferenças entre o tempo de viagem atual e o desejado	Limites superiores nos tempos de embarque e desembarque	Capacidade dos veículos	Heurística itera entre as fases de roteamento e planejamento.	$n \approx 85$
Toth e Vigo - 1996	Multi-veículos, estático	Minimizar o custo total de serviço	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem	Heurística de inserção paralela seguida por movimentos intra e inter rotas	$276 \leq n \leq 312$
Toth e Vigo - 1997	Multi-veículos, estático	Minimizar o custo total de serviço	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem	Heurística de inserção paralela seguida por movimentos intra e inter rotas e utilizou um método derivado da Busca Tabu	$276 \leq n \leq 312$
Znamensky e Cunha - 1999	Multi-veículos, estático	Minimizar o custo total de serviço	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem	Heurística de inserção paralela seguida por movimentos intra e inter rotas	$n \leq 349$

Cordeau e Laport - 2002	Multi-veículos, estático	Minimizar o tamanho das rotas	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem, tempo máximo do usuário dentro do veículo	Heurística Busca Tabu com reinserção de vértices	$24 \leq n \leq 295$
Jorgensen et al - 2006	Multi-veículos, estático	Minimizar o custo total de serviço	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem, tempo máximo do usuário dentro do veículo	Heurística Algoritmos Genéticos e heurística de sequência de roteamento	$24 \leq n \leq 144$
Cordeau - 2006	Multi-veículos, estático	Minimizar o custo total de serviço	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem, tempo máximo do usuário dentro do veículo	Algoritmo *Branch-and-Cut*	$16 \leq n \leq 36$
Marui e Lorena - 2009	Multi-veículos, estático	Minimizar o custo total de serviço	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem, tempo máximo do usuário dentro do veículo	*Simulated Anneling*	$24 \leq n \leq 144$
Kaiser - 2009	Multi-veículos, estático	Minimizar o custo total de serviço	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem, tempo máximo do usuário dentro do veículo	*Simulated Anneling* e *Cluster Search*	$24 \leq n \leq 144$
Parragh et al - 2009	Multi-veículos, estático	Minimizar o custo total de serviço	Nos pontos de embarque e desembarque	Capacidade dos veículos, tempo máximo de viagem, tempo máximo do usuário dentro do veículo	*Veriable Neighborhood Search*	$24 \leq n \leq 144$

Tabela 2.1: Resumo dos trabalhos relacionados ao DARP

2.2 Modelo Matemático

2.2.1 Formulação geral do Problema *Dial-a-Ride*

O modelo matemático que representa a formulação geral do DARP tem como objetivo minimizar os custos operacionais e a insatisfação dos usuários. Este consiste de *n* clientes (requisições de transporte) e *m* veículos homogêneos para os atenderem. Para cada requisição de transporte temos um ponto de embarque *i* seguido de desembarque *(i+n)*. São definidos os seguintes conjuntos para represetar o problema:

- K: conjunto dos veículos homogêneos disponíveis ($|K| = m$);
- G^-: conjunto de garagens de origem;
- G^+: conjunto de garagens de destino;
- P: conjunto dos locais de embarque;
- U: conjunto dos locais de desembarque;
- $N = \{G^- \cup P \cup U \cup G^+\}$: conjunto com todos os locais (vértices do grafo).

Para cada cliente $i \in P$ existe uma carga q_i (quantidade de assentos disponíveis), onde terá um valor positivo quando o cliente embarcar e negativo no desembarcar. Para cada ponto de embarque ou desembarque temos uma janela de tempo associada, representadas por $[e_i, l_i]$ e $[e_{i+n}, l_{i+n}]$, respectivamente.

Para cada cliente $i \in P$ existe um tempo máximo de viagem R_i, ou seja, o tempo máximo que o cliente poderá ficar dentro do veículo. Para todo local $i \in \{P \cup U\}$ é associado um tempo máximo de espera W_i, sendo o tempo máximo em que os veículos poderão ficar esperando até o início do serviço. O serviço é a ação do embarque ou desembarque do cliente em um determinado local, sendo atribuído para essa ação caso necessário, um tempo adicional s_i. Após as atribuições dos pontos que representam as garagens de embarque e desembarque e os pontos de requisições de transporte, obtém-se as distâncias $d_{i,j}$, os tempos de duração das viagens $t_{i,j}$ e os custos de deslocamento $c_{i,j}$ entre os pontos i e j, onde $i, j \in N$ e $i \neq j$.

Para cada veículo $k \in K$ existe uma capacidade Q_k associada que corresponde a quantidade de assentos disponíveis. T_k é o tempo máximo de duração da rota, ou seja, o tempo máximo de viagem que o veículo poderá fazer. A garagem específica de cada automóvel é dada por g_k^-, a garagem de origem da rota, e g_k^+, a garagem de destino da rota. A garagem de destino pode ser tanto uma garagem distinta da origem ou não. Cada garagem possui sua janela de tempo.

Têm-se ainda os chamados requisitos "essenciais", um conjunto de restrições fortes para a resolução do problema *Dial-a-Ride*. Essas restrições tem como objetivo garantir a validade das soluções. São elas:

- Tempo de duração da rota atribuída ao veículo $k \in K$ não deverá passar de T'_k (o tempo máximo permitido) ;
- Tempo de viagem do cliente $i \in P$ não deverá exceder o tempo máximo de viagem permitido R'_i;
- Tempo de espera no local $i \in \{P \cup U\}$ não deverá exceder o tempo máximo de espera permitido W'_i;
- A capacidade do veículo $Q'_k, k \in K$ não poderá ser excedido em nenhum momento do percurso;
- O serviço em cada ponto de embarque $i \in P$ deverá pertencer às janelas de tempo, $[e_i, l_i]$ e $[e_{i+n}, l_{i+n}]$, pré-estabelecidas.

Para as variáveis de decisão para o processo de roteirização e programação dos veículos, consideram-se:

- A_i^k: horário de chegada no ponto $i \in N$ pelo veículo k, onde:

$$A_i^k = \begin{cases} 0 & \text{se} \quad i \in G^- \\ D_{i-1}^k + t_{i-1,i} & \text{se} \quad i \in \{P \cup U \cup G^+\} \end{cases}$$

- D_i^k: horário de partida no ponto $i \in N$ pelo veículo k, onde:

$$D_i^k = \begin{cases} 0 & \text{se} \quad i \in G^+ \\ B_i^k + s_i & \text{se} \quad i \in \{P \cup U\} \\ B_i^k & \text{se} \quad i \in G^- \end{cases}$$

- B_i^k: horário de início de serviço no ponto $i \in N$, onde:

$$B_i^k = \begin{cases} D_i^k & \text{se} \quad i \in G^- \\ max\{e_i, A_i^k\} & \text{se} \quad i \in \{P \cup U \cup G^+\} \end{cases}$$

- W_i^k: tempo antes de começar o serviço no ponto $i \in N$ pelo veículo $k \in K$, onde:

$$W_i^k = \begin{cases} 0 & \text{se} \quad i \in G^- \\ B_i^k - A_i^k & \text{se} \quad i \in \{P \cup U \cup G^+\} \end{cases}$$

- Q_i^k: carga do veículo $k \in K$ após atendimento no ponto $i \in N$, onde:

$$Q_i^k = \begin{cases} 0 \quad \text{se} \quad i \in \{G^- \cup G^+\} \\ Q_{i-1}^k + q_i \quad \text{se} \quad i \in \{P \cup U\} \end{cases}$$

- R_i^k: tempo de viagem do cliente $i \in P$ pelo veículo k, onde: $R_i^k = B_{n+i}^k - D_i^k$

- $x_{i,j}^k$: deslocamento veículo k do ponto i para o ponto j, $\forall i, j \in N$, onde:

$$x_{i,j}^k = \begin{cases} 1 \quad \text{se } \exists\, i \to j \\ 0 \quad \text{se } \nexists\, i \to j \end{cases}$$

2.2.2 Modelo propospo por Mauri e Lorena

A resolução exata do DARP somente através de formulação matemática, usualmente tem alcance limitado não sendo viável para situações que se assemelham a casos reais [21], onde nos deparamos usualmente com elevadas dimensões.

Cordeau [12], visando simplificar o modelo matemático do DARP, reduz o número de variáveis e restrições. Essa simplificação do modelo inviabiliza a sua aplicação em muitos casos reais, uma vez que as frotas são homogênea, garagem única e com custo fixo. Esse modelo não permite priorizar a minimização dos custos operacionais ou a insatisfação dos usuários.

