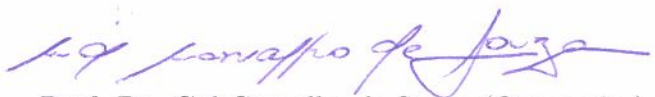
Instituto de Computação
Universidade Estadual de Campinas

Análise de algoritmos heurísticos para problemas "ricos" de roteamento de veículos

Peterson Katagiri Zilli

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Peterson Katagiri Zilli e aprovada pela Banca Examinadora.

Campinas, 13 de Maio de 2011.



Prof. Dr. Cid Carvalho de Souza (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA POR
Ana Regina Machado – CRB8/5467– BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA - UNICAMP

Zilli, Peterson Katagiri
Z65a Análise de algoritmos heurísticos para problemas "ricos" de roteamento de veículos / Peterson Katagiri Zilli.-- Campinas, SP : [s.n.], 2011.

Orientador: Cid Carvalho de Souza.
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

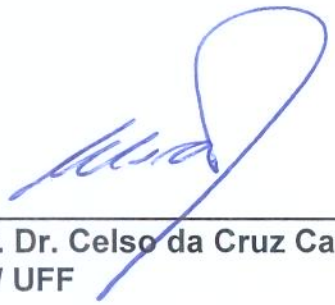
1. Otimização combinatória. 2. Problema de roteamento de veículos. 3. Algoritmos heurísticos. I. Souza, Cid Carvalho de, 1963-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: Analysis of heuristic algorithms for rich vehicle routing problems
Palavras-chave em inglês:
Combinatorial optimization
Vehicle routing problem
Heuristic algorithms
Área de concentração: Pesquisa Operacional
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Cid Carvalho de Souza [Orientador]
Celso da Cruz Carneiro Ribeiro
Flávio Keidi Miyazawa
Data da defesa: 13-05-2011
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 13 de maio de 2011, pela Banca examinadora composta pelos Professores Doutores:



Prof. Dr. Celso da Cruz Carneiro Ribeiro
ICC / UFF

Prof. Dr. . Flávio Keidi Miyazawa
IC / UNICAMP

Prof. Dr. Cid Carvalho de Souza
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Análise de algoritmos heurísticos para problemas "ricos" de roteamento de veículos

Peterson Katagiri Zilli[1]

Maio de 2011

Banca Examinadora:

- Prof. Dr. Cid Carvalho de Souza (Orientador)
- Prof. Dr. Celso Ribeiro (Membro Titular Externo)
  Instituto de Ciência da Computação, Universidade Federal Fluminense
- Prof. Dr. Arnaldo Moura (Membro Titular Interno)
  Instituto de Computação – UNICAMP
- Prof. Dr. Carlile Lavor (Membro Suplente Externo)
  Instituto de Matemática, Estatística e Computação Científica – UNICAMP
- Prof. Dr. Flávio Miyazawa (Membro Suplente Interno)
  Instituto de Computação – UNICAMP

[1]Suporte financeiro de: Bolsa FAPESP (processo 2007/06864-6) 2008–2011

Resumo

O Problema de Roteamento de Veículos (*VRP*, em inglês) foi proposto por Dantzig e Ramser em 1959 e, desde então, um grande número de artigos foi dedicado à solução de suas variantes. O problema original consiste em determinar rotas otimais que serão usadas por veículos de capacidade limitada para servirem a um conjunto de clientes. Neste trabalho focamos o estudo e a implementação dos modelos chamados de *"ricos"* na literatura, os quais englobam variantes complexas do VRP e conseguem representar situações mais próximas dos problemas logísticos encontrados em sistemas de distribuição reais. A principal motivação para esta pesquisa é uma aplicação prática referente ao problema de roteamento dos ônibus fretados pela UNICAMP para o transporte de seus funcionários, que se caracteriza como um modelo rico. O objetivo final é a otimização de tal processo através da minimização da distância total percorrida ou do número de veículos empregados, com a consequente redução dos gastos incorridos pela Universidade. Portanto, além do seu aspecto científico, esta dissertação produz resultados com chances reais de trazer benefícios à administração de uma instituição pública de ensino. Para que isto venha a ocorrer, as heurísticas desenvolvidas foram inseridas em um sistema de informações geográficas, que será usado pela universidade no processo de criação e otimização das rotas a serem licitadas publicamente.

Abstract

The Vehicle Routing Problem (*VRP*) was first proposed by Dantzig and Ramser in 1959 and, since then, a large number of papers has been devoted to the solution of its variants. The original problem consists in determining an optimal set of routes to be used by vehicles of limited capacity that serve a set of customers. In this paper we focus on the study and implementation of models called *"rich"* in the literature, which include complex variants of the VRP that represent situations closer to the logistical problems encountered in real distribution systems. The main motivation for this research is a practical problem concerning the routing of buses chartered by UNICAMP for transporting a part of its employees, which is characterized as a rich model. The goal is to optimize this process by minimizing the total travel distance or the number of vehicles used, with a consequent reduction of the expenses incurred by the University. Therefore, in addition to its scientific aspect, this dissertation gives results with real chances to benefit the administration of a public university. For this to happen, the heuristics developed were entered into a geographic information system, which will be used by the university in the process of creation and optimization of routes to be publicly auctioned.

Agradecimentos

Gostaria de agradecer

A Deus.

Ao meu orientador Prof. Cid Carvalho de Souza e a todos os meus mestres, especialmente aqueles que compuseram a banca. Por terem me ensinado muito mais do que cabe aqui, nesta dissertação. E por terem ocupado suas horas preciosas comigo dedicando-se a corrigir meus erros e a aperfeiçoar meus acertos (e, também, por *doarem* esta mesma oportunidade, sempre com o mesmo afinco, a dezenas de outras pessoas de potencial todos os anos).

Aos meus pais, Pedro e Alexsandra, minha inspiração maior. Aos meus irmãos Pedro Augusto e Marco Aurélio. A mamãe Sayuri. Aos meus avós e a toda minha família. Pelo apoio incondicional, por todo o amor e por todos os momentos especiais que passamos juntos. Como vocês fazem falta aqui pertinho...

Aos meus amigos, que foram chegando de mansinho e acabaram por transformar dias os ruins em bons, e os bons em dias melhores ainda.

Sou muito grato e devo muito a todos esses.
Dedico esta dissertação a eles.

Sumário

Resumo **vii**

Abstract **ix**

Agradecimentos **xi**

1 Introdução **1**

2 Conceitos Básicos **5**
2.1 Problema de Roteamento de Veículos 5
2.1.1 O Problema de Roteamento de Veículos com Restrições de Capacidade 6
2.1.2 Variantes *Clássicas* e Variantes *Ricas* do VRP 8
2.2 Heurísticas e Metaheurísticas 9
2.2.1 Heurísticas 9
2.2.2 Metaheurísticas 10

3 O Problema do Caixeiro Comprador Viajante **17**
3.1 Introdução 17
3.2 Revisão da Literatura 19
3.3 Novo Algoritmo Heurístico para o TPP 25
3.3.1 Uma Heurística Baseada em LNS-VNS para o TPP 25
3.3.2 Um método especializado para o UTPP simétrico 31
3.4 Estruturas de Dados 31
3.4.1 Estrutura de Dados: `rota`. 32
3.4.2 Estrutura de Dados: `aquisições`. 33
3.5 Experimentos Computacionais 35
3.5.1 Descrição das Instâncias utilizadas 35
3.5.2 Experimentos Preliminares 37
3.5.3 Análise Comparativa dos Resultados Obtidos Sobre as Instâncias da Literatura 39

3.6 Conclusões e Trabalhos Futuros 44

4 O Problema dos m Caixeiros Compradores Viajantes 47
4.1 Introdução 48
4.2 Revisão da Literatura 48
4.3 Descrição da Heurística para o CmTPP 50
4.3.1 O procedimento de construção de solução inicial 50
4.3.2 O procedimento de busca local 53
4.3.3 A Metaheurística LNS 53
4.4 Descrição de uma redução do CmRSP para o CmTPP 55
4.5 Experimentos Computacionais com instâncias do CmRSP 56
4.6 Conclusões e trabalhos futuros 59

5 O Problema "Rico" de Transporte dos Funcionários da UNICAMP 61
5.1 Introdução 61
5.2 Descrição do Problema da UNICAMP 63
5.3 Restrições do Problema: os algoritmos e o engenheiro de rotas 65
5.4 Revisão de Problemas Relacionados 66
5.5 Uma Breve Apresentação da Interface Gráfica com o Usuário do Sistema . 67
5.6 Método de Resolução e o Papel Interface Gráfica com o Usuário do Sistema 69
5.7 Instâncias Reais 71
5.7.1 Testes Preliminares com os Dados Levantados 73
5.8 Conclusões e Trabalhos Futuros 75

6 Conclusões 79

A Resultados Detalhados do Capítulo 3 81

B Descrição de alguns problemas clássicos de otimização citados no texto 99
B.1 Problema do Caixeiro Viajante 99
B.2 Problema do Empacotamento em Caixas (unidimensional) 99

Bibliografia 101

Lista de Tabelas

3.1 Comparação da qualidade da solução e tempo de execução produzido pelo esquema VND completo com abordagens que omitem um dos seus operadores e com resultados de um de seus operadores 38
3.2 Comparação da qualidade da solução e tempo de execução produzido pelo método heurístico de acordo com conjuntos distintos de parâmetros para a fase 3 39
3.3 Performance do Algoritmo H8-sutpp (deste trabalho) em comparação com a performance dos outros algoritmos da literatura. 40
3.4 Resultados para as Instâncias Simétricas não Capacitadas Para as Quais a Solução Ótima Ainda não foi Provada 41
3.5 Comparação da qualidade da solução e tempo de execução produzido pelo método heurístico H8 nas instâncias da Classe Simétrico Capacitado 43
3.6 Qualidade da solução e tempo de execução produzido pelo método heurístico H8 nas instâncias da Classe Assimétrico Não Capacitado 44
3.7 Qualidade da solução e tempo de execução produzido pelo método heurístico H8 nas instâncias da Classe Assimétrico Capacitado 45

4.1 Tabela Resultados Classe A 58
4.2 Tabela Resultados Classe B 60

A.1 Resultados detalhados da H8 para as Instâncias Simétricas sem Restrições de Capacidade 82
A.2 Resultados detalhados da H8 para as Instâncias Simétricas sem Restrições de Capacidade (continuação) 83
A.3 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 50 mercados e 50 produtos 84
A.4 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 50 mercados e 100 produtos 85
A.5 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 50 mercados e 150 produtos 86

A.6 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 50 mercados e 200 produtos 87
A.7 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 100 mercados e 50 produtos 88
A.8 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 100 mercados e 100 produtos 89
A.9 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 100 mercados e 150 produtos 90
A.10 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 100 mercados e 200 produtos 91
A.11 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 150 mercados e 50 produtos 92
A.12 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 150 mercados e 100 produtos 93
A.13 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 150 mercados e 150 produtos 94
A.14 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 150 mercados e 200 produtos 95
A.15 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 200 mercados e 50 produtos 96
A.16 Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 200 mercados e 100 produtos 97

Lista de Figuras

2.1 Uma Instância de Entrada e uma Solução do CVRP 7
2.2 Fluxogramas da Heurística Construtiva e da Heurística de Busca Local . . 11

3.1 Ilustração dos operadores `2-opt`, `or-opt`, `i-opt` e `3-opt`. 29
3.2 Estrutura de dados `rota`: detalhe da representação de um mercado na rota em memória. 32
3.3 Estrutura de dados `rota`: representação em memória. 34
3.4 Estrutura de dados `aquisições`: representação em memória. 36

4.1 Estrutura de dados `aquisições`: representação do problema de fluxo de custo mínimo. 53

5.1 Ilustração de uma linha na interface gráfica. 70
5.2 Ilustração do problema da "curva impossível". 76
5.3 Uma possível solução do problema da "curva impossível". 77

Lista de Algoritmos

1 Exemplo de Heurística Construtiva 10
2 Exemplo de Heurística de Busca Local 11
3 A Metaheurística LNS 12
4 A Metaheurística VNS 13
5 14

6 Heurística de Adição de Mercadorias de Pearn e Chien 20
7 Heurística de Boctor et al. 21
8 Heurística de Riera-Ledesma e Salazar-González 22
9 Procedimento l-consecutive-exchange 22
10 Heurística de Bontoux e Feillet 23
11 Heurística de Goldbarg, Bagi e Goldbarg 24
12 Heurística H8 26
13 H8 Fase 1: Algoritmo de Construção de uma Solução Viável 27
14 H8 Fase 2: VND 30
15 H8 Fase 3: LNS 30
16 H8 Fase 2 para UTPP Simétrico 31

17 HCM Fase 1: Algoritmo de Construção de uma Solução Viável 51
18 HCM Fase 2: VND 54
19 HCM Fase 3: LNS 55

Lista de Acrônimos

BPP Problema do Empacotamento em Caixas [*Bin Packing Problem*] 7

Cm-RSP Problema Capacitado dos m-Anéis Estrelados [*Capacitated m-Ring-Star Problem*] 2

CVRP Problema de Roteamento de Veículos com Restrições de Capacidade [*Capacitated Vehicle Routing Problem*] 5

CmTPP Problema dos Múltiplos Caixeiros Compradores Viajantes Com Restrições de Capacidade [*Capacitated m Traveling Purchasers Problem*] 2

CTPP Problema do Caixeiro Comprador Viajante Capacitado 21

HFVRP Problema de Roteamento de Veículos com Frota Heterogênea Fixa [*Heterogeneous Fleet VRP*] 8

LNS Large Neighborhood Search [*LNS*] 12

MDVRP Problema de Roteamento de Veículos com Múltiplos Depósitos [*Multiple Depot VRP*] 8

MFVRP Problema de Roteamento de Veículos com Decisão de Frota Mista [*Mix Fleet VRP*] 8

OVRP Problema de Roteamento de Veículos com Rotas "Abertas" [*Open VRP*] 8

PDVRP Problema de Roteamento de Veículos com Coleta e Entrega [*Pick-Up and Delivery VRP*] 8

PTFSU Problema do Transporte Fretado dos Servidores da UNICAMP 3

TPP Problema do Caixeiro Comprador Viajante [*Traveling Purchaser Problem*] 2

TSP Problema do Caixeiro Viajante [*Traveling Salesman Problem*] 7

UNICAMP Universidade Estadual de Campinas 1

UNITRANSP Departamento de Transportes da Prefeitura do Campus 1

UTPP Problema do Caixeiro Comprador Viajante não-Capacitado [*Uncapacitated TPP*] 21

VNS Variable Neighborhood Search [*VNS*] 12

VND Variable Neighborhood Descendent [*VND*] 13

VRP Problema de Roteamento de Veículos [*Vehicle Routing Problem*] 5

VRPSD Problema de Roteamento de Veículos com Entregas Fracionadas [*Split Delivery VRP*] 8

VRPTW Problema de Roteamento de Veículos com Restrições de Janela de Tempo [*Vehicle Routing Problem with Time Windows*] 8

Capítulo 1

Introdução

O serviço de transporte fretado tornou-se uma prática comum no dia-a-dia de instituições de grande porte do país – tais como a 3M do Brasil, o Sistema Brasileiro de Televisão e a BASF – principalmente a partir da publicação da lei número 7.418 que regulamentou o uso de vale-transportes [19]. Isto porque o fretado confere mais conforto, comodidade e segurança aos colaboradores destas empresas ao mesmo tempo em que diminui os riscos de atrasos e outros contratempos, influenciando diretamente o nível de produtividade nestas organizações.

A Universidade Estadual de Campinas (UNICAMP) também pertence ao grupo de instituições que provêem um sistema de fretamento de ônibus para a locomoção de parte de seus funcionários entre suas residências e locais de trabalho. Este sistema é administrado pelo Departamento de Transportes da Prefeitura do Campus (UNITRANSP).

São os técnicos da própria UNITRANSP que projetam as linhas de ônibus e contratam empresas de transporte de passageiros através de licitações públicas, a fim de implementarem o serviço na prática. Atualmente, o sistema compreende mais de 80 linhas de ônibus – cada uma delas é composta de duas rotas: uma no sentido residência/Universidade e outra no sentido inverso – que atendem a cerca de 2.500 servidores. O custo de manutenção deste serviço é de cerca de R$ 7 milhões anuais [53].

Além do alto investimento para a universidade, o processo de projeto das linhas é complexo e demorado, e o resultado final bastante dependente da habilidade do engenheiro de rotas – uma vez que as rotas são traçadas manualmente sobre mapas em papel e que existem muitos requisitos complicantes a serem atendidos.