Minimizar:

$$\omega_0 \sum_{k \in K} \sum_{i \in N} \sum_{j \in N; j \neq i} (d_{i,j}.x_{i,j}^k) + \omega_1 \sum_{k \in K} \sum_{j \in P} x_{g_k^-,j}^k + \tag{2.1}$$

$$\omega_2 \sum_{k \in K} (B_{g_k^+} - D_{g_k^-}) + \omega_3 \sum_{i \in P} R_i + \omega_4 \sum_{i \in \{P \cup U\}} W_i + \tag{2.2}$$

$$\beta_0 \sum_{k \in K} \max\{0, (B_{g_k^+} - D_{g_k^-}) - T'_k\} + \beta_1 \sum_{i \in P} \max\{0, R_i - R'_i\} + \tag{2.3}$$

$$\beta_2 \sum_{i \in \{P \cup U\}} \max\{0, W_i - W'_i\} + \beta_3 \sum_{k \in K} \max\{0, (Q_i \sum_{i \in \{P \cup U\}} \sum_{j \in \{P \cup U\}; j \neq i; j \neq n-i} x^k_{i,j}) - \tilde{Q}\} + \tag{2.4}$$

$$\beta_4 \sum_{i \in N} (\max\{0, e_i - B_i\} + \max\{0, B_i - l_i\}) \tag{2.5}$$

Sujeito a:

$$\sum_{j \in (P \cup g_k^+)} x^k_{g_k^-, j} = 1 \qquad \forall k \in K \tag{2.6}$$

$$\sum_{i \in (U \cup g_k^-)} x^k_{i, g_k^+} = 1 \qquad \forall k \in K \tag{2.7}$$

$$\sum_{k \in K} \sum_{j \in P \cup U; j \neq i} x^k_{i,j} = 1 \qquad \forall i \in P \tag{2.8}$$

$$\sum_{j \in P \cup U; j \neq i} x^k_{i,j} - \sum_{j \in \{P \cup U \cup g_k^+\}; j \neq i; j \neq n+i} x^k_{n+i,j} = 0 \qquad \forall k \in K; i \in P \tag{2.9}$$

$$\sum_{j \in \{P \cup U \cup g_k^-\}; j \neq i; j \neq n+i} x^k_{i,j} - \sum_{j \in \{P \cup U\}; j \neq i} x^k_{i,j} = 0 \qquad \forall k \in K; i \in P \tag{2.10}$$

$$\sum_{jk \in \{P \cup U\}; j \neq i} x_{i,j} - \sum_{j \in \{P \cup U \cup g_k^+\}; j \neq i; j \neq n-i} x^k_{i,j} = 0 \qquad \forall k \in K; i \in U \tag{2.11}$$

$$B_j = (B_i + s_i + t_{i,j} + W_j) \sum_{k \in K} x_{i,j} \qquad \forall i, j \in N; i \neq j \tag{2.12}$$

$$Q_j = (Q_i + q_j) \sum_{k \in K} x^k_{i,j} \qquad \forall i, j \in N; i \neq j \tag{2.13}$$

$$A_i = B_i - W_i \qquad \forall i \in \{P \cup U \cup G^+\} \tag{2.14}$$

$$D_i = B_i + s_i \qquad \forall i \in \{P \cup U \cup G^-\} \tag{2.15}$$

$$R_i = B_{n+i} - D_i \qquad \forall i \in P \tag{2.16}$$

$$A_{g_k^-} = D^+_g = Q^-_g = Q^+_g = W^-_g = 0 \qquad \forall k \in K \tag{2.17}$$

$$A_i, W_i, B_i, D_i, Q_i \qquad \forall i \in N \tag{2.18}$$

$$R_i \geq 0 \qquad \forall i \in P \tag{2.19}$$

$$x_{i,j}^{k} \in 0,1 \qquad \forall k \in K; \forall i,j \in N; j \neq i \tag{2.20}$$

A função objetivo pode ser dividida em duas partes, a primeira (2.1 e 2.2) que visa minimizar os requisitos não-essenciais e a segunda (2.3 a 2.5) que busca diminuir as violações nos requisitos essenciais do DARP. Os termos 2.1 representam, respectivamente a distância total percorrida pelos veículos e o número de veículos utilizados para atender os clientes. Os termos 2.2 representam o tempo total de duração das rotas, de viagem dos clientes e de espera dos veículos. O termo da equação 2.3 mostra o tempo que excede o tempo máximo de duração das rotas e o tempo total que excede os tempos máximos de viagem permitidos para os clientes. Já para 2.4 temos o tempo total que excede os tempos máximos de espera permitidos para cada local nas rotas e o excesso na capacidade dos veículos. Para finalizar a função objetivo, o termo 2.5, que representa o total dos tempos que violam as janelas de tempo.

Os coeficientes da função objetivo representam as penalizações aplicadas, onde os vetores de números inteiros positivos (pesos) $\omega = [\omega_0, \omega_1, \omega_2, \omega_3, \omega_4]$ para os requisitos não-essenciais, e $\beta = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4]$ para os requisitos essenciais.

As restrições 2.6 e 2.7 garantem que cada veículo sairá da sua garagem de origem e chegará a sua garagem de destino uma única vez, 2.8 garantem que cada cliente será atendido uma única vez, por um único veículo. As restrições 2.9 garante que um local de embarque estará sempre na mesma rota que seu respectivo local de desembarque. As restrições 2.10 e 2.11 nos informam a continuação do fluxo, onde tudo que entra é igual a tudo que sai. Já a restrição 2.12 determina o horário de início do serviço, o tempo de espera em cada local e também o veículo que atenderá, as igualdades 2.13 determinam a carga dos veículos em cada local, 2.14 é responsável pelo cálculo correto dos horários de chegada, 2.15 dizem respeito à partida nos locais, 2.16 são tempos de viagem dos clientes, 2.17 inicializa algumas variáveis referentes às garagens, 2.18 e 2.19 garantem que as demais variáveis sejam irrestritas, 2.20 definem o domínio das variáveis x.

2.3 Metaheurísticas

Heurísticas são definidas como sendo técnicas inspiradas em processos intuitivos que procura uma boa solução a um custo computacional aceitável, sem que garanta a sua otimalidade e

nem quão próximo está de uma solução ótima.

Devido a necessidade de produzir soluções tão próximas quanto possível do valor da solução ótima, no menor tempo possível, é que temos heurísticas extremamente específicas. Essa alta especificidade dificulta a aplicação ou inviabiliza o seu uso para uma classe mais ampla de problemas [33].

O termo metaheurística foi introduzido por Glover em 1986. Este termo foi a junção das palavras heurística com a plavra meta, que significa nível superior, maior generalidade [11]. A metaheurística provê uma heurística mais flexível, que pode ter sua ideia facilmente modificada para um determinado problema a ser resolvido.

Os procedimentos metaheurísticos mais conhecidos incluem: Algoritmos Genéticos (AG), *Simulated Annealing* (SA), Busca Tabu (BT), GRASP, Redes Neurais, ColÃ´nia de Formigas, *Variable Neighborhood Search*, *Iterated Local Search*. Falaremos da metaheurística (*Iterated Local Search -Variable Neighborhood Search*) que será aplicada neste trabalho.

2.3.1 *Iterated Local Search*

Iterated Local Search (ILS) é uma metaheurística que foca na busca local no subespaço definido pelas soluções que apresentam ótimos locais [34]. Para a sua aplicação são necessários quatro procedimentos: Solução Inicial, responsável pela construção da solução inicial; Busca Local, refina a solução obtida inicialmente; Perturbação, gera um novo ponto de partida, afim de tentar escapar de ótimos locais de baixa qualidade; Critério de Aceitação, determina em qual solução corrente prosseguir a busca.

Juntamente com o ILS será utilizada a metaheurística *Variable Neighborhood Search* (VNS), afim de analisar um vizinho aleatório e aceitá-lo somente se estritamente melhor que a solução corrente. Caso este vizinho não obtenha melhoras, a solução anterior será mantida, e será gerado outro vizinho aleatório. Após um número máximo de iterações sem melhoras na solução corrente o método é finalizado [34].

VNS consiste em explorar o espaço de soluções gradativamente através de trocas sistemáticas de estruturas de vizinhança, concentrando a busca em torno de uma solução que obtenha movimentos de melhora. O algoritmo do VNS recebe uma solução inicial, em cada iteração seleciona-se aleatoriamente um vizinho s' aplicando uma determinada estrutura de vizinhança, aplica-se uma busca local em s' e caso esta busca retorne uma melhora recomeça

Algoritmo 1: ILS-VNS

```
1  s_0 ← Solução Inicial ;
2  s' ← VNS( s_0 , maxIterações) ;
3  iteração ← 0 ;
4  enquanto (iteração < iteração_max) faça
5      iteração++;
6      s' ← Perturbação(s', histórico);
7      s'' ← VNS(s', maxIterações);
8      se s'' < s' então
9          s' ← s'';
10         melhorIteração ← iteração;
11     fim
12 fim
```

Figura 2.1: Algoritmo ILS-VNS proposto

da estrutura de vizinhaça inicial e reinicia o contador de iterações. Caso contrário continua na próxima estrutura de vizinhança e incrementa o contador de iterações, até que a condição de parada seja atingida (Ver Figura 2.2).

O objetivo da perturbação é escapar de um ótimo local s, onde através de uma perturbação obtém-se uma solução s', a qual permitirá sair desta região. Através da busca local ao redor de s' encontra-se s'', que poderá ser uma solução melhor que a s encontrada anteriormente (Ver Figuras 2.1 e 2.3).

Neste capítulo foram expostos um modelo matemático e metaheurísticas para o DARP. A seguir serão apresentadas as tecnologias que serão aplicadas; o algoritmo que será desenvolvido no presente estudo; as instâncias a serem utilizadas; e como será a avaliação dos resultados obtidos.