Neste contexto, a automatização do processo de construção e manutenção das rotas torna-se desejável e se revela interessante sob pelo menos três aspectos práticos: (a) o trabalho do técnico é facilitado; (b) as soluções geradas são consistentes entre si mesmo quando projetadas por especialistas diferentes, pois o resultado não dependerá exclusivamente da habilidade do engenheiro de rotas; e (c) a *minimização* do tamanho do percurso

das rotas traz uma oportunidade real de economia para a universidade, visto que o custo de contratação de uma linha é proporcional ao comprimento total dos percursos das suas rotas.

Esta dissertação tem como objetivo prover um pacote de software para a automatização deste processo, a fim de diminuir o tempo do projeto de rotas otimizadas e de gerar economia para a instituição pública de ensino em questão.

Para este fim, é necessário estudar os problemas *ricos*[1] de roteamento enfrentados pela UNITRANSP; projetar e implementar algoritmos de otimização para estes problemas; e construir uma interface gráfica com o usuário para a operação destes algoritmos.

Três problemas de otimização são estudados no decorrer desta dissertação. Dois destes correspondem a operações simples no sistema de transporte da UNICAMP: (1) a otimização de uma rota e (2) a otimização simultânea de múltiplas rotas do mesmo turno. O terceiro problema surge da necessidade da manutenção do sistema como um todo e abrange, além de outras características mais específicas, os dois problemas anteriores.

Enumeramos estes problemas nos itens que seguem:

- No caso da **inserção ou remoção de um funcionário do sistema**, é necessário otimizar apenas as duas rotas da linha em que o funcionário foi alocado, uma por vez e de modo independente das outras cadastradas no sistema. O problema que emerge desta situação é conhecido na literatura como Problema do Caixeiro Comprador Viajante (TPP) [46] – do inglês *Traveling Purchaser Problem*, que é analisado em detalhes no Capítulo 3;

- No caso da **otimização do conjunto de rotas do sistema que atendem funcionários de um único turno de trabalho**, parte das rotas de um único turno é removida (por ocasião da remoção de linhas cujos períodos de licitação venceram, por exemplo), o que faz um subconjunto de funcionários deixar de ser atendido. Sendo assim, novas rotas otimizadas devem ser criadas a fim de atender estes colaboradores novamente. O problema que resulta desta operação é o Problema dos Múltiplos Caixeiros Compradores Viajantes Com Restrições de Capacidade (CmTPP) – do inglês *Capacitated m Traveling Purchasers Problem*, proposto nesta dissertação. O CmTPP é um modelo rico de roteamento de veículos que possibilita uma abordagem unificada de outros problemas da literatura, entre eles o Problema Capacitado dos m-Anéis Estrelados (Cm-RSP) [5] – do inglês, *Capacitated m-Ring-Star Problem*. O CmTPP é descrito em maiores detalhes no Capítulo 4;

[1] Problemas/modelos *ricos* são aqueles que tratam de diversas restrições simultaneamente com o objetivo de modelar situações cada vez mais próximas daquelas encontradas na prática. Veja mais na Seção 2.1.2.

- Já na **manutenção de todo o conjunto de rotas do sistema**, é necessário gerenciar as rotas e os funcionários cadastrados, absorvendo de forma facilitada as modificações que ocorrem no sistema ao longo do tempo. O problema que surge desta situação é chamado nesta dissertação de Problema do Transporte Fretado dos Servidores da UNICAMP (PTFSU). Este problema pode ser visto como uma extensão do CmTPP que considera funcionários de vários turnos de trabalho simultaneamente. Utiliza-se, para a resolução deste problema, a interação do o usuário final com o sistema por meio de um ambiente gráfico intuitivo, o qual permite que o usuário invoque os métodos de otimização e opere sobre os resultados destes, iterativamente, conforme descrito no Capítulo 5.

Em cada um destes três capítulos o leitor encontrará, logo no início, uma breve revisão da literatura dos problemas e métodos relacionados aos estudados nos próprios.

As principais contribuições desta dissertação são:

- estudo e levantamento dos dados do problema prático de transporte de funcionários da UNICAMP;
- projeto e implementação de um algoritmo heurístico para o TPP que, até onde conhecemos, é superior àqueles encontrados atualmente na literatura;
- proposta de um modelo rico de roteamento, o CmTPP, que provê uma abordagem unificada para a solução de outros problemas da literatura, em conjunto com um algoritmo heurístico para a resolução deste modelo;
- a construção de uma interface gráfica que possibilita o uso dos algoritmos projetados no ambiente da UNITRANSP.

A solução proposta será validada pela UNITRANSP, através de um pacote de software de roteamento que integra os algoritmos de otimização estudados a um sistema de informações geográficas.

O restante texto desta dissertação está organizado como segue. O Capítulo 2 apresenta alguns fundamentos teóricos necessários para a melhor compreensão do trabalho desenvolvido. O Capítulo 3 discute o Problema do Caixeiro Comprador Viajante, enquanto o Capítulo 4 trata do Problema dos Múltiplos Caixeiros Compradores Viajantes Com Restrições de Capacidade. A seguir, o Capítulo 5 descreve a solução proposta para a manutenção do sistema de transporte da UNICAMP, em funcionamento na UNITRANSP. Finalmente, o Capítulo 6 apresenta as conclusões da dissertação e sugere extensões para este trabalho.

Capítulo 2

Conceitos Básicos

Neste Capítulo o leitor é apresentado a uma breve revisão dos conceitos básicos necessários para uma melhor compreensão desta dissertação.

Na Seção 2.1, descreve-se a classe de problemas de roteamento de veículos na sua generalidade, também denominada pela literatura de Problema de Roteamento de Veículos (VRP) – do inglês, *Vehicle Routing Problem* – e introduz-se o Problema de Roteamento de Veículos com Restrições de Capacidade (CVRP). O CVRP é um dos problemas mais simples da classe dos VRPs e a sua exposição neste Capítulo tem o propósito de familiarizar o leitor com a notação empregada no restante do texto e amparar a discussão sobre outros problemas desta classe que surgem através da inclusão de novas *restrições* no modelo do CVRP.

Sabe-se que os problemas de roteamento de veículos, até mesmo os mais simples como o CVRP, são problemas computacionalmente difíceis. Isto pode ser evidenciado pelo fato de pertencerem à classe de problemas $\mathcal{NP}$-difícil [17]. Este fato reduz a possibilidade de se obter a solução exata de instâncias grandes, como as que surgem do problema da UNICAMP, em um baixo tempo computacional. Por este motivo utilizamos métodos heurísticos como uma alternativa para a resolução aproximada destas instâncias. Na Seção 2.2, alguns dos conceitos sobre heurísticas são introduzidos e os métodos heurísticos os quais serviram de alicerce para os algoritmos produzidos nesta dissertação são descritos.

2.1 Problema de Roteamento de Veículos

O Problema de Roteamento de Veículos consiste no planejamento de rotas para o transporte de pessoas ou produtos. Ele está presente em inúmeras situações práticas, tais como nos serviços de entrega de mercadorias, no abastecimento de clientes, na coleta de lixo e no caso prático abordado neste trabalho: a construção e manutenção das linhas de ônibus fretados para o transporte de funcionários de uma empresa.

Resolver este problema significa construir e *otimizar* um conjunto de rotas que atenda a todos os funcionários e obedeça a restrições operacionais como a capacidade de cada veículo e o intervalo de tempo no qual os funcionários devem ser atendidos. O objetivo final é encontrar rotas que *minimizam* os custos de transporte, reduzindo-se a distância total a ser percorrida pelos veículos e/ou o número de veículos utilizados.

Na literatura, esta classe de problemas é frequentemente apresentada em termos da criação de rotas para um conjunto de *veículos* que estão inicialmente estacionados em um *depósito* que devem visitar um conjunto de *clientes* ou *cidades* espalhados geograficamente em uma região. No caso prático abordado neste trabalho, a UNICAMP toma lugar do depósito e os *funcionários* são os clientes do sistema. Além disto, cada rota construída é usada por um *ônibus*.

Uma exposição detalhada da variante do VRP que considera a capacidade dos veículos segue.

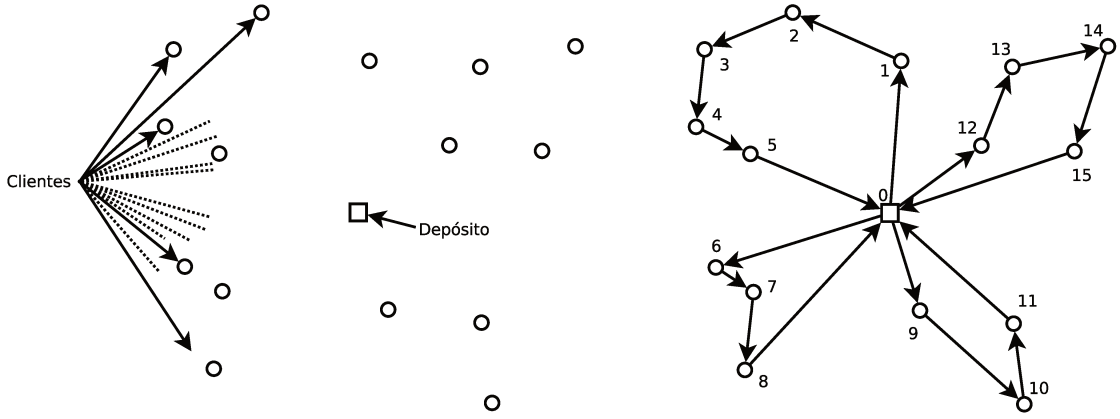
2.1.1 O Problema de Roteamento de Veículos com Restrições de Capacidade

A versão *capacitada* do problema pode ser modelada com auxílio de um grafo, da seguinte forma. Seja $G = (V, E)$ um grafo conexo, $V = \{v_0, v_1..., v_n\}$ o conjunto dos seus vértices e $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$, o conjunto de suas arestas. O vértice v_0 representa o depósito e é a base de uma frota de veículos idênticos de capacidade Q, enquanto os vértices remanescentes correspondem às cidades ou aos clientes. Cada cliente v_i, $i \neq 0$, tem uma demanda não negativa q_i, enquanto $q_0 = 0$ representa a demanda do depósito. Supõe-se que existe um número limitado m de veículos no depósito. Um peso não negativo c_{ij} está associado a cada aresta (v_i, v_j), com i e $j \in V$. Este peso representa o custo do uso da aresta na solução. Geralmente o custo c_{ij} é proporcional à distância ou tempo de percurso entre os vértices v_i e v_j.

O Problema de Roteamento de Veículos com Restrições de Capacidade (CVRP) [13] consiste em determinar o conjunto de rotas que deverão ser percorridas pelos veículos minimizando o custo de transporte, dado que são respeitadas as seguintes condições:

1. Cada rota começa e termina no depósito, v_0;
2. Todo vértice v_i, com $i \neq 0$, é visitado somente uma vez e por somente uma rota;
3. A soma da demanda dos clientes de cada rota não deve exceder a capacidade Q do veículo usado.

Assim, uma solução do CVRP pode ser representada por um conjunto de rotas cada uma delas dada por uma seqüência de vértices (depósito ou clientes), numerados de 0 a n, que sempre começa e termina no depósito.



(a) Entrada Típica de um CVRP (b) Solução de um CVRP

Figura 2.1: Uma Instância de Entrada e uma Solução do CVRP

Na Figura 2.1(a) observa-se um exemplo de instância de entrada para o CVRP. Nesta figura, os círculos representam as localizações dos clientes e o quadrado representa a localização do depósito.

Uma solução possível do problema – considerando-se uma frota de veículos de capacidade máxima $Q = 5$ e consumidores de demanda $q_i = 1$ – é o conjunto de rotas representado na Figura 2.1(b). Nesta solução, quatro veículos saem do depósito, visitam um subconjunto de clientes e voltam ao depósito novamente, seguindo o sentido das setas. Podemos representar esta solução S como:

$$S = \{(0, 1, 2, 3, 4, 5, 0), (0, 6, 7, 8, 0), (0, 9, 10, 11, 0), (0, 12, 13, 14, 15, 0)\}.$$

Sabe-se que o CVRP descrito nos parágrafos anteriores é uma generalização de dois problemas bastante estudados na Teoria de Computação: o Problema do Caixeiro Viajante (TSP)[1] [16] – do inglês, *Traveling Salesman Problem* – Problema do Empacotamento em Caixas (BPP)[2] [31] – do inglês, *Bin Packing Problem*.

O Problema do Caixeiro Viajante pode ser visto como um caso particular do CVRP no qual somente um veículo é permitido e a capacidade de carga Q do mesmo é igual a soma

[1]Veja uma descrição sucinta do problema em B.1
[2]Veja uma descrição sucinta do problema em B.2

da demanda dos clientes. Já o Problema do Empacotamento em Caixas é equivalente a um caso onde o número de veículos m é, também, o número de caixas disponíveis e o custo associado a todas as arestas do grafo é nulo, ou seja, $c_{ij} = 0$ para todo $i, j \in V$.

Restrições específicas impostas às soluções do CVRP dão origem a outras variantes mais complexas do problema. Algumas das encontradas mais frequentemente na literatura estão listadas abaixo:

1. **Problema de Roteamento de Veículos com Restrições de Janela de Tempo (VRPTW):** cada cliente deve ser atendido dentro de uma janela de tempo [2, 14, 27, 33, 44, 45, 56];
2. **Problema de Roteamento de Veículos com Múltiplos Depósitos (MD-VRP):** mais de um depósito é utilizado [44];
3. **Problema de Roteamento de Veículos com Rotas "Abertas" (OVRP):** a distância considerada em cada rota é a distância desde o primeiro cliente atendido até o depósito. [44]
4. **Problema de Roteamento de Veículos com Coleta e Entrega (PDVRP):** cada cliente tem um ponto de carregamento e um ponto de entrega definidos, que não são necessariamente um depósito central [50];
5. **Problema de Roteamento de Veículos com Entregas Fracionadas (VRPSD):** a demanda dos clientes pode ser dividida entre mais de um veículo [3];
6. **Problema de Roteamento de Veículos com Decisão de Frota Mista (MF-VRP) e Problema de Roteamento de Veículos com Frota Heterogênea Fixa (HFVRP):** deve-se decidir a composição da frota heterogênea de veículos ou deve-se usar os uma frota heterogênea pré-existente [18, 23, 38, 43, 57, 61].

2.1.2 Variantes *Clássicas* e Variantes *Ricas* do VRP

Desde que o problema foi introduzido na literatura, por Dantzig e Ramser em 1959 [17], centenas de outros trabalhos foram dedicados à solução de suas variantes e a bibliografia disponível sobre o VRP é extensa. A tarefa de compilar os trabalhos publicados sobre o assunto atraiu esforços consideráveis da comunidade científica que resultaram em textos relevantes, dentre eles o livro editado por Toth e Vigo [58] e uma atualização por Cordeau [15].

Algumas variantes do VRP – como o CVRP e o VRPTW – são chamadas de clássicas pelo fato de aparecerem frequentemente na literatura. Estas variantes geralmente consideram poucas restrições operacionais simultaneamente e estão distantes de modelarem problemas que aparecem na prática.

Recentemente, uma grande atenção é dada a variantes mais complexas do VRP, algumas vezes denominadas de VRPs "ricos" (cf. [9, 26]), cujos modelos estão mais próximos de problemas práticos de distribuição que os modelos de VRP mais comuns na literatura. Em particular, essas variantes são caracterizadas por considerarem diversas restrições simultaneamente. Um *survey* específico sobre VRPs ricos pode ser encontrado no trabalho de Bräysy e colaboradores [9].

Nesta dissertação são propostos métodos para resolver os modelos ricos que surgem da situação real do fretamento de ônibus enfrentado pela Universidade Estadual de Campinas para o transporte de seus funcionários de suas respectivas residências aos locais de trabalho e vice-versa.

A análise de algoritmos heurísticos para estes problemas justifica o título da dissertação.

2.2 Heurísticas e Metaheurísticas

Sabe-se que, até mesmo uma das variantes mais simples de roteamento de veículos, o CVRP, descrito na Seção anterior, é um problema da classe $\mathcal{NP}$-difícil. Na prática este fato reduz a possibilidade de se obter algoritmos eficientes para resolvê-la de forma exata. Assim, a utilização de métodos heurísticos se justifica como uma forma de encontrar boas soluções em tempo razoável, ainda que não sejam as melhores soluções possíveis.

O objetivo desta Seção é introduzir alguns dos conceitos sobre heurísticas e metaheurísticas da área de Otimização Combinatória, além de apresentar os arcabouços metaheurísticos que serviram de alicerce para os métodos implementados neste trabalho.

2.2.1 Heurísticas

Chama-se de *Heurística* um algoritmo que produz rapidamente uma solução viável – preferencialmente, de boa qualidade – mesmo que não seja garantidamente a solução ótima.

O estudo das heurísticas é, geralmente, dividido em duas áreas:

Heurísticas *Construtivas* literalmente constroem soluções a partir do zero. Estas heurísticas começam com uma solução parcial vazia e, iterativamente, inserem elementos nesta, até que ela se torne uma solução completa (veja o Algoritmo 1 e Figura 2.2(a)). A cada iteração, o elemento que será introduzido na solução é escolhido por meio de um critério guloso e/ou probabilístico e, a função de seleção do próximo elemento é atualizada a fim de refletir o critério escolhido nas próximas iterações;

Heurísticas de *Busca Local* executam uma sequência de mudanças locais sobre uma solução inicial – fornecida possivelmente por uma heurística construtiva – a fim de melhorar o valor da função objetivo a cada passo, até que um mínimo local seja encontrado. Ou seja, a cada iteração, uma solução x' da *vizinhança* $\mathcal{N}(x)$ toma o lugar da solução corrente x. O processo iterativo para quando não se pode obter outra solução melhor que a atual em sua vizinhança (ver Algoritmo 2 e Figura 2.2(b)).

Algoritmo 1: Exemplo de Heurística Construtiva

```
1 inicialização;
2 enquanto a condição de parada não é atendida faça
3 |   seleciona um elemento;
4 |   adiciona o elemento escolhido à solução atual;
5 |   atualiza a função de seleção;
6 retorna a solução encontrada;
```

Chama-se de *vizinhança* de x o conjunto $\mathcal{N}(x)$ de soluções obtidas ao se aplicarem *pequenas* perturbações – que atendem a um conjunto bem definido de regras – na solução x.

Uma solução $x' \in \mathcal{N}(x)$ é chamada de *solução vizinha* da solução original, x. *Estrutura da vizinhança* é o nome que se dá ao conjunto de regras que definem as perturbações utilizadas para gerar a vizinhança.

A definição destas estruturas é bastante dependente do problema que está sendo tratado. Um exemplo de estrutura possível para CVRP é permitir somente a troca de alguns clientes entre duas rotas. Outro exemplo de estrutura de vizinhança para o mesmo problema é permitir apenas a troca da ordem em que o veículo visita os clientes dentro da mesma rota em que se encontram. Mais exemplos de estruturas de vizinhança serão vistos nos próximos Capítulos desta dissertação, no qual definimos estruturas de vizinhança para o TPP e para o CmTPP.

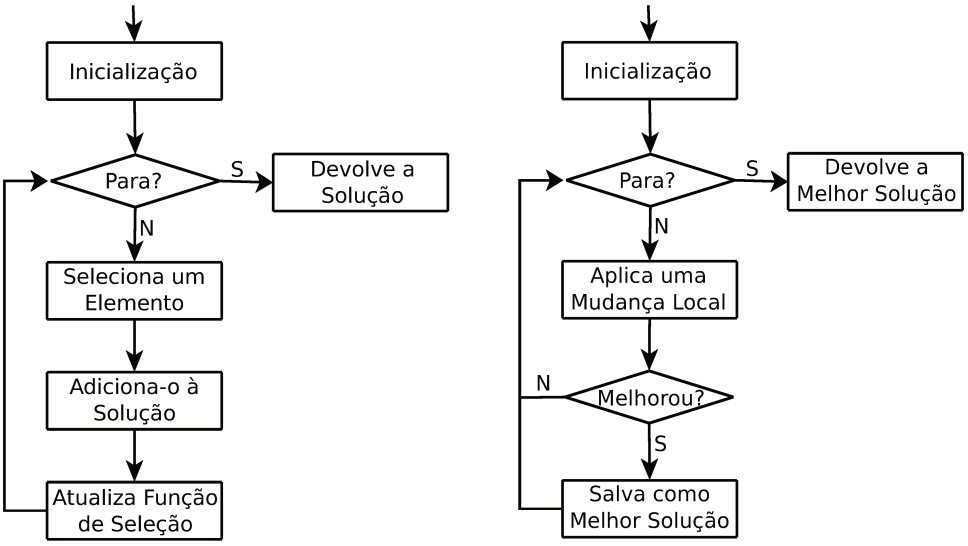
Assim apresentamos as definições sobre os algoritmos de busca local que são usadas neste trabalho. Um tratado sobre os algoritmos de Busca Local para diversos problemas clássicos de otimização combinatória e outros exemplos de estruturas de vizinhança é encontrado no livro de Aarts e Lenstra [1] e é recomendado ao leitor que deseje aprofundar seu conhecimento no assunto.

2.2.2 Metaheurísticas

Recentemente, pesquisadores tem proposto novas camadas de controle sobre heurísticas simples, como as construtivas e de busca local. Estas camadas de controle estendem as

Algoritmo 2: Exemplo de Heurística de Busca Local

```
1 inicialização;
2 enquanto a condição de parada não é atendida faça
3     encontra uma perturbação da solução atual;
4     se a perturbação encontrada melhora a solução então
5         executa a perturbação;
6         salva a solução perturbada como sendo a melhor solução;
7 retorna a melhor solução encontrada;
```



(a) Heurística Construtiva (b) Heurística de Busca Local

Figura 2.2: Fluxogramas da Heurística Construtiva e da Heurística de Busca Local

heurísticas de várias formas, a fim de que a trajetória de busca não fique presa a um mínimo local.

Dá-se o nome de *metaheurísticas* a estas camadas de controle adicionais, que modificam o comportamento das heurísticas sobre as quais elas foram adaptadas.

Atualmente, as metaheurísticas conhecidas como Large Neighborhood Search (LNS) e Variable Neighborhood Search (VNS) tem se mostrado muito eficazes na resolução de problemas de roteamento. Nos parágrafos que seguem apresentamos estas metaheurísticas. Os conceitos nelas explorados foram utilizados como fundamentos para os algoritmos implementados neste trabalho.

Large Neighborhood Search

A metaheurística chamada de Large Neighborhood Search foi proposta por Shaw [52] em 1998 e estendida por Ropke e Pissinger [51] em 2005. A LNS é um algoritmo iterativo que destrói uma parte da solução a cada iteração, usando uma heurística de "desconstrução" e a re-otimiza na tentativa de se encontrar uma solução de melhor qualidade. O resultado da primeira etapa – a de desconstrução – é uma solução parcial não necessariamente viável. Na segunda etapa – a de re-construção – uma solução viável é "re-construída" de forma a utilizar inteligentemente a informação contida na solução parcial que resulta da primeira etapa.

Esta metaheurística também pode ser vista como um método que alterna entre a fixação de algumas características da solução e a re-otimização das características restantes.

Algoritmo 3: A Metaheurística LNS

```
1 inicialização;
2 enquanto a condição de parada não é atendida faça
3 |   aplica a heurística H⁻ de "Desconstrução";
4 |   aplica a heurística H⁺ de "Reconstrução";
5 |   se a solução for a melhor encontrada até o momento então
6 |   |   salva como melhor solução;
7 retorna a melhor solução encontrada;
```

Variable Neighborhood Search

A metaheurística Variable Neighborhood Search foi proposta por Mladenovic e Hansen [40] em 1997. .

Uma heurística de busca local simples segue sempre a mesma estrutura de vizinhança até encontrar o mínimo local. A VNS, por sua vez, não segue a trajetória de uma única

estrutura de vizinhança. Ela explora estruturas de vizinhanças cada vez mais distantes da solução corrente, aceitando novas soluções somente quando há uma melhoria no custo da função objetivo.

Desta forma, VNS consegue – frequentemente – manter as características positivas da melhor solução corrente (como por exemplo, muitas variáveis que já estão em seu valor ótimo) que serão usadas para obter novas soluções na vizinhança, enquanto escapa dos mínimos locais da função objetivo com maior habilidade do que uma busca local simples, pela característica de explorar mais do que uma estrutura de vizinhança.

Um exemplo de VNS se encontra no Algoritmo 4. Esta metaheurística é alimentada com uma solução inicial e com um conjunto de vizinhanças $\mathcal{V}$, para o problema em questão.

Iterativamente, uma estrutura deste conjunto de vizinhanças é selecionada e aplicada na solução, gerando uma perturbação no caminho da busca. Esta solução é então melhorada por alguma busca local. Se a solução produzida por estes passos provoca uma melhoria no custo da função objetivo ela é aceita e a primeira estrutura do conjunto de vizinhanças é escolhida novamente. Caso contrário, a próxima estrutura do conjunto de vizinhanças é selecionada. O algoritmo para quando o número de iterações alcança um limite previamente especificado.

Algoritmo 4: A Metaheurística VNS

```
1  inicialização;
2  V = conjunto de vizinhanças para o problema em questão;
3  enquanto a condição de parada não é atendida faça
4      escolha a primeira vizinhança v no conjunto V;
5      aplique um movimento qualquer de v na solução atual gerando uma perturbação no
       caminho de busca;
6      aplique uma busca local para melhorar a solução perturbada;
7      se o valor da função objetivo desta nova solução é o melhor encontrado então
8          salva a solução atual como melhor solução;
9          volte ao passo 3
10     senão
11         escolha a próxima vizinhança v no conjunto V;
12         volte ao passo 5
13 retorna a melhor solução encontrada;
```

Uma heurística de busca local frequentemente utilizada – no Passo 6 do Algoritmo 4 – de implementações da VNS é a Variable Neighborhood Descendent (VND), também descrita por Hansen e Mladenovic [25]. A VND (ver Algoritmo 5) consiste em utilizar as vizinhanças escolhidas numa ordem pré-determinada e, tão logo que um movimento que melhora o custo da função objetivo, voltar a utilizar a primeira vizinhança da ordem

novamente. O algoritmo é aplicado até que a última vizinhança da ordem chegue a um mínimo local.

Algoritmo 5:

```
1  inicialização;
2  V = conjunto de vizinhanças para o problema em questão;
3  enquanto a condição de parada não é atendida faça
4      escolha a primeira vizinhança v no conjunto V;
5      aplique o melhor movimento de v na solução atual;
6      se o valor da função objetivo foi melhorado então
7          salva a solução atual como melhor solução;
8          volte ao passo 3
9      senão
10         escolha a próxima vizinhança v no conjunto V;
11         volte ao passo 5
12 retorna a melhor solução encontrada;
```

As Metaheurísticas LNS e VNS na literatura do VRP

A escolha destes dois paradigmas, LNS e VNS, como base para a criação dos métodos heurísticos nesta dissertação foi motivada por trabalhos de outros autores que apresentam metaheurísticas bastante eficazes para problemas reais e de grande porte, como aqueles que abordamos nesta dissertação, e por este motivo influenciaram a direção da nossa pesquisa. Revisamos, a seguir, três destes trabalhos.

Pissinger e Ropke [44] desenvolveram uma heurística unificada na qual instâncias de cinco variantes do VRP são transformadas em instâncias de um modelo rico de problema de *coleta e entrega* que é, então, resolvido num LNS. A heurística é capaz de resolver o problema de roteamento de veículos com janelas de tempo (VRPTW), com restrições de capacidade (CVRP), com múltiplos depósitos (MDVRP), com restrições dependentes de sítio (SDVRP) e o problema de roteamento de veículos aberto (OVRP). Alguns dos resultados obtidos por essa heurística estão entre os melhores já obtidos para instâncias de Solomon [56], Homberger [27]. O maior atrativo dessa heurística é o baixo custo computacional para obter soluções de boa qualidade, característica bastante atraente quando se trata da resolução de instâncias grandes obtidas na prática.

Prescott-Gagnon e colegas [45] propuseram outra abordagem para o VRPTW, desta vez fundamentada sobre uma algoritmo de geração de colunas heurístico inserida

no contexto da LNS. Esta metaheurística obteve resultados bastante significativos nas instâncias de benchmark, especialmente nas instâncias maiores, de 200 a 1000 clientes. O algoritmo alcançou 145 novas melhores soluções de 356 instâncias do conjunto de instâncias de [56] e [27].

Kytöjoki e colegas [35] apresentaram um algoritmo VNS que encontra rapidamente boas soluções para o CVRP com até vinte mil (20.000) clientes. A metaheurística atua em duas fases. Primeiro encontra uma boa solução inicial com um algoritmo construtivo de *inserção mais barata* que é hibridizado com sete (7) heurísticas de busca local. Estas heurísticas de busca local são aplicadas em vários estágios da construção. Depois, na segunda fase da metaheurística, as estruturas de vizinhança das mesmas 7 heurísticas de melhoramento são aplicadas numa estratégia VNS.

Capítulo 3

O Problema do Caixeiro Comprador Viajante

Apontamos, ao final do Capítulo 1, que os problemas estudados nesta dissertação surgem de operações sobre o sistema de roteamento de ônibus fretados para o transporte de funcionários na UNICAMP.

Neste Capítulo estudamos o primeiro destes problemas: a otimização de uma única rota de ônibus quando se considera a distância máxima que o funcionário pode percorrer a pé de sua residência até o ponto de embarque. Esta situação é vista aqui como um caso especial do problema conhecido na literatura como o Problema do Caixeiro Comprador Viajante (TPP) [21] – do inglês, *Traveling Purchaser Problem* [7, 22, 42, 46] – uma importante *generalização* do Problema do Caixeiro Viajante, conforme veremos no decorrer do texto.

3.1 Introdução

O TPP foi originalmente proposto por Ramesh [46] em 1967 e pode ser definido formalmente como um problema em grafos, como segue. Seja $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ um grafo completo no qual um conjunto $\mathcal{V} = \{v_0, ..., v_n\}$, $|\mathcal{V}| = n + 1$, onde v_0 representa uma entidade especial chamada de *base* ou *depósito* e $\{v_1, ..., v_n\}$ representa o conjunto de mercados de uma região. As arestas do conjunto $\mathcal{E}$ representam os menores caminhos entre os elementos de $\mathcal{V}$ e seus pesos $c_{i,j}$, com $(i, j) \in \mathcal{E}$, refletem os custos de viagem entre os mesmos.

Considere, também, $\mathcal{P}$ o conjunto de todos os produtos disponíveis nos mercados. Cada produto $p \in \mathcal{P}$ pode ser comprado somente num subconjunto $\mathcal{V}_p \subseteq \mathcal{V}$ de mercados em que é ofertado, uma vez que cada mercado $v \in \mathcal{V}$ oferta somente um subconjunto $\mathcal{P}_v \subseteq \mathcal{P}$ de produtos.

Dada uma lista com a demanda d_p de cada produto p (esta lista é chamada de *lista*

de requisições a partir deste ponto), o objetivo do TPP é encontrar uma rota que passa por um subconjunto de mercados a serem visitados pelo comprador a fim de que ele possa adquirir todos os produtos na quantidade necessária para atender a demanda declarada na lista de requisições, de tal forma que o custo total de transporte e de compra dos produtos seja minimizado.

Assume-se que: (a) cada mercado pode ser visitado apenas uma vez; que (b) o preço unitário $u_{p,v}$ e a quantidade ofertada $q_{p,v}$ de um mesmo produto p pode variar dependendo do mercado $v \in \mathcal{V}_p$ considerado; e, (c) o depósito não oferta produto algum.

Uma solução S para o problema é representada por dois conjuntos de dados: (1) a sequência de mercados que o comprador visita, chamada de *rota* e (2) a quantidade de cada produto que é comprada em cada um dos mercados visitados, ou *lista de aquisições*. É possível avaliar o custo de uma solução a partir da sua rota e de sua lista de aquisições.

Na prática, o TPP pode ser visto como um problema logístico de dois níveis que devem ser resolvidos simultaneamente – enquanto o primeiro nível se concentra nas decisões a respeito de quais produtos comprar, de que mercados e em que quantidade, o segundo considera as decisões acerca da programação da viagem do comprador viajante de forma a percorrer o menor caminho entre o depósito e mercados visitados.

No caso específico deste trabalho, o primeiro nível corresponde ao agrupamento dos funcionários em pontos de ônibus; e, o segundo, consiste em construir a rota do ônibus pelos pontos escolhidos.

Outras aplicações comuns do TPP ocorrem no transporte e armazenamento de produtos [54]. Uma aplicação interessante no campo do agendamento de produção é descrito por Buzacott e Dutta [11]. Nesta aplicação uma máquina multi-propósito pode assumir algumas configurações i e cada tarefa $k \in K$ deve ser executada usando uma configuração em um conjunto M_k. Custos de roteamento $c_{i,j}$ correspondem ao custo de troca entre as configurações da máquina e os custos de aquisição correspondem aos custos de execução das respectivas tarefas.

A partir da descrição do TPP em sua forma mais geral podemos estabelecer dois tipos de classificações, as quais apresentamos nos dois próximos parágrafos.