Algoritmo 2: VNS

```
1  s_0 ← solução inicial;
2  s ← s_0; //Solução corrente
3  k ← número de estruturas vizinhança;
4  iterações ← 0;
5  enquanto (iterações < número máximo de iterações) faça
6      iterações++;
7      k ← 1;
8      enquanto (k < número de estruturas vizinhança) faça
9          Gera um vizinho aleatório s' ∈ N^k(s);
10         s'' ← Aplica Busca Local (s') ;
11         se (f(s'') < f(s)) então
12             s ← s'';
13             k ← 1;
14             iterações ← 0;
15         fim
16         senão
17             k ← k + 1;
18         fim
19     fim
20 fim
21 Retorne s;
22 fim VNS;
```

Figura 2.2: Algoritmo VNS

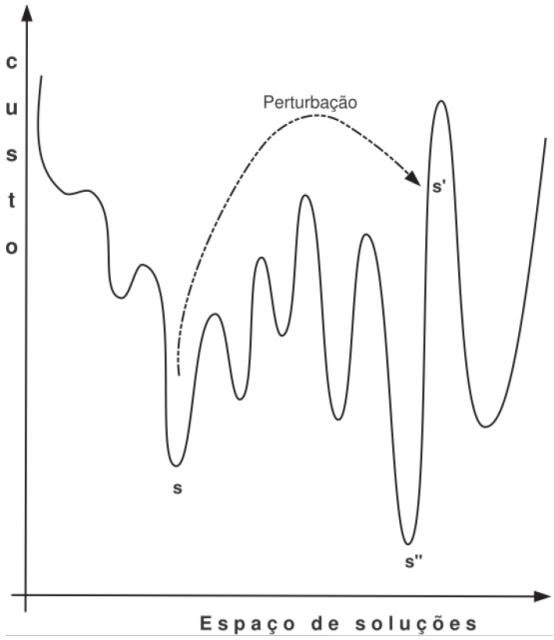

Figura 2.3: Representação esquemática do ILS (Souza, 2009).

Capítulo 3

Material e Métodos

3.1 Métodos Propostos

3.2 Algoritmo Desenvolvido

Para o presente trabalho foi aplicada a metaheurística ILS-VNS descrita anteriormente. Os procedimentos necessários para o funcionamento são descritos abaixo.

3.2.1 Pré-Processamento

O pré-processamento consiste na penalização de arcos que são inviáveis para o problema. Assim como o método apresentado por Cordeau [12], os arcos são avaliados da seguinte maneira:

- arcos $(0, n+i)$, $(i, 2n+1)$ e $(n+i, i)$ são inviáveis para $i \in P$;
- arcos (i, j) com $i, j \in N$ se $e_i + s_i + t_{i,j} > l_j$;
- arcos (i, j) e $(j, n+i)$ com $i \in P$, $j \in N$ são ambos inviáveis se $t_{i,j} + s_i + t_{j,n+i} > R$.

Para os arcos inviáveis é atribuído um valor M muito alto, tal que force não pertencer a solução.

3.2.2 Solução Inicial

A solução inicial gerada baseada no método proposto por Parragh *et al.* [25]. Este consiste em ordenar os embarques das requisições pelo início da janela de tempo. Todas as rotas são inicializadas pelas primeiras m requisições da lista. Para as demais requisições $(n - m)$ são utilizados quatro critérios de inserção, comparando com cada elemento da solução, para diminuir a distância das rotas.

O primeiro critério considera a origem da última requisição da rota e a origem da requisição a ser inserida. O segundo critério verifica a origem da última requisição da rota e o destino da requisição a ser inserida. O terceiro critério calcula o destino da última requisição da rota e a origem da requisição a ser inserida. O quarto critério determina a menor distância dentre as rotas utilizando o destino da última requisição da rota e o destino da requisição a ser inserida. Os critérios são selecionados aleatoriamente. O método é finalizado quando todas as requisições estiverem inseridas (Ver Figura 3.1).

3.2.3 Heurística de Programação

Com o objetivo de diminuir as violações das janelas de tempo, duração das rotas e os tempos de viagem dos clientes aplica-se a heurística de Programação. Esta heurística aplica o conceito de atraso proposto por Savelsbergh [29], o qual propõe atrasar o horário de partida da garagem inicial (g^-) e o início do serviço nos vértices de embarque sempre que possível. Cordeau e Laporte [13] adaptaram o conceito para o DARP,

$$F_i = min_{i \leq j \leq z} \left\{ \sum_{i < p \leq j} W_p + (min\{l_j - B_j, R - R_j\})^+ \right\},$$

onde W_p simboliza o tempo de espera no vértice p, z denota o último elemento da rota, B_j é o tempo de início de serviço no vértice j e R_j representa o tempo de origem do cliente cujo o destino é $j \in \{n+i, ..., 2n\}$ dado que $j - n$ é visitado antes de i na rota; $P_j = 0$ para todos os outros vértices.

Mauri e Lorena [22] desenvolveram o algoritmo seguinte (Figura 3.2), derivado do algoritmo proposto por Cordeau e Laporte [13]. Este método define primeiramente (linha 1) o horário de partida da garagem de origem (g^-), atribuindo o início da sua respectiva janela de tempo. Na linha 2 os cálculos de horário de chegada, início do serviço, tempo de espera,

Algoritmo 3: Solução Inicial

```
1  listaEmbarque ← Ordena a lista das requisições pela janela de tempo dos vértices de
   embarques ;
2  para cada rota k, k = 1, ..., m faça
3      Insere na rota k o primeiro elemento da listaEmbarque;
4      Remove primeiro elemento da listaEmbarque;
5  fim
6  para cada embarque na lista P, P = 1, ..., n-m faça
7      critérioInserção ← Seleciona aleatoriamente um dos quatro critérios de inserção;
8      para cada rota k,k = 1, ..., m faça
9          Calcula a distância dos elementos de acordo com o critérioInserção do
           primeiro elemento da listaEmbarque;
10     fim
11     Insere o primeiro elemento da listaEmbarque na rota que apresentou a menor
       distância;
12     Remove o primeiro elemento da listaEmbarque;
13 fim
```

Figura 3.1: Heurística para Solução Inicial

horário de partida e carga do veículo. Estes são efetuados para todos os vértices pertencetes à rota $\forall i \in K$. Na próxima linha calcula-se o atraso (este cálculo é explicitado no parágrafo seguinte) para a partida da garagem de orgiem (g^-). Na linha 4 ajusta-se o horário de partida para que não aumente as violações nas janelas de tempo. Em seguida atualiza-se as variáveis (A, B, W, D, Q) dos vértices posteriores à garagem inicial e calcula-se o tempo de viagem dos clientes. Em seguida para cada ponto de embarque $i \in P$ pertencente à rota, calcula-se seu atraso. Após calculado o atraso, ajusta-se ao horário do início de serviço de forma a reduzir a duração da rota e do tempo de viagem dos clientes e não aumentar violações nas janelas de tempo. Posteriormente atualizam-se os horários dos vértices após o vértice i e em seguida o tempo de viagem dos clientes posteriores ao i.

Algoritmo 4: Heurística de Programação

1 $B_0 \leftarrow e_0$; $D_0 \leftarrow B_0$;
2 Calcular($A_i, B_i, W_i, D_i, Q_i, \forall v_i \in V_K$ e $v_i \neq v_0$);
3 Calcular(F_0);
4 $B_0 \leftarrow e_0 + min\left\{F_0, \sum_{0<p\leq z} W_p\right\}$; $D_0 \leftarrow B_0$;
5 Atualizar($A_i, B_i, W_i, D_i, Q_i, \forall v_i \in V_K$ e $v_i \neq v_0$);
6 Calcular($R_i \forall v_i \in V_K$ e $v_i \in P$);
7 **para** *(cada vértice* $v_i \in V_k$ *e* $v_i \in P$ *)* **faça**
8 Calcular(F_i);
9 $B_i \leftarrow B_i + min\left\{F_i, \sum_{0<p\leq z} W_p\right\}$;
10 $D_i \leftarrow B_i + s_i$; $W_i \leftarrow B_i - A_i$
11 Atualizar($A_j, B_j, W_j, D_j, Q_j, \forall v_j \in V_K$ e v_j posterior a v_i);
12 Atualizar($R_j \forall v_j \in V_K$, $v_j \in P$ e $v_n + j$ posterior a v_i);
13 **fim**

Figura 3.2: Heurística de Programação

3.2.4 Estruturas de Vizinhança

Para explorar uma vizinhança $N(S)$ e buscar melhorar a solução corrente, utilizam-se dos movimentos: Reordenar Rota, Realocar Ponto, Trocar Pontos, Reverse, *ShiftRota(1)*, *ShiftRota(2)*, *ShiftRota(3)*, *Swap(1,1)*, *Swap(2,1)*, *Shift(1,0)*, *Shift(2,0)*. Os movimentos não considerarão as garagens, pois estas são fixas nas rotas.

Alguns movimentos utilizam-se da Inserção pelo Vértice Crítico apresentado por Cordeau [13]. Esta inserção consiste em inserir primeiramente o vértice crítico (VC) e em seguida inserir o vértice não crítico (VNC). Desta maneira objetiva-se inserir os vértices resultando no menor custo possível, ou diminuição da função objetivo (em caso de métodos com movimentos intra-rota).