Distingue-se o problema entre *simétrico* e *assimétrico* de acordo com as características das arestas do grafo $\mathcal{G}$. Sempre que o custo de viajar diretamente de um mercado v_i a outro v_j for o mesmo que o custo para viajar no sentido oposto, para todo par de mercados $(v_i, v_j) \in \mathcal{V} \times \mathcal{V}$, dizemos que o TPP é simétrico. Caso contrário, problema é definido como assimétrico. A variante assimétrica pode ser encontrada na prática, por exemplo, quando consideramos o roteamento sobre um mapa digital de vias no qual algumas delas tem *mão* única. Neste caso, o menor caminho de *ida* de um mercado v_a para um mercado v_b pode ser diferente do menor caminho de *volta*. Note que a variante simétrica é, na verdade, um caso particular da variante assimétrica.

Com relação às demandas dos produtos na lista de requisições e suas ofertas nos mercados, podemos diferenciar o TPP em *capacitado* e *não capacitado*. Na versão não capacitada do TPP (UTPP, em inglês) assume-se que o mercado tem disponível quantidade suficiente do produto para atender demanda da lista de requisições por completo sempre que ele oferta um produto. Se pelo menos um mercado oferta algum produto p em quantidade menor do que a sua demanda d_p na lista de requisições então o problema é dito *capacitado* (CTPP, em inglês). Observe que a versão não capacitada do TPP pode ser considerada como um caso especial da versão capacitada em que as demandas e as ofertas dos produtos são unitárias, $d_p = 1$ e $q_{p,v} \in \{0, 1\}$, para todos os produtos $p \in \mathcal{P}$.

O restante deste Capítulo está organizado como se segue. Apresentamos uma breve revisão da literatura na Seção 3.2. Na Seção 3.3, os novos algoritmos heurísticos projetados neste trabalho são descritos. As estruturas de busca local e algumas estruturas de dados usadas nestes algoritmos são analisadas na Subseção 3.3.1 e na Seção 3.4, respectivamente. Os resultados computacionais obtidos são comparados com os da literatura na Seção 3.5. A Seção 3.6 apresenta as conclusões e algumas das possíveis extensões deste trabalho.

3.2 Revisão da Literatura

Descrevemos nesta seção alguns dos métodos heurísticos para o TPP publicados na literatura mais recente.

Pearn e Chien [42] apresentaram, em 1998, uma heurística chamada por eles de "algoritmo de adição de mercadorias em ordem aleatória", ou *randomized order commodity adding algorithm*, em inglês. Este algoritmo atende apenas à variante simétrica e não capacitada do TPP.

O funcionamento deste algoritmo é dividido em duas etapas. Na primeira, cria-se uma solução viável que é então melhorada por heurísticas de busca local na segunda. As etapas estão na listagem do Algoritmo 6.

No início de cada iteração deste algoritmo, gera-se, de forma aleatória, uma nova ordem para os produtos na lista de requisições. Então, constrói-se uma solução parcial inicial contendo o depósito e o mercado que minimiza o custo total (que corresponde ao custo de roteamento adicionado ao custo de compra) para compra do primeiro item que aparece na nova ordem da lista de requisições.

A cada iteração, um novo produto p da lista de requisições é considerado. Se a demanda d_p deste produto ainda não está atendida pela solução parcial, o procedimento insere na solução o mercado v que oferta este produto, $v \in V_p$, e que minimiza, entre todos os mercados de V_p, o incremento do custo de viagem somado com o custo de aquisição

do produto e inicia uma nova iteração, considerando o próximo produto da lista. Se a demanda do produto p já foi atendida, ele passa a considerar o próximo produto da lista sem inserir mercado algum à solução parcial.

A primeira fase termina assim que o algoritmo encontra uma solução que atende a toda demanda na lista de requisições.

Na segunda fase, uma heurística de busca local inspirado no trabalho de Basart e Huguet [6] e os procedimentos *market-drop* e *market-exchange* descritos na Subseção 3.3.1 são usados para melhorar a qualidade da solução encontrada na fase anterior.

As duas fases são repetidas um certo número de vezes, modificando-se apenas a ordem em que os produtos são considerados na primeira fase. Desta forma, há uma diversificação das soluções obtidas na primeira fase, fazendo com que as heurísticas de busca local explorem novas regiões do espaço de soluções, criando novas oportunidades para que soluções mais promissoras sejam obtidas.

Algoritmo 6: Heurística de Adição de Mercadorias de Pearn e Chien

```
1  para i = 1 até k faça
2      /* primeira etapa */
3      L = cria uma nova ordenação da lista de requisições;
4      v = mercado que minimiza o custo da compra do primeiro item da lista
       ordenada;
5      σ = solução parcial contendo o depósito e o mercado v;
6      para cada produto p a partir do segundo na lista L faça
7          m = mercado que oferta p e minimiza a soma do incremento no custo de
           roteamento e aquisição de p;
8          adiciona o mercado m na solução parcial σ;
9      /* segunda etapa */
10     aplica buscas locais em σ;
11     atualiza a melhor solução encontrada como σ, se apropriado;
12 retorna a melhor solução encontrada;
```

Faz-se necessário esclarecer que, o algoritmo de adição de mercadorias é descrito de forma ambígua por Pearn e Chien em [42]. Desta forma, a heurística descrita no Algoritmo 6 pode ser entendido como uma interpretação daquele texto.

Boctor et al. [7], 2003, estendem o trabalho de Pearn e Chien [42], desenvolvendo novos mecanismos de perturbação da solução para melhorar a qualidade das soluções obtidas pelo algoritmo de adição de mercadorias. Estes novos mecanismos propostos são aplicados após a criação de uma solução viável por uma heurística construtiva. Estes

foram os primeiros métodos da literatura testados nas instâncias da `TPPlib` – que fora utilizada como conjunto de instâncias de benchmark também no trabalho atual. Os algoritmos propostos por Boctor e colegas resolvem tanto o Problema do Caixeiro Comprador Viajante não-Capacitado (UTPP) como o Problema do Caixeiro Comprador Viajante Capacitado (CTPP) simétricos.

O Algoritmo 7 ilustra as fases da heurística. Na primeira fase uma solução viável é criada por um método construtivo baseado no Algoritmo 6. Na segunda fase, os procedimentos de perturbação da solução e operadores de busca local são aplicados a fim de melhorar a qualidade da solução obtida.

Algoritmo 7: Heurística de Boctor et al.

```
1 /* primeira etapa */
2 σ = solução viável criada através de uma heurística de adição de mercadorias;
3 /* segunda etapa */
4 aplica procedimentos de perturbação da solução e busca local em σ;
5 retorna a solução encontrada;
```

Riera-Ledesma e Salazar-González [48] publicaram, em 2005, um método heurístico eficiente para o UTPP e CTPP simétricos.

Este algoritmo também funciona em duas etapas. Na primeira etapa, uma solução inicial que contém todos os mercados é criada por meio de uma heurística construtiva do tipo *vizinho mais próximo* e melhorada através da aplicação do procedimento de Lin-Kernighan [37].

Esta solução é melhorada durante segunda etapa através da aplicação de duas famílias heurísticas de busca local (BL). O primeiro procedimento de BL, a *l-consecutive-exchange*, executa iterativamente a troca de l mercados consecutivos da solução viável incumbente por um conjunto de mercados que não pertencem ao ciclo. O segundo procedimento de BL, chamado de *market-add*, simplesmente insere mercados na solução enquanto esta operação implicar numa redução no custo da função objetivo.

Uma vez que a busca chegue a um mínimo local, durante a segunda etapa um esquema de diversificação denominado *shaking* é chamado para perturbar a solução incumbente e diversificar a busca.

Se o critério de parada for satisfeito, o procedimento *shaking* declara o término da busca.

Este algoritmo alcançou resultados significativamente melhores que os resultados do trabalho de Boctor e colegas [7].

Algoritmo 8: Heurística de Riera-Ledesma e Salazar-González

```
1  /* Primeira etapa */
2  σ = ciclo hamiltoniano criado com a heurística construtiva do tipo vizinho mais
   próximo;
3  σ = Lin-Kernighan(σ);
4  inicializa melhor solução encontrada com σ;
5  /* Segunda etapa */
6  repita
7  |  repita
8  |  |  σ = l-consecutive-exchange(σ);
9  |  |  σ = market-add(σ);
10 |  |  σ = Lin-Kernighan(σ);
11 |  até que não se obtenha melhoria no custo da solução;
12 |  adiciona mercados na solução parcial σ;
13 |  atualiza a melhor solução encontrada como σ, se apropriado;
14 até not Shaking(σ);
15 return a melhor solução encontrada.
```

A principal contribuição dos autores desta metaheurística foi a definição da estrutura de vizinhança de busca local l-consecutive-exchange (descrita na listagem do Algoritmo 9). Esta vizinhança é dividida em dois estágios. No primeiro – chamado de *l-consecutive-drop* – ocorre a remoção de l vértices consecutivos. Já no segundo estágio – chamado de *restore-feasibility* – recupera-se a viabilidade da solução, caso a solução tenha se tornado inviável com a remoção dos mercados no primeiro estágio.

Algoritmo 9: Procedimento l-consecutive-exchange

```
1  inicializa um valor de l;
2  enquanto l ≥ 1 faça
3  |  σ' = l-consecutive-drop(σ, l);
4  |  se σ' não é viável então
5  |  |  σ' = restore-feasibility(σ');
6  |  se f(σ') ≥ f(σ) ou σ' não é viável então
7  |  |  l = l − 1
8  |  senão
9  |  |  σ = σ';
10 return σ.
```

O procedimento *l-consecutive-drop* avalia a retirada das sequências de l mercados da solução, uma a uma, registrando a redução do custo de roteamento e no custo de aquisição

dos produtos. dos produtos. A sequência de l mercados que maximiza a soma destes valores é removida da solução incumbente.

O Procedimento *restore-feasibility* insere um conjunto de mercados na solução incumbente σ afim de torná-la viável.

O valor de l é reduzido toda vez que não se encontrar uma solução viável ou que a busca atingir um ótimo local. Este algoritmo alcançou resultados significativamente melhores que os apresentados no trabalho de Boctor e colegas [7], contudo estes resultados foram superados pelos dois trabalhos posteriores, apresentados nos próximos parágrafos.

Bontoux e Feillet [8] apresentaram, em 2008, uma heurística híbrida para o UTPP simétrico, baseada na meta-heurística colônia de formigas e buscas locais. Nesse algoritmo (veja listagem do Algoritmo 10) cada formiga da colônia constrói uma solução do TPP de modo iterativo, adicionando novos mercados em sua solução parcial explorando a informação obtida pela experiência passada através do feromônio depositado por outras formigas nas arestas do TPP, e por movimentos heurísticos baseados em critérios gulosos.

O algoritmo começa gerando uma primeira solução viável a partir da criação da primeira formiga da colônia. A cada iteração do laço principal toda formiga da colônia *anda* exatamente uma aresta até que aconteça uma de duas situações: (1) o custo cumulado da solução parcial da formiga até o momento ultrapassa uma função do custo da melhor solução encontrada até o momento ou (2) a formiga completou uma solução viável e chega ao depósito novamente.

No caso (1) a formiga morre e uma cópia da melhor formiga da história da busca *nasce* no depósito, simulataneamente. Caso aconteça (2) BLs são aplicadas para melhorar a solução da formiga e feromônio é depositado no caminho da solução melhorada numa quantidade que é função do custo final da solução obtida.

Goldbarg, Bagi e Goldbarg [22] publicaram, em 2009, uma heurística transgenética para o UTPP e CTPP simétricos. Um algoritmo transgenético é uma metaheurística evolutiva inspirada na transferência horizontal de genes e endosimbiose [20].

Inicialmente – acompanhe o Algoritmo 11 – uma população de cromossomos é gerada, através de uma heurística construtiva que insere mercados aleatoriamente até que o cromossomo represente uma solução viável. Cada rota da solução gerada é melhorada com a aplicação do procedimento de Lin-Kernighan. Um banco de informações do hospedeiro (BIH) é inicializado com um ciclo hamiltoniano melhorado através da aplicação de Lin-Kernighan e com as quatro melhores soluções da população atual.

A cada iteração gera-se um vetor transgenético – escolhe-se entre gerar um plasmídeo ou um transposon. Plasmídeos transferem informações genéticas do BIH para os cromos-

Algoritmo 10: Heurística de Bontoux e Feillet

```
1  gera uma solução viável inicial com a primeira formiga;
2  enquanto o tempo não acaba faça
3      para cada formiga faça
4          move a formiga para o próximo mercado;
5          se a formiga está no caso (1) então
6              mata a formiga;
7              cria um clone da melhor formiga no vértice do depósito;
8          se a formiga está no caso (2) então
9              aplica busca local na solução da formiga;
10             deposita feromônio pelo caminho da solução, de acordo com seu custo;
11             atualiza a melhor solução obtida, se apropriado;
12 return a melhor solução encontrada.
```

somos. Tranposons não usam as informações do BIH – eles realizam modificações locais em cada cromossomo utilizado.

A geração de plasmídeos é incentivada nas primeiras iterações e a de transposons, nas demais. A razão para isso é que, no início, as soluções geradas aleatoriamente provavelmente encontram-se distantes de ótimos locais. Portanto, trechos de informação obtidas das fontes do banco podem contribuir para melhorar as soluções aproximando-as de ótimos locais. Enquanto isso, os rearranjos genéticos produzidos pelos transposons realizam procedimentos de intensificação em partes do cromossomo, portanto, à medida que as soluções tornam-se melhores o emprego dos transposons se torna cada vez mais interessante.

Uma vez que um vetor é gerado ele é aplicado nos cromossomos da população. Atualiza-se o BIH e a melhor solução obtida, quando apropriado.

Algoritmo 11: Heurística de Goldbarg, Bagi e Goldbarg

```
1  gerar uma população inicial de cromosomos;
2  gerar banco de informações do hospedeiro;
3  para número de iterações faça
4      gerar um vetor transgenético (um plasmídeo ou um transposon);
5      aplicar um vetor gerado transgenético na população de cromossomos;
6      atualizar banco de informações do hospedeiro;
7      atualizar a melhor solução encontrada, se apropriado;
8  return a melhor solução encontrada.
```

3.3 Novo Algoritmo Heurístico para o TPP

Embora muitas heurísticas e metaheurísticas para o TPP já tenham sido publicadas na literatura, ainda há uma lacuna a ser preenchida: não há heurísticas que possibilitem a obtenção de boas soluções em curto intervalo de tempo de execução para instâncias de larga escala, como as advindas de problemas reais semelhantes ao que motivou este estudo.

A aplicação abordada neste trabalho demanda um método resolvedor que tenha uma resposta rápida – em tempo real ou quase – para cenários de centenas de funcionários, já que interage com um usuário do sistema (o engenheiro de rotas) através de uma interface gráfica, durante o projeto das linhas que são usadas na prática pelo sistema de roteamento de ônibus fretados da UNICAMP. Esta situação nos levou a propor um novo algoritmo heurístico para o TPP.

O novo algoritmo é apresentado na Subseção 3.3.1 e resolve tanto a variante simétrica quanto assimétrica do TPP, seja a instância capacitada ou não – considerando, assim, todas as variantes citadas no início deste capítulo.

Uma versão deste algoritmo otimizada para resolver a variante simétrica não capacitada do problema é apresentada na Subseção 3.3.2. Resultados desse algoritmo indicam que ele está entre os métodos resolvedores do estado-da-arte atuais para esta variante do problema.

A metaheurística proposta explora os conceitos da Large Neighborhood Search (LNS) e da Variable Neighborhood Search (VNS), que – até onde chega o nosso conhecimento – ainda não haviam sido aplicados simultaneamente na resolução de instâncias do TPP.

3.3.1 Uma Heurística Baseada em LNS-VNS para o TPP

Nossa abordagem progride de maneira iterativa em três fases distintas durante a resolução do Problema do Caixeiro Comprador Viajante. Na primeira fase geramos uma solução viável, obtida a partir de uma solução parcial arbitrária, através de uma heurística gulosa não determinística. Na segunda, um esforço é feito no sentido de reduzir o custo total da função objetivo. Nesta fase nós usamos a busca local VND. Na terceira fase, a próxima solução parcial que será usada na próxima iteração do algoritmo é preparada. É neste ponto que o algoritmo decide entre recomeçar a busca de uma solução parcial vazia ou aplica uma heurística de remoção de mercados fundamentada nos conceitos da Large Neighborhood Search. Chamamos este método de H8 e sua listagem é apresentada no Algoritmo 12.

A seguir, descrevemos mais detalhadamente as fases do algoritmo proposto:

Fase 1: Construção de uma solução viável. Nesta fase, uma heurística construtiva

Algoritmo 12: Heurística H8

```
1  σ = solução parcial que contém somente o depósito;
2  enquanto o número de iterações realizadas não é maior do que o número de
   iterações máximo faça
3      // transforma σ numa solução viável;
4      σ = resultado da aplicação da Fase 1 sobre σ;
5      // melhora a qualidade de σ pela aplicação de uma heurística VNS;
6      σ = resultado da aplicação da Fase 2 sobre σ;
7      se σ é de qualidade melhor que a melhor solução já encontrada então
8          salva σ como a melhor solução já encontrada;
9      //escolhe a próxima solução parcial que alimenta a próxima execução da Fase 1;
10     σ = resultado da aplicação da Fase 3 sobre σ;
11 retorna a melhor solução encontrada no caminho da busca;
```

inspirada no *algoritmo de adição de mercadorias em ordem aleatória* de Pearn e Chien, apresentado na Seção 3.2, transforma uma solução parcial arbitrária – que não atende a todas as requisições da lista de requisições – em uma solução viável.

A solução parcial que alimenta esta heurística pode ser tanto uma solução incompleta que contém alguns mercados, quanto uma solução que contém apenas o depósito. Na primeira iteração da nossa metaheurística para o TPP, utilizamos uma solução que contém apenas o depósito. Nas próximas iterações, a solução parcial depende do resultado da Fase 3.

Da mesma forma que o algoritmo de Pearn e Chien, este re-ordena a lista de requisições e considera um produto da lista por vez, inserindo mercados na solução de forma a atender a demanda do produto da maneira mais barata possível, até chegar a uma solução viável que atenda à demanda de todos os produtos.

É importante notar que o algoritmo original de Pearn e Chien foi projetado para resolver a variante não capacitada do TPP simétrico. Afim de tratar também a versão capacitada do problema, modificamos o algoritmo para que, uma vez escolhido o próximo produto, ele adicione tantos mercados quantos forem necessários para atender a demanda total do produto em questão. Esta mudança está refletita nas linhas 5 a 9 do Algoritmo 13. Observe que na resolução de instâncias do problema não capacitado este laço é repetido apenas uma vez, o que mimetiza o comportamento do algoritmo original de Pearn e Chien [42].

Outra modificação presente no nosso algoritmo é a presença de um mecanismo de diversificação baseado numa função probabilística de *ruído* na escolha do próximo mercado a ser inserido na solução. Ou seja, ao invés de inserir o mercado que

minimiza o custo total (custo de roteamento + custo de aquisição) para a compra do próximo produto – conforme o algoritmo original de Pearn e Chien – o algoritmo insere o mercado que minimiza a soma do custo total com uma função de ruído. Esta função de ruído é uma função que gera pesos de forma probabilística a fim de diversificar a solução construída. O algoritmo termina quando a solução atinge a viabilidade, mesmo que ainda seja possível chegar a uma solução melhor pela adição de mercados que ficaram fora da solução.

Algoritmo 13: H8 Fase 1: Algoritmo de Construção de uma Solução Viável

```
1 L = cria uma nova ordenação da lista de requisições;
2 v = mercado que minimiza a soma entre o custo de transporte e aquisição do
    primeiro item da lista L com uma função de ruído;
3 σ = solução parcial contendo o depósito e o mercado v;
4 para cada produto p a partir do segundo na lista L faça
5     repita
6         adiciona, na solução parcial σ, o mercado que minimiza a soma entre uma
          função de ruído com o custo de roteamento e aquisição do produto p;
7     até que a demanda total por p seja atendida ;
8 retorna a solução σ encontrada;
```

Fase 2: Melhorando o custo da solução através da VND. Nesta fase a busca local VND é utilizada para melhorar a qualidade da solução. Nossa implementação da VND oscila entre sete operadores de busca local descritos em detalhes no que segue. Quatro destes operadores: `2-opt`, `or-opt`, `i-opt` e `3-opt` foram projetados para realizar otimização no custo da solução atuando apenas na re-ordenação da sequência de mercados que estão na rota. Estes operadores não inserem novos mercados na rota nem retiram da rota os mercados previamente inseridos. Todavia, há outros três operadores projetados para esta função: `market-add`, `market-drop` e `market-exchange`. Todas os operadores de busca local implementados usam a estratégia conhecida na literatura como *best-accept*, ou seja, aceitam a melhor solução da vizinhança depois de testar todos os vizinhos da solução corrente.

O operador `2-opt` reverte uma sequência de mercados visitados pela rota. O operador `or-opt` primeiro retira da rota uma sub-sequência de mercados visitados e a re-insere na rota, numa outra posição que não a original, de forma com que os mercados sejam percorridos pelo comprador viajante na mesma ordem da rota original. O operador `i-opt` é uma generalização do `or-opt` que re-insere uma sub-sequência de mercados visitados na mesma ordem de visitas ou na ordem reversa. O operador `3-opt` é uma generalização do `i-opt` que, além de verificar a re-inserção da

sequência retirada nas duas ordenações, também verifica o ganho obtido por reverter a sequência remanescente – ou seja, aquela que permaneceu como rota parcial enquanto a outra sub-sequência fora retirada. A Figura 3.1 ilustra estas operações.

A avaliação de um movimento para estes operadores é realizado em complexidade $O(1)$ mesmo para as instâncias assimétricas, devido a estrutura de dados utilizada para armazenar a sequência de mercados visitados, nomeada de `rota` e descrita na próxima Subseção. A execução de um destes movimentos tem complexidade de tempo $O(n)$, onde n aqui é o número de mercados na rota corrente.

Estes operadores não influenciam no conjunto ótimo de aquisições de produtos – uma vez que o conjunto de mercados na solução não é modificado, o conjunto de ótimo de aquisições de produtos por estes mercados não muda – e por isso o custo computacional incorrido é apenas o que provém apenas das operações na estrutura de dados `rota`. O operador `2-opt` tem uma vizinhança de tamanho $O(n^2)$ enquanto os outros três têm tamanho $O(n^3)$. Como as avaliações dos movimentos são realizados em $O(1)$, a complexidade de se encontrar o melhor dos movimentos nestas estruturas de vizinhança corresponde ao tamanho das mesmas: $O(n^2)$ e $O(n^3)$, respectivamente

Por sua vez, o operador `market-add` adiciona um mercado à solução sempre que for vantajoso, e o `market-drop` remove um mercado, na mesma situação. O operador `market-exchange` retira um mercado da rota e insere outro fora da solução atual sempre que este conjunto de movimentos melhorar o custo da função objetivo. A complexidade de avaliação e execução de cada um destes operadores é $O(|V||P|^2)$, onde $|P|$ é o número de produtos na instância.

Fase 3: Diversificação e intensificação apoiados nos conceitos da LNS. Nesta fase aplicamos os conceitos de destruição/reconstrução da LNS a fim de gerar diversificação e intensificação no caminho da busca.

Usamos um algoritmo probabilístico que, tem $x\%$ de chance de retornar uma solução parcial vazia, $y\%$ de chance de retornar uma solução parcial baseada na retirada de mercados da solução corrente e $100-x-y\%$ de chance de retornar uma solução parcial baseada na retirada de mercados da melhor solução encontrada até o momento na busca. Realizamos experimentos computacionais preliminares a fim de decidir os valores x e y. Veja Seção 3.5.2.

O Algoritmo 15 da listagem deste procedimento. A função retira-mercados-probabilístico remove cada mercado da rota com probabilidade 80%.

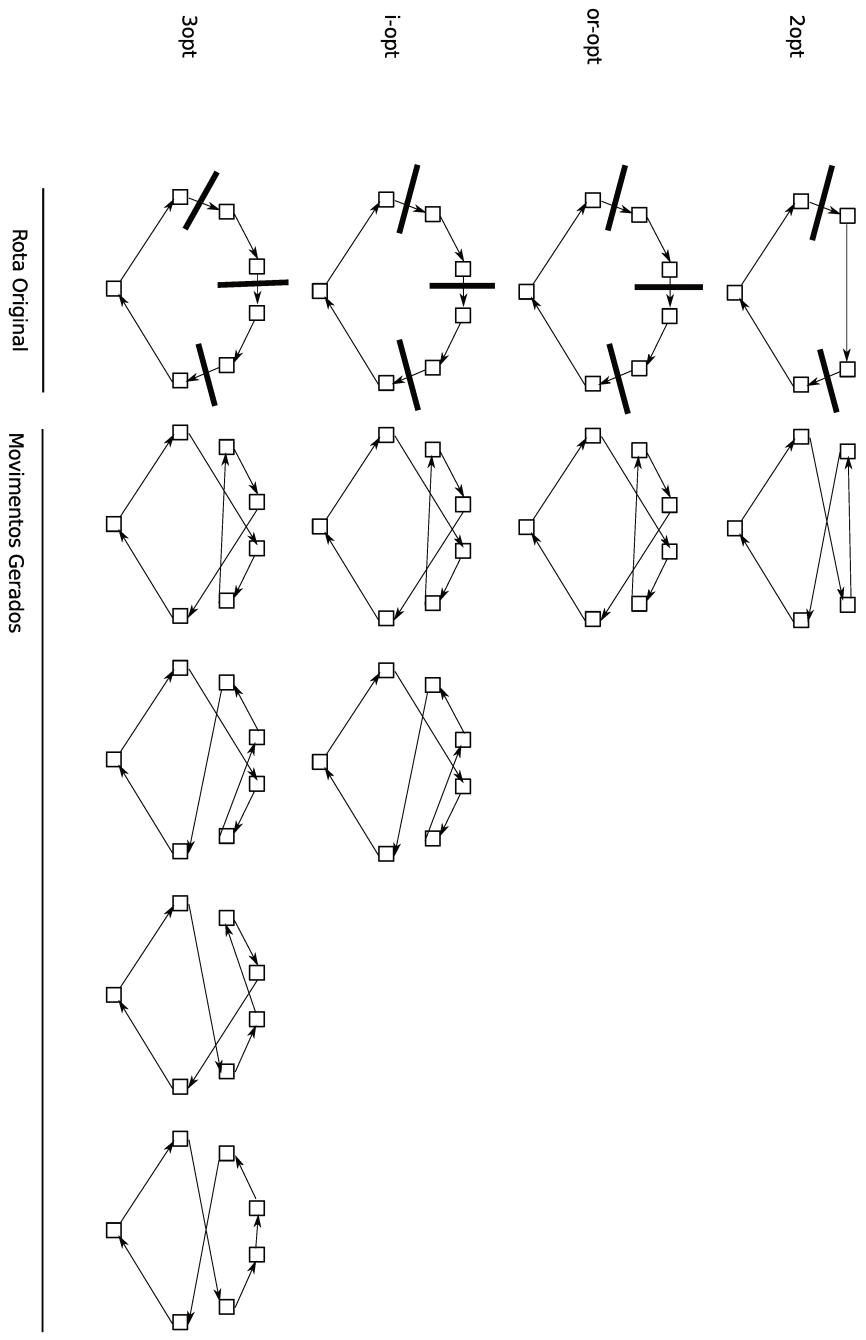


Figura 3.1: Ilustração dos operadores `2-opt`, `or-opt`, `i-opt` e `3-opt`.

Algoritmo 14: H8 Fase 2: VND

```
σ = solução viável inicial;
se market-add(σ) é melhor que σ então
    σ = market-add(σ);
    volte para o passo 2;
se market-drop(σ) é melhor que σ então
    σ = market-drop(σ);
    volte para o passo 2;
se market-exchange(σ) é melhor que σ então
    σ = market-exchange(σ);
    volte para o passo 2;
se 2-opt(σ) é melhor que σ então
    σ = 2-opt(σ);
    volte para o passo 2;
se or-opt(σ) é melhor que σ então
    σ = or-opt(σ);
    volte para o passo 2;
se i-opt(σ) é melhor que σ então
    σ = i-opt(σ);
    volte para o passo 2;
se 3-opt(σ) é melhor que σ então
    σ = 3-opt(σ);
    volte para o passo 2;
retorna a solução σ encontrada;
```

Algoritmo 15: H8 Fase 3: LNS

```
ρ = um número uniformemente distribuído de 1 a 100;
se ρ ≤ x então
    σ = solução-vazia;
senão
    se ρ ≤ x + y então
        σ = solução atual;
        σ = retira-mercados-probabilístico(σ);
    senão
        σ = melhor solução até o momento;
        σ = retira-mercados-probabilístico(σ);
retorna a solução parcial σ encontrada;
```

3.3.2 Um método especializado para o UTPP simétrico

Durante testes preliminares com a combinação das vizinhanças na Fase 2, percebemos que nosso algoritmo heurístico encontrava soluções muito boas para as instâncias simétricas não capacitadas, mesmo quando somente as vizinhanças `market-add`, `market-drop` e `market-exchange` eram usadas (não intercaladas com os operadores de rota `2-opt`, `or-opt`, `i-opt` e `3-opt`). Por este motivo criamos uma versão especializada do método resolvedor proposto para a variante simétrica não capacitada, listada no Algoritmo 16. Temos, então, uma versão que é chamada de **"versão especializada para o UTPP simétrico"** e outra versão que chamamos de **"completa"**, a qual usa todas as vizinhanças implementadas.

Algoritmo 16: H8 Fase 2 para UTPP Simétrico

```
1  σ = solução viável inicial;
2  se market-add(σ) é melhor que σ então
3  |  σ = market-add(σ);
4  |  volte para o passo 2;
5  se market-drop(σ) é melhor que σ então
6  |  σ = market-drop(σ);
7  |  volte para o passo 2;
8  se market-exchange(σ) é melhor que σ então
9  |  σ = market-exchange(σ);
10 |  volte para o passo 2;
11 retorna a solução σ encontrada;
```

3.4 Estruturas de Dados

Escrevemos nos parágrafos anteriores que o custo de uma solução pode ser avaliado a partir de duas informações: a sequência em que os mercados são visitados pelo comprador – esta informação nos permite calcular os custos de transporte/roteamento – e os custos de aquisição dos produtos em cada mercado. Nesta seção apresentamos as duas principais estruturas de dados que foram utilizadas na implementação das nossas heurísticas a fim de acelerar a avaliação dos movimentos dos operadores de busca local.

Toda vez que uma solução é modificada, seu custo deve ser re-avaliado. A avaliação eficiente da modificação no custo da solução após um movimento de um operador de busca local é um dos fatores que contribuem para o sucesso de um algoritmo heurístico.

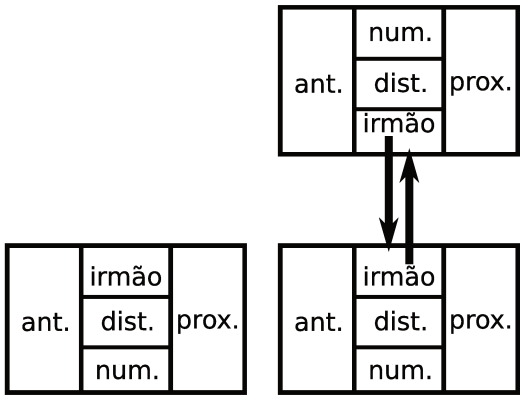
Para auxiliar o cálculo dos custos de roteamento, desenvolvemos uma estrutura de

dados chamada de `rota`. Também sugerimos uma melhoria na estrutura usada por Riera-Ledesma e Salazar-González [48] para a obtenção do custo de aquisição dos produtos de forma ainda mais eficiente. No nosso trabalho esta estrutura de dados é chamada de `aquisições`.

3.4.1 Estrutura de Dados: `rota`.

A estrutura de dados que guarda a sequência de mercados visitados deve também permitir que operações de inserção, remoção, translocação de uma sequência contígua de mercados e a reversão de uma subsequência de mercados sejam avaliadas e executada de forma eficiente. Para este fim, representamos uma rota através de duas listas duplamente ligadas de mercados. Uma destas listas representa a rota no sentido direto, enquanto a outra, a rota no sentido reverso.

Cada uma destas listas duplamente ligadas tem como elementos a estrutura da Figura 3.2(a). Este é o elemento mais básico da estrutura de dados. Dois destes, dispostos como na Figura 3.2(b) representam um mercado da rota do caixeiro comprador – um deles pertence a representação da rota no sentido direto e o outro, no reverso. O campo `irmão` é a ligação entre estes dois elementos que representam o mesmo mercado.



(a) Detalhe da representação de um Mercado

(b) A representação de um Mercado

Figura 3.2: Estrutura de dados `rota`: detalhe da representação de um mercado na rota em memória.

A rota da solução apresentada na Figura 3.3(a), é representada na memória como aparece na Figura 3.3(b). Nesta última figura os mercados da rota ligam-se com seus

vizinhos da direita e da esquerda pelos apontadores `ant.` e `prox.` respectivamente. Note que o depósito (com o número 1) aparece duplicado no início e no final da representação.

O campo `dist.` guarda a distância acumulada do início da rota até o mercado em questão, se ele está na sequência que representa a rota direta, ou, a distância acumulada do final da rota até o mercado, se ele representa a rota reversa. Estes dados de distância possibilitam avaliar uma translocação ou uma reversão de uma parte de caminho em complexidade de tempo $O(1)$ mesmo nas instâncias com custos de viagem assimétricos. O caso da reversão está representado de forma simplificada na Figura 3.3(c).