O vértice denominado crítico possui a janela de tempo menor dentre os vértices da requisição. Já o vértice não crítico tem uma janela de tempo mais larga que o seu par. Neste método o VC é sucessivamente inserido até que encontre uma posição válida na rota, e cujo valor da função objetivo seja mínimo. Após esta inserção adiciona-se o vértice não crítico (VNC), de modo a respeitar a ordem de precedência (embarque antes do desembarque) e com o menor custo da função objetivo possível.

3.2.4.1 Reordenar Rota - $O(n)$

O movimento Reordenar Rota [22], consiste em escolher aleatoriamente uma das m rotas pertencente à solução corrente, denominada de R1. Seleciona-se ao acaso um ponto qualquer da rota e uma nova posição válida para inseri-lo. Caso não tenha posições válidas, outro vértice é selecionado. Caso a posição selecionada não respeite a restrição de precedência, onde o vértice de embarque está localizado antes do desembarque na rota, é selecionada outra posição para a inserção do vértice. Na Figura 3.3 a) temos um vértice de embarque selecionado, V1-. Este por ter seu vértice de desembarque (V1+) na terceira posição na lista, limita-se a poder trocar de posição com os elementos da rota antes do seu desembarque, ou seja, só poderá mudar de posição com o V2-. Já no exemplo mostrado na Figura 3.3 b) temos um limite também imposto pela restrição de prescedência. Neste caso quem limita é o vértice de embarque (V1-), onde o vértice V1+ só poderá mover-se para posições posteriores ao V1- na rota.

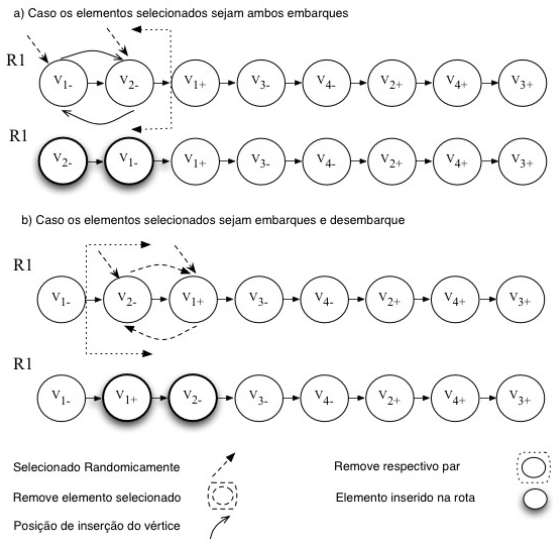

Figura 3.3: Movimento reordenar rota (Mauri, 2008).

3.2.4.2 Realocar Pontos - $O(n)$

Realocar pontos [22], compreende selecionar ao acaso duas rotas da solução corrente (R1 e R2) e escolher aleatoriamente uma requisição em R1. Remove-se os pontos de embarque, como no exemplo da Figura 3.4, onde é selecionado o vértice de embarque V2- e seu respectivo desembarque V2+, e os insere na rota R2 em uma posição aleatória, desde que não viole a restrição de precedência. Esta restrição consiste em um ponto de embarque vir primeiro que o seu respectivo par de desembarque.

3.2.4.3 Trocar Pontos - $O(n)$

A operação Trocar pontos [22], seleciona aleatoriamente duas rotas pertencentem à solução (R1 e R2) e uma requisição em cada uma das rotas. Na Figura 3.5 é selecionada os vértices V2 - na rota R1 e V8+ em R2. Remove-se os vértices escolhidos e seus respectivos complementos da requisição. Efetua-se a troca das requisições, o par de vértices que estava na rota R1 pelos vértices da rota R2. Esta troca consiste apenas em permutar os vértices.

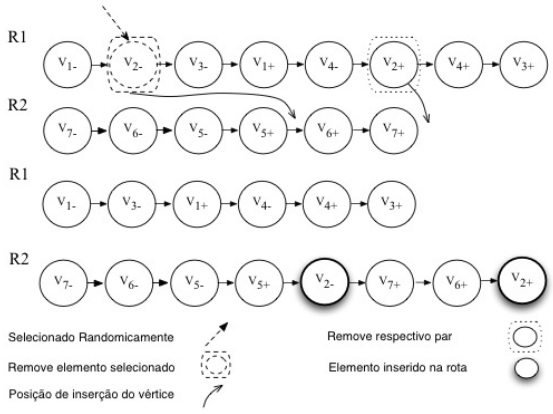

Figura 3.4: Movimento realocar pontos (Mauri, 2008).

3.2.4.4 *Reverse* - $O(n)$

Este método consiste em inverter o sentido da rota, por exemplo, o primeiro embarque a ser atendido será o ultimo vértice a ser atendido e o primeiro desembarque será o primeiro vértice a ser atendido, e em seguida trocar os embarques por desembarques e desembarques por embarques (Figura 3.6).

3.2.4.5 *ShiftRota(1)/ ShiftRota(2)/ ShiftRota(3)* - $O(n)$

Consistem em selecionar aleatoriamente uma rota R1 da solução corrente, selecionar o primeiro elemento e deslocá-lo de uma/ duas/ três posição(ões). Caso a inserção não obedeça a ordem de precedência, o desembarque é inserido na melhor posição da rota, de acordo com a função objetivo. Na Figura 3.7 tem-se o deslocamento do vértice V1-, cuja posição é a primeira da rota, em uma posição. Como neste caso não tem-se a violação da restrição de precedência, pode-se inserir na posição desejada sem modificar a posição do seu respectivo desembarque. Já para o caso mostrado na figura 3.8, onde temos a inserção do elemento V1- na segunda posição da rota, há a violação da restrição de precedência. Para este caso, removemos o seu desembarque (V1+) e inserimos o primeiro vértice na posição a qual queremos inserir (segunda posição). Após inserido o vértice de embarque, então insere-se o respectivo desembarque na melhor posição da rota, que neste caso foi a quinta posição na rota R1.

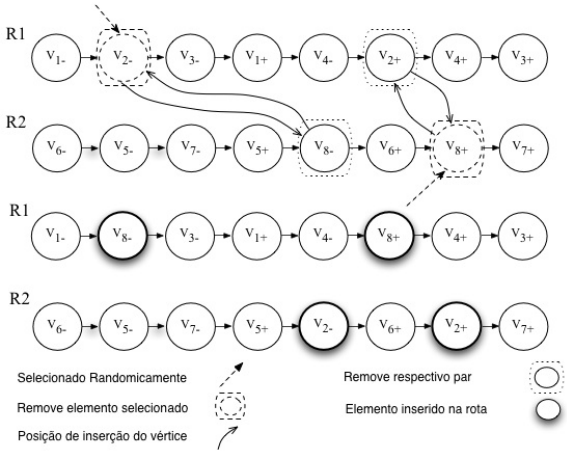

Figura 3.5: Movimento trocar pontos (Mauri, 2008).

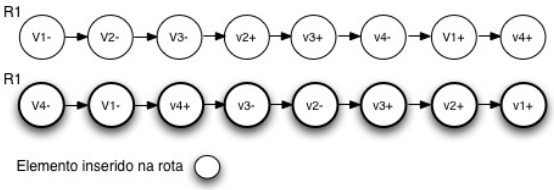

Figura 3.6: Movimento *Reverse*.

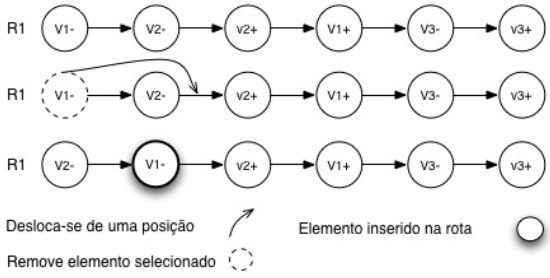

Figura 3.7: Movimento *ShiftRota* deslocando sobre um elemento.

3.2.4.6 *Swap(1,1)* - $O(n^2)$

Este movimento consiste em escolher randomicamente duas rotas da solução corrente (R1 e R2) além de aleatoriamente selecionar um vértice em cada uma dessas rotas, como mostramos na Figura 3.9 ao selecionar os vértices V3- e V6+. Em seguida removem-se as requisi-

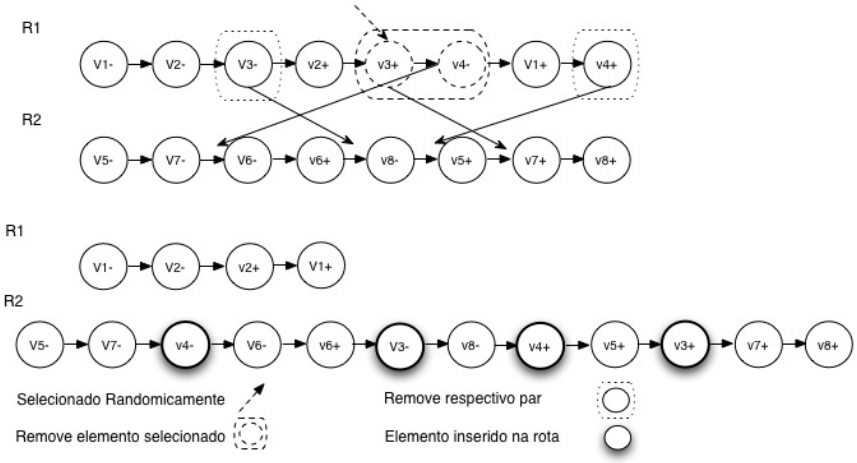

Figura 3.8: Movimento *ShiftRota* deslocando sobre dois elementos.