Dados os apontadores para os mercados envolvidos, é possível implementar as operações de avaliação de inserção, remoção, translocação e reversão todas em complexidade $O(1)$. A execução propriamente dita destes movimentos é feita em $O(c)$ onde c é o número de mercados presentes da rota.

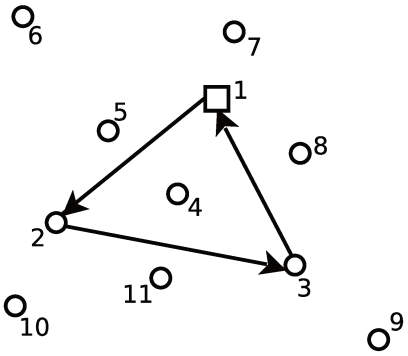
3.4.2 Estrutura de Dados: `aquisições`.

O projeto desta estrutura de dados está apoiado no trabalho de Riera-Ledesma [48]. Esta estrutura consiste num vetor de listas ligadas de ofertas de produtos. Uma oferta de um produto p num mercado v possui quatro campos com informações estáticas (lidas da instância) nos quais armazenam-se os identificadores do mercado v e do produto p a que se refere, o preço unitário deste produto neste mercado e a quantidade ofertada do produto no mercado (os campos `merc.`, `prod.`, `preço` e `q.o.` na Figura 3.4(a), respectivamente). Há ainda três campos que são utilizados para a representação das aquisições na estrutura de dados: o campo `q.u.` salva a quantidade desta oferta adquirida pelo comprador viajante e os campos `ant.` e `prox.` apontam da oferta atual para outras duas ofertas da lista.

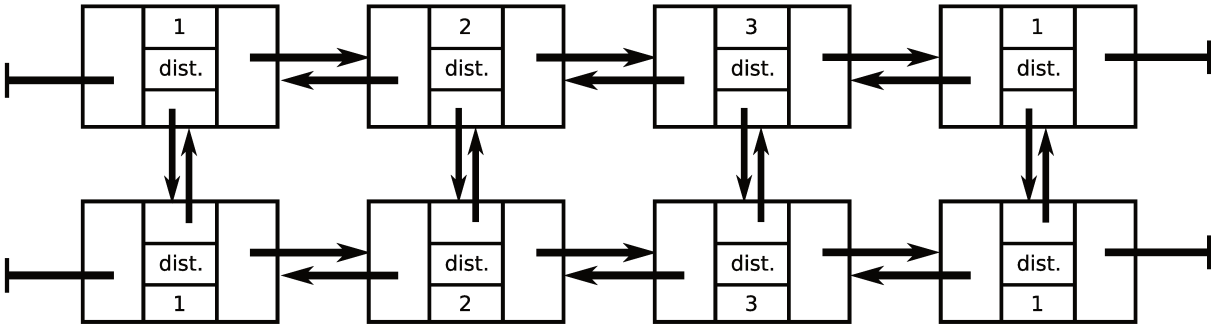
As ofertas de produtos de uma solução parcial σ tem uma representação interna que consiste numa lista duplamente ligada para cada produto $p_i \in P$. Cada componente destes vetores corresponde a uma oferta do produto num mercado v_i na solução σ. Estas ofertas são pré-ordenadas de acordo com o custo unitário de aquisição, conforme mostra a Figura 3.4(b).

Como armazenamos em `aquisições` todas os produtos que são ofertados pelos mercados visitados em ordem crescente de preço unitário, avaliar o custo de aquisição dos produtos de uma solução se torna simples: basta percorrer a lista de ofertas de cada produto somando os custos dos itens adquiridos, até que a demanda do produto na lista de requisições seja completamente atendida.

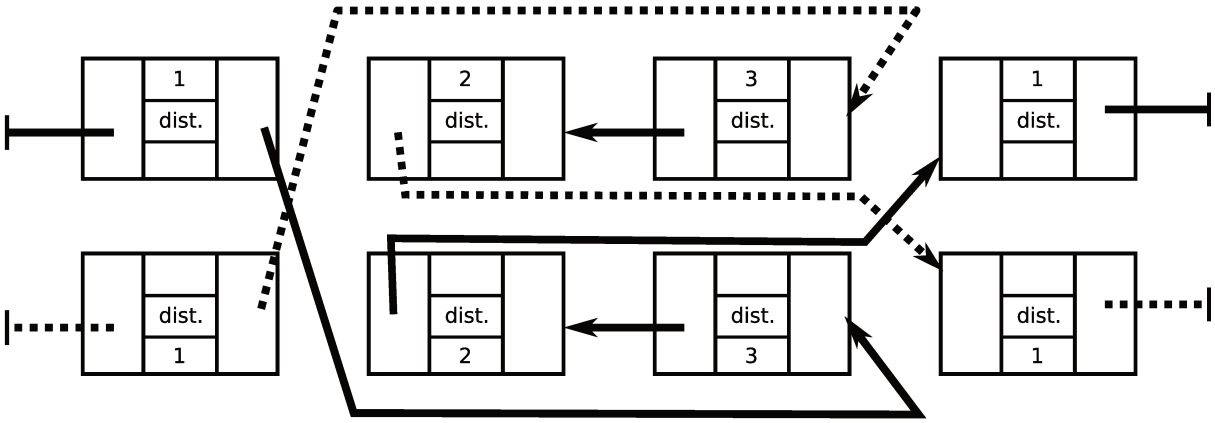
A inserção de um mercado na solução implica em inserir cada uma de suas ofertas dentro do vetor dinâmico, o que é executado em $O(|V||P|)$ onde $|P|$ é o número de produtos da instância e $|V|$ o número de mercados, visto que para cada uma das $O(|P|)$ ofertas do mercado adicionado a solução deve ser inserido numa lista ordenada indexada



(a) Exemplo de uma rota típica



(b) A representação da rota na estrutura de dados



(c) Uma reversão dos mercados na estrutura de dados

Figura 3.3: Estrutura de dados `rota`: representação em memória.

pelo produto da oferta, que tem tamanho $O(|V|)$.

Para a avaliação da inserção de uma oferta para um determinado produto na estrutura de aquisições – para o caso das variantes capacitadas – necessitamos percorrer a lista ordenada – por custo unitário – de ofertas para o produto até que a soma das unidades ofertadas seja maior que a quantidade damandada – e neste caso a função custo não muda o valor pois a oferta inserida não foi utilizada – ou, que a posição de inserção da nova oferta seja encontrada enquanto a soma das unidades ofertadas seja menor que a demanda. Então, a partir desta posição precisamos verificar a diferença na função de custo. Contudo, há uma oportunidade de ganho computacional na avaliação do custo na inserção de uma oferta dentro da lista de ofertas para o produto: guardando-se a quantidade total comprada de cada produto e um ponteiro para a última oferta da lista de cada produto na qual a quantidade usada é maior que zero podemos acelerar tanto a avaliação quanto a execução das operações de inserção e remoção de ofertas na estrutura de dados; mantendo, porém, a mesma complexidade assintótica.

A vantagem de avaliar a lista de ofertas a partir da última oferta utilizada é que avaliamos apenas a parte absolutamente necessária da lista ao invés de avaliarmos a lista desde o início, toda vez que a usamos.

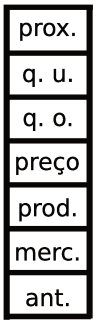
3.5 Experimentos Computacionais

Esta seção descreve os experimentos computacionais realizados a fim de testar os métodos heurísticos propostos na seção anterior. Os experimentos com as instâncias Simétricas com Restrições de Capacidade foram executados em uma máquina Core2 Quad 2.4GHz com 8Gb de RAM. Os demais experimentos foram executados num notebook Core2 Duo 2.0GHz com 4Gb de RAM.

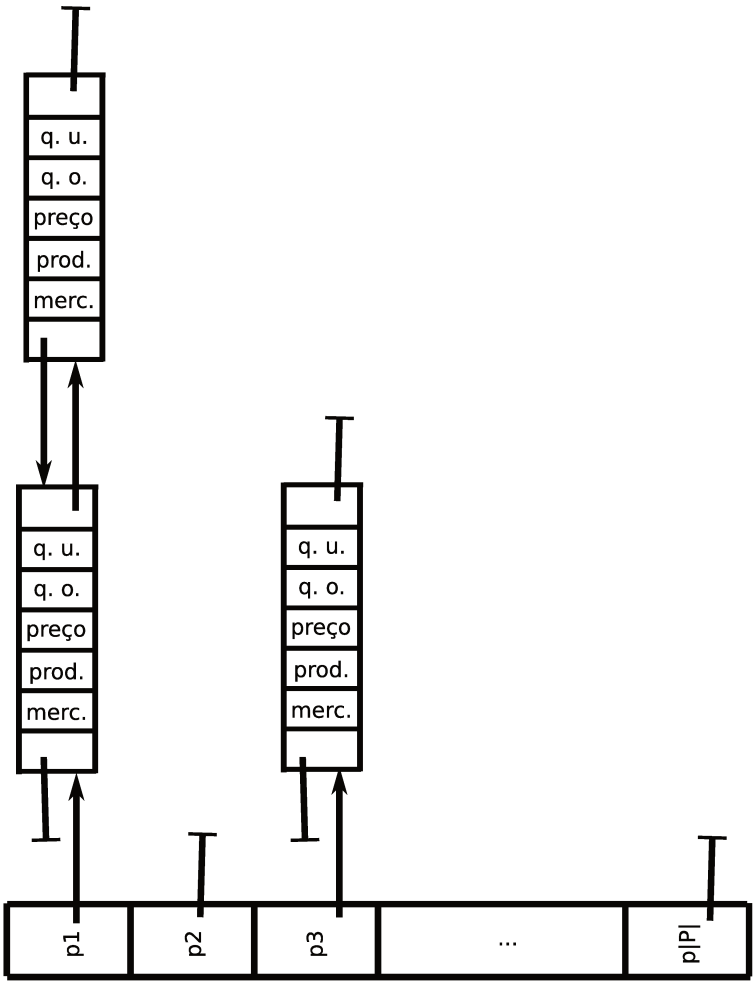
3.5.1 Descrição das Instâncias utilizadas

As instâncias usadas para testar o desempenho do nosso código foram trazidas da literatura e estão disponíveis publicamente no site http://webpages.ull.es/users/jriera/TPP.htm de autoria de Riera-Ledesma.

A descrição das instâncias simétricas é dada no trabalho de Boctor e colegas [7] e segue. A fim de se construrir cada uma destas instâncias primeiro gerou-se vértices com coordenadas inteiras no quadrado $[0, 1000] \times [0, 1000]$ de acordo com uma distribuição uniforme e definindo os custos de roteamento usando-se as distâncias euclidianas. Cada produto p foi associado com $|M_p|$ mercados aleatoriamente selecionados, onde $|M_p|$ foi aleatoriamente gerado no intervalo $[1, m]$. Os preços dos produtos foi gerado no intervalo de $[1, 500]$ de acordo com uma distribuição uniforme discreta. Nestas instâncias tem-se



(a) Detalhe da representação de uma Oferta



(b) A representação das aquisições

Figura 3.4: Estrutura de dados `aquisições`: representação em memória.

$m \in \{50, 100, 150, 200, 250, 300, 350\}$ mercados e $n \in \{50, 100, 150, 200\}$ produtos. Cinco instâncias foram geradas para cada combinação de m e n. Entretanto, o valor ótimo é conhecido apenas para 89 destas 140 instâncias; nos 51 casos restantes o melhor resultado heurístico é usado como índice de comparação.

De forma similar, nas instâncias do problema capacitado tem-se $m \in \{50, 100, 150, 200\} \cup \{250, 300, 350\}$ mercados e $n \in \{50, 100, 150, 200\}$ produtos, com sete valores para um parâmetro λ que controla a demanda de um produto p. Nestas instâncias, q_{pi} – a quantidade do produto p ofertada pelo mercado i – é aleatoriamente gerado dentro do intervalo $[1, 15]$ e $d_p = \lambda \text{Max}_{i \in M_p} q_{pi} + (1 - \lambda) \sum_{i \in M_p} q_{pi}$ onde $\lambda \in \{0.1, 0.5, 0.7, 0.8, 0.9, 0.95, 0.99\}$ é a demanda do produto p. Novamente cinco instâncias são geradas para cada combinação de m, n e λ resultando em 980 instância, 439 delas com a solução ótima conhecida.

As instâncias assimétricas descritas no trabalho de Riera Ledesma [49] foram construídas da seguinte forma: os custos de roteamento c_{ij} foram gerados no intervalo $[1, \tau]$ onde τ é gerado no intervalo $[15, 140]$. Cada mercado vende um número de produtos aleatoriamente gerados no intervalo $[1, n]$ no qual n é o número de produtos. Os custos de aquisição são gerados aleatoriamente no intervalo $[0, \omega]$, onde ω é gerado dentro do intervalo $[5, 75]$ para cada mercado. Estas instâncias tem $m \in \{50, 100, 150, 200, 250\} \cup \{300, 350\}$ mercados e $n \in \{50, 100, 150, 200\}$ produtos. Definiram-se cinco instâncias para cada combinação de m e n, considerando-se cinco diferentes sementes.

Para construir as instâncias assimétricas com restrições de capacidades o gerador anterior foi estendido da seguinte maneira: Para cada produto p e cada mercado i, a quantidade em oferta q_{pi} a quantidade do produto p ofertada pelo mercado i – q_{pi} – é aleatoriamente gerado dentro do intervalo $[1, 15]$ e $d_p = \lambda \text{Max}_{i \in M_p} q_{pi} + (1 - \lambda) \sum_{i \in M_p} q_{pi}$ onde $\lambda \in \{0.1, 0.5, 0.7, 0.8, 0.9, 0.95, 0.99\}$, de forma similar as instâncias simétricas.

3.5.2 Experimentos Preliminares

Realizamos uma série de experimentos preliminares a fim de determinar: (a) uma boa composição de operadores para segunda fase do algoritmo; e, (b) um bom conjunto de valores dos parâmetros x e y usados na fase 3 do algoritmo. Em cada Subseção que segue explicitamos a motivação dos testes e apresentamos os resultados obtidos.

Importância Comparativa dos Operadores de Busca Local na Fase 2

A fim de chegar a uma boa combinação de operadores de busca local na segunda fase do algoritmo, implementamos diversas variações do nosso procedimento VNS. Cada variação considerou um subconjunto dos operadores de busca local.

Os resultados obtidos pelas nossas implementações são apresentados na Tabela 3.1. A linha (a) desta tabela corresponde ao método VND que chamamos de `VND FULL`, pois exe-

	Método	T. (s)	50.1	50.7	100.8	100.95
(a)	VND FULL	7.43	461048	135232	166370	41582
(b)	VND - market-add	7.74	461050	135234	166373	41584
(c)	VND - market-drop	4.06	461052	135235	166373	41585
(d)	VND - market-exchange	6.62	461050	135237	166373	41590
(e)	VND - 2-opt	6.49	461050	135237	166374	41591
(f)	VND - or-opt	6.61	461053	135237	166371	41591
(g)	VND - i-opt	7.54	461053	135238	166372	41593
(h)	VND - 3-opt	8.25	461054	135238	166374	41594
(i)	Sem VND	0.14	462981	144393	178944	43397
(j)	Só construção	0.04	463835	151168	182141	43576

Tabela 3.1: Comparação da qualidade da solução e tempo de execução produzido pelo esquema VND completo com abordagens que omitem um dos seus operadores e com resultados de um de seus operadores

cuta todos os sete operadores de busca local implementados neste trabalho. As instâncias escolhidas para o teste foram as primeiras instâncias de 100 mercados do conjunto de assimétricas com restrições de capacidade. A identificação $|P|.\lambda$ no cabeçalho de cada coluna da tabela referem-se aos parâmetros número de produtos e o λ da instância, respectivamente.

Os métodos representados nas linhas de (a) até (h) são modificações do método `VND FULL` nas quais um dos operadores foi retirado. Por exemplo, na linha (d), são listados os resultados de tempo acumulado e da qualidade de solução para cada uma das instâncias testadas com o procedimento VND que executa 6 operadores de busca local – market-add, market-drop, 2-opt, or-opt, i-opt e 3-opt – e não usa o operador market-exchange. Na linha (i) nenhum operador de busca local é usado – ou seja, apenas as fases 1 e 3 do algoritmo são executadas. Na linha (j) apenas o algoritmo de construção é executado.

Os resultados indicam que a combinação `VND FULL` não é dominada pelas outras variações e seus resultados sempre estão entre os melhores obtidos, mesmo que a um custo computacional mais caro. Por este motivo, escolhemos esta combinação como combinação final para o nosso algoritmo.