ções relacionadas aos vértices selecionados. O vértice que estava na rota R1 agora é inserido na rota R2 e o que estava na rota R2 é inserido à rota R1. Esta inserção é feita de acordo com o vértice crítico da requisição.

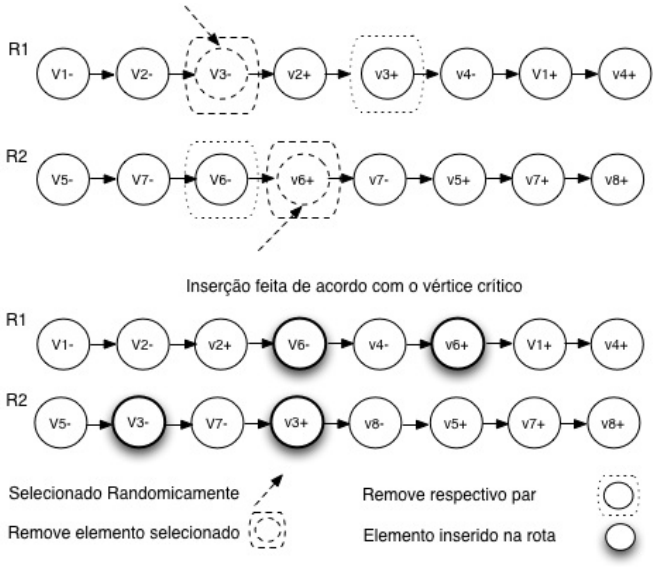

Figura 3.9: Movimento *Swap* de um elemento em cada uma das duas rotas selecionadas.

3.2.4.7 *Swap(2,1)* - $O(n^2)$

Neste procedimento escolhe-se ao acaso duas rotas R1 e R2 da solução corrente, seleciona-se randomicamente dois vértices na rota R1 e um na rota R2. No exemplo na Figura 3.10, os vértices selecionados da R1 são V3- e V2+, e da rota R2 é o V6+. Estes vértices são removidos, juntamente com o seu complemento de requisição, da sua rota original. Após a remoção ocorre a inserção das duas requisições na rota R2 e da requisição unitária na rota R1. A inserção é feita de acordo com o vértice crítico, inserindo uma requisição por vez.

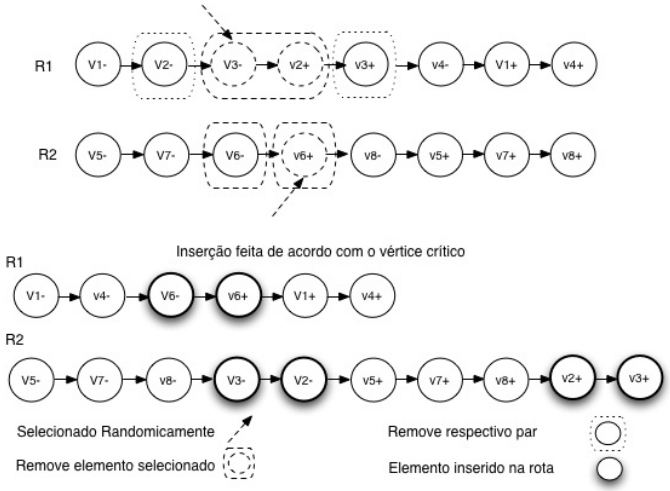

Figura 3.10: Movimento Swap com dois elementos da rota R1 e um elemento da rota R2.

3.2.4.8 *Shift(1,0)* - $O(n^2)$

Neste procedimento selecionam-se duas rotas R1 e R2 ao acaso. Na rota R1 seleciona-se aleatoriamente um vértice, remove-se o vértice escolhido e seu respectivo complemento de requisição. Em seguida insere-se esta requisição na rota R2. A inserção é feita de acordo com a inserção pelo vértice crítico (Figura 3.11).

3.2.4.9 *Shift(2,0)* - $O(n^2)$

Neste procedimento selecionam-se duas rotas R1 e R2 ao acaso. Na rota R1 selecionam-se aleatoriamente dois vértices, removem-se os vértices escolhidos e seus respectivos comple-

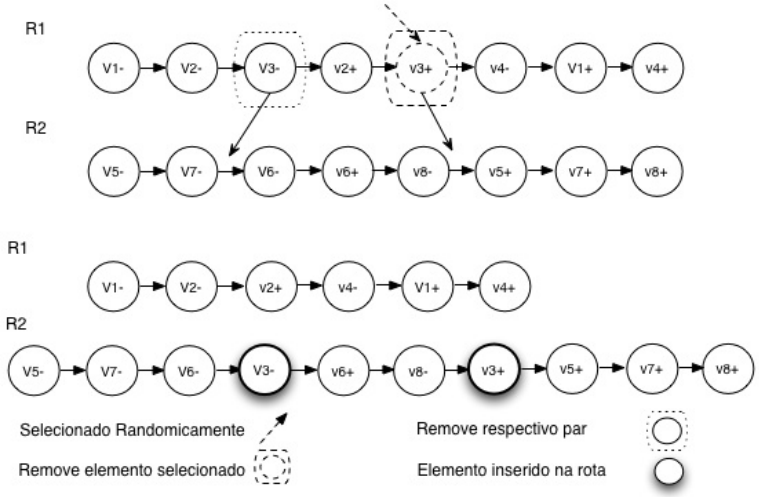

Figura 3.11: Movimento Shift(1,0) com um elemento.

mentos de requisição. Após a remoção insere-se cada uma das duas requisições, na rota R2, de acordo com a inserção pelo vértice crítico (Figura 3.12).

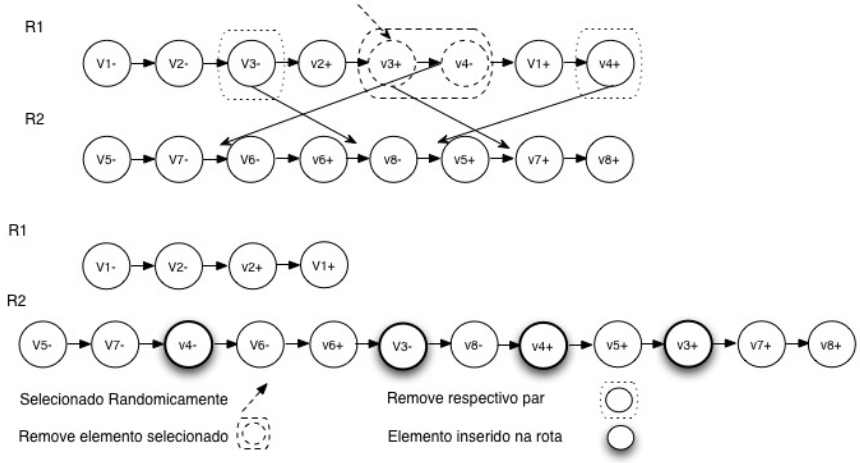

Figura 3.12: Movimento Shift(2,0) com dois elementos.

3.2.5 Perturbação

Utiliza-se o mecanismo de perturbação com o intuito de escapar de ótimos locais correntes, afim de explorar outras regiões. Essa perturbação por um lado não pode ser muito forte, pois perderá a qualidade da solução corrente, nem muito fraca, pois poderá rapidamente voltar a solução corrente. Neste trabalho utilizaremos como mecanismo de perturbação os métodos Realocar Blocos de Embarque, *Swap(m,n)* e o método Várias Vizinhanças. Estes serão explicados a seguir:

3.2.5.1 Realocar Blocos de Embarque - $O(n)$

Realocar blocos de pontos de embarque consiste em selecionar aleatoriamente uma rota e mover os vértices de embarque para uma lista, até que encontre o primeiro ponto referente a um desembarque. Esta lista é invertida e reinserida na rota em questão (Figura 3.13).

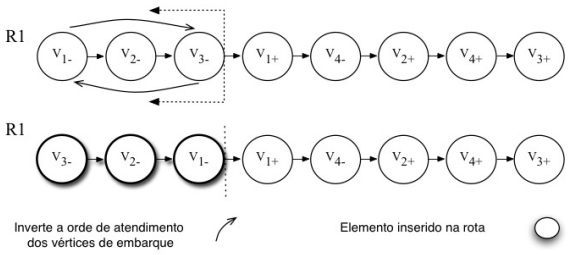

Figura 3.13: Movimento realocar blocos de embarque.

3.2.5.2 *Swap(m,n)* - $O(n^2)$

Esta estrutura de vizinhança consiste em escolher duas rotas aleatoriamente R1 e R2, além de escolher randomicamente m e n requisições (embarques e desembarque do usuário) das rotas R1 e R2 respectivamente, também escolhidas randomicamente da solução corrente. A quantidade de elementos selecionados é no máximo a quantidade de vértices menos dois vértices (embarque e respectivo desembarque), devido a restrição de não haver rotas vazias. As requisições m e n são removidas de suas rotas originais e inseridos os pares de embarque e desembarque m na rota R2 e os pares n na rota R1 (Figura 3.14).