Determinação dos Parâmetros da Terceira Fase

A terceira fase do nosso algoritmo tem dois parâmetros que devemos ajustar. A fim de determinar os parâmetros x e y da Fase 3 do algoritmo rodamos 10 iterações para cada um dos parâmetros sobre cinco instâncias. Novamente as instâncias escolhidas para o teste

(x, y)	T. (s)	50.5	100.1	150.7	150.95	200.99
$(10, 30)$	9.71	234773	871632	391865	63744	18889
$(15, 20)$	9.64	234773	871632	391865	63744	18889
$(50, 40)$	9.92	234773	871632	391866	63745	18889

Tabela 3.2: Comparação da qualidade da solução e tempo de execução produzido pelo método heurístico de acordo com conjuntos distintos de parâmetros para a fase 3

foram as primeiras instâncias de 100 mercados do conjunto de assimétricas com restrições de capacidade. Os resultados encontrados estão na Tabela 3.2. (x, y) são os parâmetros que estamos ajustando. Os valores testados para x e y foram 10 e 30, 15 e 20 e 50 e 40. A segunda coluna mostra o tempo em segundos utilizado pelos testes para cada um dos conjuntos de parâmetros. A identificação $|P|.\lambda$ no cabeçalho das colunas restantes da tabela refere-se aos parâmetros número de produtos e o λ da instância, respectivamente.

Verificamos que o algoritmo não é sensível a escolha destes parâmetros, e escolhemos a configuração $(15, 20)$, que obteve o menor tempo de execução para encontrar a melhor solução para estas instâncias – para nossa configuração final da heurística.

3.5.3 Análise Comparativa dos Resultados Obtidos Sobre as Instâncias da Literatura

Nesta seção reportamos e comparamos os resultados obtidos sobre as instâncias da literatura para as diferentes variantes do TPP.

Resultados nas instâncias simétricas não capacitadas da TPPlib

Executamos o nosso método heurístico especializado para as instâncias simétricas não capacitadas – aquele sem as vizinhanças `2-opt`, `or-opt`, `i-opt` e `3-opt` – por 10 vezes, com sementes distintas para o gerador de números pseudo-aleatórios, sobre cada uma das instâncias desta classe. Comparamos os resultados obtidos pela heurística especializada, a partir de agora chamada de H8-sutpp, produzida neste trabalho com os resultados dos três métodos mais recentes da literatura. O método RL-SG é a heurística de Riera-Ledesma e Salazar-González [48], executada num PC Celeron 500MHz. DMD-ATA é o resolvedor de Bontoux e Feillet [8], que fora executado num PC Pentium 4 2GHz. TA é o algoritmo transgenético de Goldbarg e colegas [22] executado numa máquina Pentium 4 2,8GHz. Os resultados foram consolidados nas Tabelas 3.3 e 3.4.

A Tabela 3.3 agrupa apenas os resultados nas instâncias cuja solução ótima é conhecida. Os resultados detalhados nestas instâncias podem ser vistos nas tabelas A.1 e A.2.

Método		m					n			
		50	100	150	200	250	50	100	150	200
RL-SG	%gap	0.07	0.14	0.03	0.32	0.06	0.07	0.24	0.10	0.08
	T(s)	3	10	14	19	25	5	13	20	21
DMD-ATA	%gap	0	0	0.08	0.02	0.01	0	0.05	0	0.03
	T(s)	2	20	172	232	154	37	154	96	165
TA	%gap	0	0	0.03	0	0	0	0	0	0.03
	T(s)	0.83	1.42	2.15	2.54	2.60	0.95	1.58	2.22	2.74
H8-sutpp	**%gap**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	T(s)	**0.07**	**0.24**	**0.48**	**0.83**	**0.74**	**0.43**	**0.86**	**1.45**	**1.62**

Tabela 3.3: Performance do Algoritmo H8-sutpp (deste trabalho) em comparação com a performance dos outros algoritmos da literatura.

Conforme os resultados da Tabela 3.3 verificamos que o algoritmo RL-SG obteve gap de 0.07% em média, nas instâncias de 50 mercados e de 50 a 250 produtos; 0.14 nas de 100 mercados; 0.03% , 0.32% e 0.06% de gap médio nas instâncias de 150, 200 e 250 mercados. Já nas instâncias de 50 produtos esta heurística obteve um gap de 0.07%. Obteve também gaps de 0.24%, 0.1% e 0.08% em média nas instâncias de 100, 150 e 200 produtos.

Os resultados obtidos pela H8-sutpp nestas instâncias melhoram os resultados do último algoritmo publicado na literatura (TA): as melhores soluções obtidas pelo TA não são melhores que as obtidas pelo H8-sutpp, enquanto o tempo computacional para obtê-las é reduzido utilizando-se máquinas similares.

Os resultados para as instâncias em que uma solução ótima não é conhecida estão descritos na Tabela 3.4. Nesta tabela listamos os melhores valores alcançados pela literatura e os valores das soluções e o tempo médio para alcançar estes valores pelos métodos heurísticos, quando estes valores estão disponíveis.

Notamos que o método H8-sutpp encontra soluções nunca piores que os outros métodos da literatura, além de baixar o custo da melhor solução para a primeira instância com 350 mercados, 150 produtos – 350.150.1 – de 1457 para 1455.

Os tempos computacionais indicam que o procedimento implementado H8-sutpp escala melhor com o tamanho das instâncias do que os algoritmos concorrentes. A vantagem sobre os outros métodos é notável.

O método H8-sutpp é o primeiro método, até onde sabemos, que consegue obter todas as melhores soluções conhecidas das instâncias desta classe.

			Lit.	RL-SG	DMT-ATA		TA		H8-sutpp	
M	P	I	Best	Sol.	Sol.	T. (s)	Sol.	T. (s)	Sol.	T. (s)
200	150	4	2419	2426	**2419**	1216,92	**2419**	23,98	**2419**	1,46
200	200	4	2344	2353	**2344**	527,03	**2344**	99,19	**2344**	1,26
250	100	1	1301	1309	**1301**	33,84	**1301**	143,19	**1301**	0,45
250	100	4	1673	1677	**1673**	10,23	**1673**	3,55	**1673**	0,33
250	100	5	1641	1648	**1641**	550,24	**1641**	1,84	**1641**	0,16
250	150	4	1836	1840	**1836**	45,24	**1836**	2,27	**1836**	0,59
250	150	5	1531	1539	**1531**	21,10	**1531**	151,43	**1531**	0,54
250	200	2	2785	2787	**2785**	1137,65	2786	246,31	**2785**	1,97
250	200	3	1924	1934	**1924**	281,88	**1924**	16,45	**1924**	0,68
250	200	4	2116	2128	**2116**	83,83	**2116**	3,06	**2116**	0,86
250	200	5	1797	1807	**1797**	930,03	**1797**	38,97	**1797**	0,19
300	50	1	1477	1483	**1477**	160,00	**1477**	1,50	**1477**	0,10
300	50	2	813	816	**813**	116,01	**813**	1,41	**813**	0,06
300	50	3	1117	1123	**1117**	20,00	**1117**	1,46	**1117**	0,05
300	50	4	1176	1183	**1176**	2,11	**1176**	1,44	**1176**	0,08
300	50	5	1256	1262	1257	276,00	**1256**	1,57	**1256**	0,06
300	100	1	1035	1040	**1035**	55,54	**1035**	2,29	**1035**	0,16
300	100	2	1179	1184	**1179**	617,22	1180	3,98	**1179**	0,20
300	100	3	1498	1507	**1498**	103,42	**1498**	2,25	**1498**	0,20
300	100	4	1749	1759	**1749**	312,16	**1749**	37,49	**1749**	0,34
300	100	5	1774	1781	**1774**	2,74	**1774**	2,27	**1774**	0,12
300	150	1	1457	1459	**1457**	756,71	**1457**	98,66	**1457**	0,30
300	150	2	1656	1667	**1656**	483,32	**1656**	3,02	**1656**	0,48
300	150	3	2484	2492	2485	663,24	**2484**	6,34	**2484**	4,01
300	150	4	1801	1809	**1801**	95,93	**1801**	8,17	**1801**	3,53
300	150	5	1816	1825	**1816**	309,25	**1816**	41,16	**1816**	0,64
300	200	1	1803	1812	1815	488,15	**1803**	575,39	**1803**	3,71
300	200	2	1790	1798	1791	1918,52	**1790**	627,73	**1790**	2,02
300	200	3	2437	2450	2442	2852,05	**2437**	184,00	**2437**	2,49
300	200	4	1815	1829	**1815**	2946,79	**1815**	113,82	**1815**	0,64
300	200	5	2014	2022	2022	1577,83	**2014**	605,39	**2014**	1,31
350	50	1	723	727	**723**	46,04	**723**	1,70	**723**	0,08
350	50	2	736	739	**736**	25,71	**736**	13,02	**736**	0,08
350	50	3	942	946	**942**	6,00	**942**	1,82	**942**	0,22
350	50	4	805	809	**805**	379,39	**805**	5,01	**805**	0,03
350	50	5	1225	1230	**1225**	26,35	**1225**	1,67	**1225**	0,13
350	100	1	1317	1321	**1317**	1698,48	**1317**	229,99	**1317**	0,96
350	100	2	962	965	**962**	155,48	**962**	2,37	**962**	0,12
350	100	3	796	804	**796**	839,65	**796**	2,43	**796**	0,15
350	100	4	1059	1065	**1059**	13,94	**1059**	9,14	**1059**	0,06
350	100	5	1566	1576	**1566**	464,86	**566**	41,76	**1566**	0,58
350	150	1	*1*457	1459	1457	1986,42	1459	319,67	**1455**	0,88
350	150	2	1315	1322	**1315**	159,12	**1315**	16,31	**1315**	0,39
350	150	3	2553	2566	**2553**	257,69	2558	597,74	**2553**	4,32
350	150	4	1239	1248	**1239**	595,85	**1239**	3,06	**1239**	0,13
350	150	5	2288	2297	**2288**	8,93	**2288**	229,27	**2288**	1,85
350	200	1	1498	1505	1503	1033,39	**1498**	25,34	**1498**	1,16
350	200	2	1369	1377	1374	3085,09	**1369**	56,07	**1369**	0,67
350	200	3	1873	1889	**1873**	368,66	**1873**	59,05	**1873**	3,24
350	200	4	1356	1361	1385	122,24	**1356**	32,88	**1356**	0,34
350	200	5	2336	2347	2336	2385,65	**2336**	204,53	**2336**	2,40

Tabela 3.4: Resultados para as Instâncias Simétricas não Capacitadas Para as Quais a Solução Ótima Ainda não foi Provada

Resultados nas instâncias simétricas capacitadas da TPPlib

Testamos nossa heurística que considera todas as vizinhanças implementadas durante a Fase 2, a partir de agora chamada apenas de H8. e obtivemos os resultados que se encontram na Tabela 3.5.

Comparamos a heurística deste trabalho, com as heurísticas RL-SG de Riera-Ledesma e Salazar-González [48], executada num PC Celeron 500MHz e o método TA, que é o algoritmo transgenético de Goldbarg e colegas [22] executado numa máquina Pentium 4 2,8GHz. A heurística DMT-ATA que foi comparado na seção anterior não possui resultados para instâncias capacitadas e por isso não aparece na comparação desta seção.

Na Tabela 3.5 os resultados obtidos foram agrupados de três formas distintas: pelo número de mercados da instância, pelo número de produtos, e pelo parâmetro lambda, e representam os três grupos de linhas da Tabela. Este modo de agrupar os resultados veio da literatura.

A primeira linha abaixo do rótulo "Mercados", por exemplo, deve ser lida assim: os resultados obtido pelos métodos H8, RL-SG e TA sobre as instâncias da classe simétrica capacitada com 50 mercados, foram 0.46%, 0.15% e 0% piores em comparação com os resultados ótimos, respectivamente, com tempos de execução de 0.22, 5 e 0 segundos.

Notamos que o método TA bate com alguma os métodos RL-SG e H8 em termos de qualidade de solução e escala de forma similar ao H8 em termos de tempo de execução, quando comparamos as duas primeiras seções da tabela – Mercados e Produtos.

Já a seção que agrupa as instâncias pelo parâmetro lâmbda utilizado na criação das instâncias nos revelou um comportamento interessante do método H8: o tempo de execução da nossa heurística decresce com o aumento do lâmbda já apartir do valor 0.5, tornando-se inferior aos temos de execução da TA a partir do lâmbda 8.

Dado que o parâmetro lambda arbitra a demanda total de cada produto na lista de requisições e, visto que nestas instâncias, q_{pi} – a quantidade do produto p ofertada pelo mercado i – é aleatoriamente gerado dentro do intervalo $[1, 15]$ e $d_p = \lambda \text{Max}_{i \in M_p} q_{pi} + (1 - \lambda) \sum_{i \in M_p} q_{pi}$ onde $\lambda \in \{0.1, 0.5, 0.7, 0.8, 0.9, 0.95, 0.99\}$ é a demanda do produto p, podemos verificar que um lâmbda pequeno obriga o caixeiro comprador viajante a visitar mais mercados para conseguir suprir a demanda de um produto, enquanto que um lâmbda grande não o obriga a fazer isto. Sendo assim, as instâncias com lâmbda grande são as mais parecidas com as instâncias não capacitadas, nas quais o nosso algoritmo provou ser bom.

	H8		RL-SG		TA	
	%gap	T (s)	%gap	T (s)	%gap	T (s)
Mercados:						
50	0.46	0.22	0.15	5	0.00	0
100	0.84	3.41	0.50	26	0.02	4
150	1.89	17.16	0.43	52	0.09	14
200	2.52	44.93	1.15	100	0.48	51
Produtos:						
50	1.75	15.05	0.72	43	0.11	15
100	1.37	16.52	0.56	38	0.15	14
150	0.86	7.45	0.42	32	0.03	6
200	0.89	8.94	0.29	32	0.03	5
Lambdas:						
0.1	0.04	24.75	0.00	8	0.00	3
0.5	0.16	25.27	0.20	25	0.15	8
0.7	0.73	19.71	0.10	29	0.01	17
0.8	1.45	13.50	0.57	62	0.09	25
0.9	3.66	3.71	1.43	86	0.26	12
0.95	2.49	1.31	0.68	26	0.05	3
0.99	0.32	0.27	0.31	19	0.03	3

Tabela 3.5: Comparação da qualidade da solução e tempo de execução produzido pelo método heurístico H8 nas instâncias da Classe Simétrico Capacitado

	H8	
	Valor	T (s)
Mercados:		
50	2849.00	0.06
100	2814.00	0.15
150	2802.00	0.32
200	2796.00	0.59
250	2793.00	0.81
300	2783.00	1.22
Produtos:		
50	1179.00	0.06
100	2279.00	0.23
150	3351.00	0.62
200	4416.00	1.20

Tabela 3.6: Qualidade da solução e tempo de execução produzido pelo método heurístico H8 nas instâncias da Classe Assimétrico Não Capacitado

Resultados nas instâncias assimétricas capacitadas e não capacitadas da TP-Plib

Testamos também o método heurístico H8 nas instâncias assimétricas. Não temos os resultados ótimos disponíveis para esta classe e somos os primeiros a publicar os resultados heurísticos para este conjunto de instâncias. As Tabelas 3.6 e 3.7 consolidam estes resultados. A coluna Valor mostra o valor total acumulado das soluções obtidas para o subconjunto indicado pela linha enquanto a coluna T(s) indica a média de tempo em segundos para alcançar a melhor solução.

3.6 Conclusões e Trabalhos Futuros

Apresentamos neste Capítulo duas heurísticas eficientes – derivadas das metaheurísticas LNS e VNS – para as variantes do Problema do Caixeiro Comprador Viajante encontradas na literatura. Além disso, descrevemos uma nova estrutura de dados para o armazenamento de uma sequência de mercados que possibilita a implementação da verificação do custo de reversão de parte da sequência em tempo constante, e uma modificação na estrutura de dados de aquisições para a avaliação eficiente da diferença de custo gerada pela inserção/remoção de um mercado na solução.

O método heurístico proposto usa uma nova abordagem iterativa em três fases. Na

	H8	
	Valor	T (s)
Mercados:		
50	169923.00	0.19
100	335574.00	2.41
150	506963.00	9.43
200	670830.00	11.72
Produtos:		
50	173362.00	5.22
100	337474.00	5.07
150	500993.00	6.49
200	671463.00	6.98
Lambdas:		
1	1376692.00	6.71
5	698576.00	5.90
7	401086.00	7.52
8	261429.00	8.03
9	128884.00	8.70
95	64887.00	3.94
99	14206.00	0.77

Tabela 3.7: Qualidade da solução e tempo de execução produzido pelo método heurístico H8 nas instâncias da Classe Assimétrico Capacitado

primeira fase uma solução viável é criada. Então, a solução criada é melhorada através de uma heurística VND com sete operadores de busca local. Por fim, uma estratégia de diversificação e intensificação baseada nos conceitos da LNS é aplicada a fim de realimentar a primeira fase, até que o número máximo de iterações seja atingido.