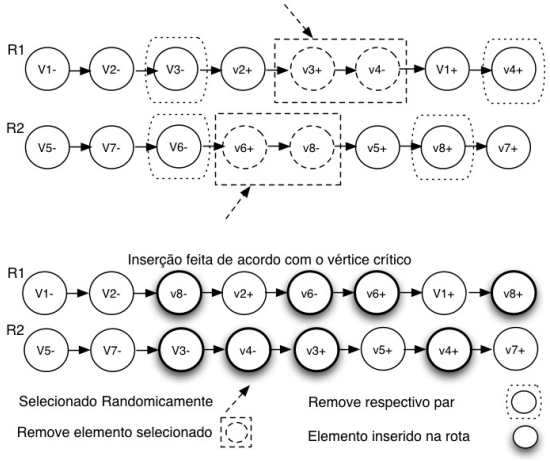

Figura 3.14: Movimento Swap dos elementos m e n.

3.2.5.3 Várias Vizinhanças - $O(n^2)$

Neste procedimento de perturbação são selecionados randomicamente uma sequência de vizinhaças apresentadas no tópico anterior. Este consiste em um laço cujo limite é um número aleatório, o que executará uma sequência randômica de estruturas de vizinhança. Esta sequência tem como número máximo a quantidade de estruturas de vizinhaças disponíveis. Neste caso haverá no máximo onze movimentos sequenciais. Para esta série de movimentos utilizaremos as seguintes estruturas de vizinhanças: Reordenar Rota, *Reverse*, *ShiftRota(1)*, *ShiftRota(2)*, *ShiftRota(3)*, *Shift(1,0)*, *Shift(2,0)*, Realocar Ponto, Trocar Pontos, *Swap(1,1)*, *Swap(2,1)*.

3.3 Instâncias Utilizadas

As instâncias utilizadas para avaliar o potencial do algoritmo desenvolvido nesse trabalho, foram as apresentadas por Cordeau e Laporte [14], disponíveis no endereço <http://www.hec.ca/chairedistributique/data/darp/>. Estas foram empregadas em vários trabalhos relevantes para a resolução do DARP [6], [12], [19], [22], [20], [25]. São disponíveis 20 instâncias, das quais foram escolhidos os 13 melhores resultados obtidos por Cordeau e Laporte [14] para serem comparados posteriormente.

Nas tabelas seguintes (3.1 e 3.2) temos as colunas " Números de veículos" que indica a quantidade de veículos disponíveis na dada instância; "Número de clientes" é a quantidade de usuários que fazem requisições de carona; "Tempo máx. da rota (min)" mostra o tempo máximo de uma rota em minutos; "Capacidade (nº de assentos)" nos diz a capacidade de carga dos veículos, ou seja, a quantidade de assentos disponíveis; e por fim, o "Tempo máximo de viagem" informa o tempo máximo que cada cliente pode ficar dentro do veículo.

Detalhamento das Instâncias Utilizadas										
Instância	R1a	R2a	R3a	R4a	R5a	R6a	R7a	R8a	R9a	R10a
Número de veículos disponíveis	3	5	7	9	11	13	4	6	8	10
Número de clientes	24	48	72	96	120	144	36	72	108	144
Tempo máx. da rota (min)	480	480	480	480	480	480	480	480	480	480
Capacidade (nº de assentos)	6	6	6	6	6	6	6	6	6	6
Tempo máximo de viagem (min)	90	90	90	90	90	90	90	90	90	90

Tabela 3.1: Informações sobre as instâncias R1a - R10a.

Detalhamento das Instâncias Utilizadas										
Instância	R1b	R2b	R3b	R4b	R5b	R6b	R7b	R8b	R9b	R10b
Número de veículos disponíveis	3	5	7	9	11	13	4	6	8	10
Número de clientes	24	48	72	96	120	144	36	72	108	144
Tempo máx. da rota (min)	480	480	480	480	480	480	480	480	480	480
Capacidade (nº de assentos)	6	6	6	6	6	6	6	6	6	6
Tempo máximo de viagem (min)	90	90	90	90	90	90	90	90	90	90

Tabela 3.2: Informações sobre as instâncias R1b - R10b.

As instâncias R1a - R10a, são formadas por janelas de tempo mais "largas", enquanto R1b - R10b são formadas por janelas mais "estreitas". São compostas de 24 a 144 requisições de transporte, com 48 a 288 pontos de embarque/desembarque e 3 a 13 veículos. Estas representam problemas com garagem única e frota homogênea, sem o conceito de tempo máximo de espera. Porém teremos problemas se considerarmos a garagem única como a garagem de origem e de destino dos veículos, e considerarmos um valor muito grande como sendo o tempo máximo de espera (o que irá eliminar violações) [22].

Capítulo 4

Resultados Computacionais

O algoritmo desenvolvido nesta dissertação foi implementado na linguagem C++. Todos os experimentos foram conduzidos em um computador equipado com o processador Intel Core 2 Duo 2.26 GHz e memória de 2 GB. Apesar do processador possuir dois cores, a aplicação utiliza apenas um. Os parâmetros utilizados foram os mesmos aplicados nos trabalhos de Jorgensen et al. (2006) [19], Mauri e Lorena (2009) [22], Kaiser(2009) [20], Parragh et al.(2010) [25].

4.1 Tabelas de Resultados

Os trabalhos comparados são: Cordeau e Laporte (2003) [13] utilizaram uma versão do Busca Tabu (BT); Jorgensen et al. (2006) [19] com um algoritmo baseado no Algoritmo Genético (AG); Mauri e Lorena (2009) [22] apresentou um *Simulated Anneling* (SA) para a resolução do DARP; Kaiser(2009) [20] desenvolveu um algoritmo utilizando as metaheurísticas *Simulated Anneling* juntamente com o *Cluster Search* (SA-CS); Parragh et al.(2010) [25] utilizou a *Variable Neighborhood Search* para solucionar o DARP; e finalmente o algoritmo desenvolvido neste trabalho o *Iterated Local Search* com o *Variable Neighborhood Search* (ILS-VNS).

Os resultados obtidos ao executar os testes estão agrupados de acordo com os aspectos distância total percorrida, duração das rotas, tempo médio de viagem, tempo médio de espera, tempo de processamento. Destes, os quatro últimos são medidos em minutos. A distância apresentada por Cordeau e Laporte [13] são a distância Euclidiana entre os pontos,

logo não apresentam unidade de medida.

Ao compararmos as distâncias obtidas (tabela 4.1) no algoritmo ILS-VSN com a Busca Tabu tem-se uma redução de 3,59% na distância percorrida. Já com os resultados do Simulated Anneling obtivemos uma melhora de 69,14%. Nos resultados do Simulated Anneling-Cluster Search obtivemos melhora de 70,11%. Em relação aos resultados encontrados no Variable Neighborhood Search tivemos melhora de 6,29%.

Distância percorrida						
Instância	BT	SA	SA- CS	VNS	ILS-VNS	*Gap*(%)
R1a	190,02	252,79	252,79	190,02	**123,00**	-35,26
R2a	302,08	437,45	431,26	302,39	**233,78**	-22,61
R3a	532,08	831,74	845,31	535,94	**489,73**	-7,96
R5a	636,97	1085,45	1.078,18	638,63	**586,03**	-8,00
R9a	672,44	1064,23	1.029,96	**670,55**	715,14	6,65
R10a	878,76	1392,09	1.440,75	875,45	**807,08**	-7,81
R1b	164,46	251,85	239,27	291,71	**114,09**	-30,63
R2b	296,06	436,69	436,29	299,40	**239,01**	-19,27
R5b	589,74	1.010,09	1.050,25	592,40	**578,55**	-1,90
R6b	**743,60**	1.289,31	1.306,93	755,47	840,21	12,99
R7b	248,21	375,67	386,38	248,21	**172,78**	-30,39
R9b	**601,96**	1.041,09	1.048,42	612,74	667,27	10,85
R10b	**798,63**	1.414,65	1.382,19	815,09	857,37	7,36
Total	6.655,01	10.883,10	10.927,98	6.828,00	**6.424,047**	

Tabela 4.1: Resultados obtidos referentes a distância percorrida.

Na Tabela 4.2 temos a duração da rota (medida em minutos). Neste quesito ver-se que o ILS-VNS obteve uma melhora de 6,95% na duração da rota se compararmos com os resultados obtidos na Busca Tabu. Ao comparar com o Algorítmo Genético obtive-se uma piora de 6,17%. Na comparação com o SA os resultados na tabela 4.2 mostra uma piora de 5,57%, o SA-CS apresenta uma piora no valor de 6,39% em relação ao ILS-VNS.

Para o quesito tempo médio de viagem computados na tabela 4.3 obteve-se uma melhora de 709,02% em relação ao Busca Tabu. Com relação ao GA tem-se melhora de 540,88%.

Duração das rotas (min)						
Instância	BT	AG	SA	SA- CS	ILS-VNS	*Gap*(%)
R1a	1.039	881,16	**831,30**	**831,30**	965,287	16,12
R2a	1.994	1.985,94	1.992,34	1.957,29	**1.412,86**	-29,09
R3a	2.781	2.579,35	**2.404,67**	2.474,62	2.433,57	1,20
R5a	4.274	3.869,95	3.920,25	3.875,25	**3.737,26**	-3,43
R9a	3.526	**3.155,49**	3.258,66	3.241,21	3.727,85	18,14
R10a	5.025	4.480,1	**4.475,42**	4.487,98	4.718,96	5,44
R1b	928	965,06	738,42	**731,95**	825,836	12,83
R2b	1.710	1.564,74	1.428,44	**1.422,84**	1.637,96	15,12
R5b	4.336	3.595,63	3.654,02	**3.479,19**	3.621,69	4,10
R6b	5.227	**4.072,47**	4.318,33	4.276,82	4.748,5	16,60
R7b	1.316	1.097,25	**1.095,67**	1.133,25	1.232,13	12,45
R9b	3.676	**3.249,29**	3.315,28	3.261,30	3.879,73	19,40
R10b	4.678	**4.040,99**	4.332,69	4.281,61	4.933,79	22,09
Total	40.508	35.537,42	35.765,49	**35.454,61**	37.875,423	

Tabela 4.2: Resultados obtidos referentes a duração das rotas em minutos.

Comparado ao SA obtemos uma melhora de 58,09%. Obteve-se uma melhora de 62,2% quando comparado aos resultados do SA-CS.

Na Tabela 4.4 temos os resultados relacionados ao tempo médio de espera. Observa-se uma melhora de 29,47% em relação à Busca Tabu. Quando comparado ao AG obtem-se uma piora de 14,84%. Na comparação com o SA também tivemos uma piora, no valor de 56,138%. Obteve-se uma piora de 59,63% ao compar com o SA-CS.

Na figura 4.1 apresenta-se graficamente o comportamento dos métodos a serem comparados para as instâncias R1a a R10a, as instâncias com a janela de tempo mais larga. É notável o crescimento do tempo em relação ao aumento da quantidade de requisições nos resultados de tempo do processamento dos algoritmos Busca Tabu, Algoritmo Genético e *Variable Neighborhood Search*. Para os algoritmos *Simulated Anneling*, *Simulated Anneling - Cluster Search* e *ILS-VNS* mostra uma variação inferior as variações dos demais algoritmos comparados.

Tempo Médio de Viagem (min)						
Instância	BT	AG	SA	SA- CS	ILS-VNS	*Gap*(%)
R1a	45,62	12,9	10,08	10,08	**5,33**	-47,12
R2a	41,18	27,72	6,46	7,14	**4,79**	-25,85
R3a	49,82	40,2	12,42	10,93	**7,26**	-33,58
R5a	51,3	40,3	7,49	7,66	**5,58**	-25,50
R9a	52,05	62,21	11,81	17,95	**6,96**	-41,07
R10a	49,75	57,92	15,31	13,53	**6,56**	-51,52
R1b	43,4	22,89	8,61	9,51	**6,22**	-27,76
R2b	49,86	27,07	6,50	6,04	**4,83**	-20,03
R5b	50,87	39,33	7,13	6,91	**5,44**	-21,27
R6b	51,02	44,42	8,65	8,3	**6,00**	-27,71
R7b	48,94	21,76	9,59	8,16	**6,23**	-23,65
R9b	51,68	49,61	10,05	9,72	**6,51**	-33,02
R10b	49,11	56,38	9,91	11,3	**6,72**	-32,19
Total	634,60	502,71	124,01	127,23	**78,44**	

Tabela 4.3: Resultados obtidos referentes ao tempo médio de viagem em minutos.

A Figura 4.2 ilustra graficamente o comportamento obtido nos trabalhos citados anteriormente. Para estes resultados foram utilizados as instâncias R1b a R10b, as quais apresentam janelas de tempo mais estreitas. Os algoritmos SA, SA-CS e ILS-VNS apresentam um comportamento menos oscilante ao longo das instância R1b a R10b, cujo janelas de tempo são mais estritas. Já os resultados obtidos nos trabalhos TB, AG e VNS mostram uma variação maior de tempo de processamento em relação à quantidade de requisições.

A Tabela 4.6 permite verificar que os resultados obtidos pelo algoritmo desenvolvido neste trabalho obtive os melhores resultados nos itens distância percorrida e tempo médio de viagem. Porém, não alcançamos os índices estabelecidos no trabalho [20] nos quesitos: duração de rotas (diferença de 6,39%), tempo médio de espera (diferença de 59,63%) e tempo de processamento (diferença de 25,76%).

Tempo Médio de Espera (min)						
Instância	BT	AG	SA	SA- CS	ILS-VNS	*Gap* (%)
R1a	4,4	5,42	**2,05**	**2,05**	9,92	383,90
R2a	7,54	5,36	6,2	5,90	**0,00**	-99,98
R3a	4,22	2,09	0,92	1,31	**0,00**	-99,89
R5a	3,47	2,2	1,81	1,65	**1,15**	-30,30
R9a	1,5	**0,15**	0,16	0,24	1,34	793,33
R10a	2,5	0,86	0,71	**0,58**	0,74	27,59
R1b	6,68	3,42	**0,14**	0,26	4,36	3.014,29
R2b	3,22	1,69	0,33	**0,28**	4,27	1.425,00
R5b	2,52	2,37	1,02	**0,12**	1,58	1.216,67
R6b	1,56	1,78	0,52	**0,31**	1,42	358,06
R7b	1,79	1,78	**0,00**	0,37	3,26	325.900,00
R9b	2,26	0,82	0,53	**0,24**	3,27	1.262,50
R10b	1,26	0,29	0,13	**0,07**	1,84	2.528,57
Total	42,92	28,23	14,52	**13,38**	33,15	

Tabela 4.4: Resultados obtidos referentes ao tempo médio de espera em minutos.

Tempo de Processamento (min)							
Instância	BT [1]	AG [2]	SA [3]	SA- CS [4]	VNS[5]	ILS-VNS[6]	*Gap*(%)
R1a	1,9	5,57	1	0,56	4,01	**0,17**	-69,64
R2a	8,06	11,43	1,2	0,79	6,72	**0,52**	-34,18
R3a	17,18	21,58	1,46	1,07	7,84	**0,82**	-23,36
R5a	50,51	40,78	2,28	**1,98**	20,14	2,42	22,22
R9a	46,24	58,23	1,79	**1,54**	31,42	2,57	66,88
R10a	87,53	65,98	2,72	**2,63**	45,52	4,77	81,37
R1b	1,93	5,46	0,92	0,56	5,10	**0,12**	-78,57
R2b	4,23	8,29	1,05	0,78	9,65	**0,38**	-51,28
R5b	8,29	11,72	1,3	**0,92**	35,98	2,12	130,43
R6b	51,28	44,66	2,26	**1,90**	42,78	4,07	114,21
R7b	54,33	58,93	1,95	1,74	5,52	**0,18**	-89,66
R9b	73,7	81,23	1,94	**1,81**	31,60	2,72	50,28
R10b	92,41	66,41	2,77	**2,59**	75,32	4,58	76,83
Total	497,59	480,27	22,64	**18,87**	321,60	25,42	

[1]2003 - Cordeau e Laporte [13], Intel Pentiun 4 2GHz

[2]2006 - Jorgensen et al. [19], Intel Celeron 2GHz

[3]2009 - Mauri e Lorena [22], Intel Celeron 2GHz

[4]2009 - Kaiser [20], AMD Athlon 64 3500 2.2GHz

[5]2010 - Parragh et al. [25], Intel Pentium D 3.2GHz

[6]2012 - Atual trabalho, Intel Core 2 Duo 2.26GHz

Tabela 4.5: Resultados obtidos referentes ao tempo de processamento em minutos.

Resumo dos resultados obtidos.							
	BT	AG	SA	SA- CS	VNS	ILS-VNS	*Gap %*
Distância percorrida	6.655,01	—	10.883,1	10.927,98	6.828	**6.424,04**	- 3,59
Duração das rotas (min)	40.508	35.537,42	35.765,49	**35.454,61**	—	37.875,42	6,39
Tempo Médio de Viagem (min)	634,6	502,71	124,01	127,23	—	**78,44**	- 58,09
Tempo Médio de Espera (min)	42,92	28,23	14,52	**13,38**	—	33,15	59,63
Tempo de Processamento (min)	497,59	480,27	22,64	**18,87**	321,60	25,42	25,76

Tabela 4.6: Resumo dos resultados obtidos.

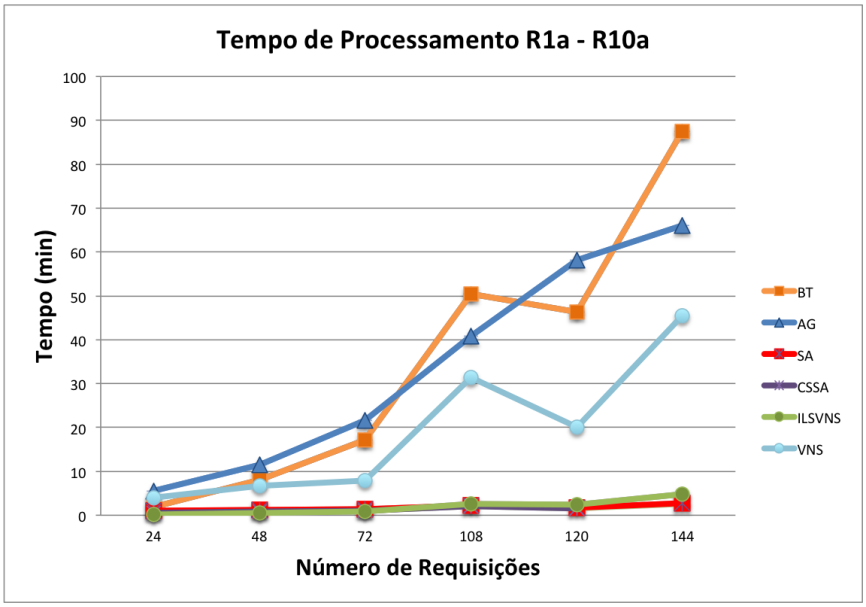

Figura 4.1: Tempo de processamento resultante da execução das instâncias R1a à R10a.

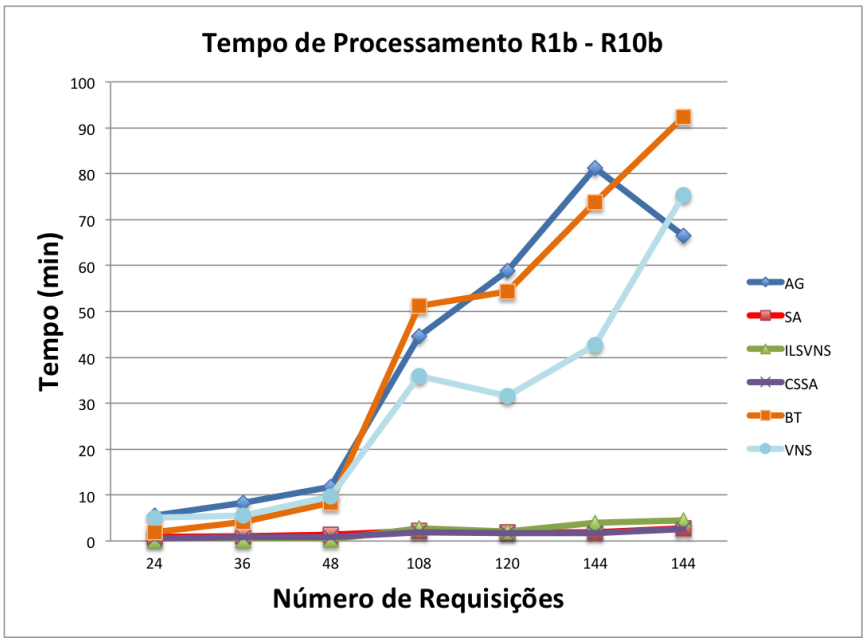

Figura 4.2: Tempo de processamento resultante da execução das instâncias R1b à R10b.

Capítulo 5

Conclusões

Esta dissertação tratou de uma nova abordagem para solução do DARP. Nesta abordagem foi utilizado a metaheurística híbrida ILS/VNS. A escolha destas metaheurísticas ocorreu devido ao fato de não encontrarmos na literatura trabalhos que utilizassem esta abordagem, assim como a facilidade de adaptação à solução do DARP.

Antes da execução da heurística de inserção, aplica-se uma penalização aos arcos do grafo que são inviáveis. Este pré-processamento consiste em atribuir aos arcos que são inviáveis um valor M muito alto, afim de inviabilizar a sua utilização durante a busca da solução.

Na construção da solução inicial utiliza-se uma heurística de inserção, a qual busca a inserção de menor custo, mas que apresenta uma pequena variação dada a aleatoriedade de onde serão inseridos os vértices da requisição. Estas poderão ser inseridas na rota de acordo com quatro posições aleatoriamente escolhidas.

A heurística de programação consiste em determinar os horários de chegada e de saída. Além de buscar reduzir as violações nas janelas de tempo, duração das rotas e tempos de viagem e de espera dos clientes, utilizando o conceito de atraso proposto por Salversbergh [29].

Na etapa de busca local foram aplicadas onze estruturas de vizinhança, onde Reordenar Rota, *Reverse*, *ShiftRota(1)*, *ShiftRota(2)*, *ShiftRota(3)*, *Shift(1,0)*, *Shift(2,0)* executam movimentos intra-rotas e Realocar Ponto, Trocar Pontos, *Swap(1,1)*, *Swap(2,1)* são movimentos inter-rotas. Para a fase de perturbação utilizamos os três seguintes mecanismos: Realocar Blocos de Embarque, *Swap(m,n)* e o método Várias Vizinhanças.

Dentre as estruturas de vizinhança supracitados podemos destacar: *Shift(1,0)*, *Shift(2,0)*, *Swap(1,1)*, *Swap(2,1)*, *Swap(m,n)*, Várias Vizinhanças como movimentos modificados neste trabalho, além do movimento Realocar Blocos de Embarque desenvolvido durante o presente estudo.

O trabalho desenvolvido foi testado pelas instância propostas por Cordeau e Laporte [14], as quais são referência no Dial-a-Ride. Obtivemos os melhores 32 de 65 resultados das instâncias. Ao compararmos os melhores resultados das somas das instâncias obtidos nos trabalhos citados anteriormente com o ILS-VNS temos: a distância percorrida tivemos uma diminuição de 3,59% ao compararmos com o melhor resultado conhecido, o Busca Tabu de Cordeau e Laporte [14]; duração das rotas tivemos uma piora de 6,39% ao compararmos com *Simulated Anneling - Cluster Search* (Kaiser [20]; com relação ao tempo médio de viagem melhoramos em 58,09% em relação aos resultados obtidos pelo *Simulated Anneling* por Mauri e Lorena [22]; para o tempo médio de espera apresentamos uma piora de 59,63% ao compararmos com o *Simulated Anneling - Cluster Search*; e para o tempo de processamento tivemos uma piora de 25,76% ao compararmos com o *Simulated Anneling - Cluster Search*.

Para trabalhos futuros sugerimos criar um sistema de caronas que permita acesso através da web, paralelizar o algoritmo, devido aos tempos computacionais que não se mostraram tão eficientes, demandando em média 2,55 minutos para uma execução; utilizar estratégias de clusterização mais eficientes; integrar a tecnologia *Vehicular Ad-Hoc Network (VANET)* [38] afim de ampliar a comunicação entre os veículos e dinamizar o processo de caronas.

Referências Bibliográficas

[1] Carona brasil, endereço eletrônico:<http://www.caronabrasil.com.br/>, acessado em: 04/01/2012.

[2] Caroneiros, endereço eletrônico:<http://www.caroneiros.com/web/>, acessado em: 04/01/2012.

[3] Caronetas, endereço eletrônico:<http://www.caronetas.com.br/>, acessado em: 04/01/2012.

[4] E-carona, endereço eletrônico:<http://www.ecarona.com.br/>, acessado em: 04/01/2012.

[5] M. Barth and S.A. Shaheen. Shared-use vehicle systems: Framework for classifying carsharing, station cars, and combined approaches. *Transportation Research Record: Journal of the Transportation Research Board*, 1791(-1):105–112, 2002.

[6] K. B Bergvinsdottir. The genetic algorithm for solving the dial-a-ride problem. Thesis, Technical University of Denmark (DTU), 2004.

[7] Brasil. Gestão dos recursos naturais - subsídios à elaboração da agenda 21 brasileira, 2000.

[8] Brasil. Censo 2010, novembro 2010.

[9] Brasil. Departamento nacional de trânsito, fevereiro 2011.

[10] Brasil. Primeiro Inventário Nacional de Emissões Atmosfíricas por Veículos Automotores Rodoviários, 2011.

[11] Antonio Augusto Chaves. *Uma Metaeurística Híbrida com Busca por Agrupamentos Aplicada a Problemas de Otimização Combinatória*. PhD thesis, Instituto Nacional de Pesquisas Espaciais - INPE.

[12] Jean-François Cordeau. A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. *Operations Research*, 54(3):573–586, May 2006.

[13] Jean-François Cordeau and Gilbert Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, July 2003.

[14] Jean-François Cordeau and Gilbert Laporte. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):89–101, June 2003.

[15] Jean-François Cordeau, Gilbert Laporte, and Stefan Ropke. Recent Models and Algorithms for One-to-One Pickup and Delivery Problems. *Distribution*, 2007.

[16] M.I. Hosny and C.L. Mumford. Single vehicle pickup and delivery with time windows: made to measure genetic encoding and operators. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 2489–2496. ACM, 2007.

[17] Irina Ioachim, Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and Daniel Villeneuve. A request clustering algorithm for door-to-door handicapped transportation. *Transportation Science*, 29(1):63–78, 1995.

[18] J.J. Jaw, A.R. Odoni, H.N. Psaraftis, and N.H.M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 20(3):243–257, 1986.

[19] R M Jorgensen, K B Bergvinsdottir, and J Larsen. Solving the Dial-a-Ride problem using genetic algorithms. *Journal of the Operational Research Society*, 58(10):1321–1331, September 2006.