Experimentos preliminares demonstraram que o uso das três fases propostas – (a) construção gulosa probabilística, (b) Busca Local com VND e, (c) a diversificação e intensificação através de LNS – é vantajosa quando comparada ao uso das fases (a) e (b) ou (a) e (c) somente. Outros experimentos mostraram que o uso de todas as sete estruturas de vizinhança descritas, dentro da estratégia VND, é mais interessante que o uso de apenas uma estrutura isoladamente quando consideramos as versão capacitadas ou as assimétricas.

Uma análise comparativa com outras heurísticas da literatura foi realizada com base nos experimentos computacionais dos métodos propostos, executados sobre instâncias da literatura. A metaheurística proposta neste trabalho alcança os melhores resultados conhecidos para as 140 instâncias da variante simétrica não capacitada deste conjunto, e é competitiva na variante simétrica capacitada. Até onde chega nosso conhecimento, apresentamos os primeiros resultados de heurísticas da literatura para as instâncias assimétricas – tanto para as instâncias da variante capacitada quanto para a não capacitada.

Trabalhos futuros vão explorar o uso de esquemas mais sofisticados para a criação de soluções viáveis na primeira fase, além de uma implementação para computadores paralelos. Atualmente, a etapa que consome a maior parte do tempo computacional é a fase 2 – que poderiam tomar vantagem de uma implementação paralela. A análise do impacto da inserção destes métodos heurísticos como fornecedores de limitantes primais em métodos exatos de programação inteira parece promissora.

Capítulo 4

O Problema dos m Caixeiros Compradores Viajantes

Neste capítulo nós apresentamos um método heurístico para resolver o CmTPP. Este problema tem muitas aplicações práticas, tanto no planejamento otimizado de rotas de transporte – que é o caso da aplicação no transporte fretado de funcionários de uma empresa, abordado diretamente neste trabalho – quanto no de redes de comunicação.

O CmTPP consiste em encontrar m rotas otimizadas que são usadas por m caixeiros compradores viajantes a fim de comprar produtos – na quantidade exata – da forma como descrito numa lista de requisições. O objetivo final é minimizar o total de custos de roteamento e de aquisição dos produtos. E mais, há uma capacidade máxima de carga dos compradores que deve ser respeitada.

Este problema pode ser visto como uma generalização de outros da literatura, como é o caso do Problema do Caixeiro Viajante e, por este motivo, é também NP-difícil; e apesar de já ter sido citado em outros trabalhos, ele é proposto formalmente aqui, pela primeira vez, pelo menos até onde está o nosso conhecimento.

Na heurística que propusemos, depois da fase de construção há uma fase de melhoramento que segue o arcabouço VNS, em que uma série de procedimentos de busca local é aplicado iterativamente um após o outro. Nossa heurística também incorpora aspectos do LNS, no qual há uma fase de relaxamento das restrições da solução com a posterior reconstrução de uma solução viável.

Experimentos computacionais sobre as instâncias de *benchmark* da literatura para o CmRSP – um problema que é um sub-caso do CmTPP – indicam que a heurística proposta consegue obter resultados compatíveis com os resultados dos métodos do estado da arte para este problema – apesar de esta classe de instâncias corresponder apenas a um caso restrito do CmTPP. Um ponto de vista interessante na análise destes resultados é o da heurística não explorar vantagem computacional alguma por tratá-lo como um CmTPP

ao invés de atacar diretamente o problema original, como fazem os métodos da literatura com que comparamos os resultados obtidos neste trabalho.

4.1 Introdução

O Problema dos m Caixeiros Compradores Viajantes com Restrições de Capacidade (CmTPP) é introduzido neste trabalho. No CmTPP tem-se um grafo $G = (V, E)$, no qual V é o conjunto de mercados (os vértices do grafo) e $E = (i, j) : i, j \in V, i \neq j$ é o conjunto de arestas ou arcos direcionados.

O conjunto de vértices V é definido como $V = \{0\} \cup M$ em que o vértice 0 representa o depósito e M o conjunto de Mercados. Consideramos um custo não negativo de transporte c_e para cada aresta $e \in E$ que liga o depósito a um mercado ou um mercado a outro mercado ou ao depósito. Além destes dois conjuntos temos o conjunto P dos produtos ofertados pelos mercados em V e, uma lista de requisições R que descreve a demanda requisitada d_p de cada produto $p \in P$. Cada mercado $m \in M$ oferece uma quantidade q_{mp} do produto p a um custo unitário não negativo de aquisição a_{mp}.

Uma solução do problema é composta por m rotas mercado-disjuntas que saem do depósito e visitam um subconjunto dos mercados a serem percorridas por diferentes compradores viajantes. Em cada mercado de uma rota deve ser indicada a quantidade adquirida de cada produto pelo comprador correspondente. A viabilidade da solução fica garantida se o total adquirido de cada produto p pelos m compradores for superior ou igual à quantidade demandada deste mesmo produto (d_p), conforme descrito na lista de requisições R. Além disso, as capacidades de carga de cada comprador deve ser respeitada. O objetivo é minimizar a soma dos custos de transporte e de aquisição dos produtos.

É importante enfatizar aqui que cada um dos compradores não possui de antemão a lista da quantidade que deve comprar. Eles devem trabalhar de forma "coordenada" a fim de tomar uma decisão em dois níveis: (a) decidir a composição de cada lista de requisições individual e (b) a rota que cada um vai seguir de maneira integrada, a fim de que o resultado global atingido seja otimizado.

O CmTPP é NP-difícil pois generaliza o Problema do Caixeiro Viajante (TSP). Nesse caso particular, $m = 1$ (temos apenas um comprador viajante) e cada mercado m_i vende, a custo zero, somente o produto p_i que identifica unicamente este mercado.

4.2 Revisão da Literatura

Não há, até onde chega o conhecimento dos autores, outros trabalhos que exploram o CmTPP no escopo de planejamento de rotas para transportes nem no projeto de redes de

comunicação. Contudo, o CmTPP pode ser visto como uma generalização do CmRSP, que por sua vez modela muitas aplicações nos sistemas de telecomunicações, em particular no problema de redes de comunicação por fibras óticas descrito por Baldacci e colegas [5].

A literatura do Problema de m-Anéis Estrelados Capacitado (CmRSP) [5] já é bastante explorada. Nesta seção, apontamos alguns dos trabalhos do CmRSP, que está fortemente relacionado com o CmTPP e que aparecem na literatura recente.

Baldacci e colegas [5] popuseram duas formulações usando Programação Linear Inteira (PLI) e desenvolveram uma abordagem Branch and Cut (BC) para o CmRSP. O algoritmo foi testado em uma extensa gama de instâncias do problema, incluindo até duas instâncias de uma aplicação real. Os resultados mostraram que o algoritmo consegue provar a otimalidade de instâncias com até 50 clientes (ou 50 mercados/produtos usando uma modelagem CmTPP) em um tempo de computação razoável. Duas heurísticas foram também propostas em [5]. A primeira chamada de Heurística Inicial é, na verdade, uma adaptação do algoritmo introduzido por Baldacci e Dell'Amico [4] para um caso mais geral do problema chamado de CmRSP com Múltiplos Depósitos, e é executada no vértice raiz da árvore de enumeração do BC. Esta heurística transforma a instância de CmRSP em uma instância de VRP com demandas unitárias e aplica uma série de movimentos de buscas locais (Squeeze e LS-multiple) a fim de transformar a solução dada pelo VRP numa solução melhor para o CmRSP (veja [4] para mais detalhes). A segunda heurística, chamada de Heurística Primal, toma vantagem da informação obtida pela solução da relaxação do programa linear inteiro a fim de construir uma solução do CmRSP e é executada em um dado conjunto de vértices da árvore de enumeração.

Em 2007, Mauttone et al. [39] propuseram uma heurística híbrida para o CmRSP que combina GRASP e Busca Tabu.

Em 2008, Hoshino e de Souza [28] propuseram uma abordagem Branch and Price (BP) para o problema e compararam os resultados obtidos pelo BP com um método BC também implementado por eles que fora baseado no trabalho anterior de Baldacci e colegas [5]. Os experimentos computacionais mostraram que nenhum destes métodos domina o outro, sugerindo que algumas instâncias eram mais fáceis de serem resolvidas por BC e outras por BP. Este resultado estimulou-os a projetar e implementar uma abordagem Branch and Cut and Price (BCP), método que tenta combinar as forças do BC e do BP na resolução do problema combinatório em questão, que foi descrito em [29]. Neste trabalho, os experimentos computacionais apontaram que o BCP melhorava vários dos resultados alcançados pelo BC.

Em 2010, Naji-Azimi et al. [41] propuseram uma heurística em duas fases que combina a contrução de solução inicial com novas estruturas de busca local que atinge quase que a totalidade dos melhores resultados reportados para as instâncias do problema a um custo computacional baixo.

Algumas variantes do CmRSP que surgem de redes de comunicação também presentes na literatura são descritas por Baldacci et al. [5] e Labbé et al. [36].

O CmTPP também pode ser utilizado para modelar várias situações de roteamento de veículos, além da aplicação mais direta em que um grupo de compradores de uma empresa precisam trazer da forma mais econômica o conjunto de produtos. Apesar de não termos conhecimento de fontes na literatura, ele pode modelar, por exemplo: (a) a construção de rotas para veículos de ronda policial, onde cada ponto que o policial precisa ter visão é modelado como um produto que precisa ser comprado, e os locais pelos quais o policial pode passar para verificar estes pontos são modelados como mercados e; (b) o roteamento de ônibus fretados para o transporte de funcionários de uma empresa. Neste caso os funcionários são modelados como produtos e os pontos de ônibus onde eles podem se dirigir respeitando as leis trabalhistas são modelados como mercados, que é o caso que motivou os autores desta pesquisa.

O restante deste capítulo é organizado como segue. Na Seção 3 nós introduzimos o nosso método heurístico, este método é capaz de resolver instâncias do CmRSP através de uma transformação entre problemas. Uma destas transformações reduz o CmRSP ao CmTPP e é apresentada na Seção 4. Os resultados obtidos em instâncias de benchmark da literatura para o CmRSP são apresentados na Seção 5. Por fim, nossas conclusões e trabalhos futuros são apresentadas na Seção 7.

4.3 Descrição da Heurística para o CmTPP

Nesta Seção nós descrevemos a heurística para o CmTPP. Este algoritmo é dividido em três fases. De forma similar ao algoritmo do Capítulo 3, nós começamos gerando uma solução viável a partir de uma solução parcial. Esta solução é, então, iterativamente melhorada com uma heurística baseada na VNS. Por fim, exploramos alguns fundamentos da LNS. São realizadas várias iterações entre estas três fases até que um limite de iterações seja atingido. Portanto, como se vê, trata-se de uma extensão do algoritmo para o TPP que descrevemos no capítulo anterior.

4.3.1 O procedimento de construção de solução inicial

Nesta fase, uma heurística construtiva fundamentado no algoritmo apresentado na Seção 3.3.1, transforma uma solução parcial – que possivelmente não atende a todas as requisições da lista de requisições – em uma solução viável.

Novamente, a solução parcial que alimenta esta primeira fase da heurística pode ser tanto uma solução incompleta que contém alguns mercados, quanto uma solução que contém apenas o depósito. Na primeira iteração do nosso método para o CmTPP, uti-

lizamos uma solução que contém apenas o depósito. Nas próximas iterações, a solução parcial depende do resultado da Fase 3.

A idéia geral é construir uma solução parcial com m rotas (Passos 1 a 5 do Algoritmo 17) e, a partir desta solução, adicionar os próximos mercados de maneira gulosa entre estas rotas, respeitando as restrições de capacidade de cada uma delas. Estes passos são executados até que uma solução viável seja alcançada – e, então, a solução é retornada.

No Passo 3 adicionamos na solução o mercado que minimiza a soma dos custos de transporte e aquisição do primeiro item da lista. Nos Passos 4 e 5 criamos as $m-1$ rotas adicionando cada um dos $m-1$ mercados mais distantes dos mercados já presentes na solução, um em cada nova rota, com o intuito de gerar uma configuração de rotas parciais distantes uma das outras.

Dos Passos 6 a 9, tentamos alcançar a viabilidade da solução através da adição de mercados na solução parcial, de forma gulosa. Note que estes passos não garantem que cheguemos a uma solução viável.

Se a viabilidade não for alcançada, o Algoritmo da Fase 3 é chamado novamente para gerar uma solução parcial com uma configuração diferente a fim de re-iniciarmos a Fase 1. A Fase 3 nunca retorna uma solução parcial com menos de m rotas.

Algoritmo 17: HCM Fase 1: Algoritmo de Construção de uma Solução Viável

```
1  L = cria uma nova ordenação da lista de requisições;
2  v = mercado que minimiza a soma entre o custo de transporte e aquisição do
   primeiro item da lista L com uma função de ruído;
3  σ = solução parcial contendo o depósito e o mercado v;
4  para i de 2 a m faça
5      insira em σ o mercado que maximiza a somatória da distâncias entre ele e os
       mercados já presentes na solução;
6  para cada produto p na lista L que a demanda ainda não tenha sido satisfeita, em
   ordem faça
7      repita
8          adiciona, na solução parcial σ, o mercado que minimiza a soma entre uma
           função de ruído com o custo de roteamento e aquisição do produto p;
9      até que a demanda total por p seja atendida ou até que não seja possível
       adicionar mais mercados as rotas ;
10 se encontrou uma solução viável então
11     retorna a solução σ encontrada;
12 senão
13     use a Fase 3 da heurística para gerar uma nova solução parcial, gere uma nova
       ordenação da lista L e reinicie do Passo 6 deste algoritmo;
```

Note que estrutura de dados para guardar as informações de rotas e executar avaliações e movimentos das buscas locais é a mesma que foi usada no capítulo anterior, com a diferença de que agora temos m destas estruturas, guardando cada uma uma rota do CmTPP ao invés de uma rota apenas.

Entretanto, a avaliação de que aquisições devem ser realizadas a fim de minimizar estes custos necessita levar em consideração a capacidade máxima de cada rota. É por este motivo que a estrutura de dados para a avaliação do custo de aquisição dos produtos para este problema é bastante diferente da estrutura apresentada no Capítulo 3.

A solução que encontramos para que a capacidade máxima em cada rota fosse considerada foi modelar as aquisições como um problema de fluxo de custo mínimo em grafo, conforme pode ser observado na Figura 4.1.

Neste problema de fluxo de custo mínimo, existem $m + |M| + |P| + 1$ vértices. Há um vértice chamado de "origem"; m vértices que representam as rotas, $|M|$ vértices que representam os mercados e $|P|$ vértices que representam os produtos a serem comprados. De cada produto $p \in |P|$, sai um fluxo de d_p que é a demanda a ser atendida considerada naquele momento. Se a demanda considerada é a demanda total requisitada na lista de requisições e o problema de fluxo tem uma solução, então dizemos que esta solução é viável com relação a aquisição dos produtos. Entretanto, o durante nossa heurística, começamos considerando todas as demandas como 0 e vamos aumentando estes valores até atingirem o valor desejado, produto a produto, durante a fase de construção de uma solução viável, seguindo a ordem pré-determinada em $\mathcal{L}$ (veja novamente o Algoritmo 17).

O fluxo D que entra na origem é a soma dos fluxos d_p que saem dos vértices que representam os produtos. Os arcos que ligam a origem aos vértices que representam as rotas tem capacidade limitada a Q, que é a capacidade máxima de cada rota. Estes arcos tem custo zero.

Os arcos que ligam as *rotas* aos *mercados* tem custo zero, contudo, a capacidade é infinita se o mercado está alocado naquela rota, ou zero, se o mercado não está alocado a rota em questão.

Os arcos que ligam os *mercados* aos *produtos* tem custo igual ao custo unitário do produto no mercado em questão, e tem capacidade limitada ao número de unidades ofertadas do produto no respectivo mercado.

É possível mostrar que ao resolvermos o problema de fluxo teremos a alocação de aquisições dos produtos em cada mercado de custo mínimo.

Lembramos aqui que a ordem dos mercados em cada uma das rotas não altera a alocação de custo mínimo para as aquisições dos produtos, de forma semelhante ao relatado no capítulo anterior, para o caso de uma rota apenas.

Portanto, após solucionarmos este problema de fluxo basta verificar o fluxo de cada uma das arestas que liga os mercados aos produtos. O volume do fluxo que passa em cada

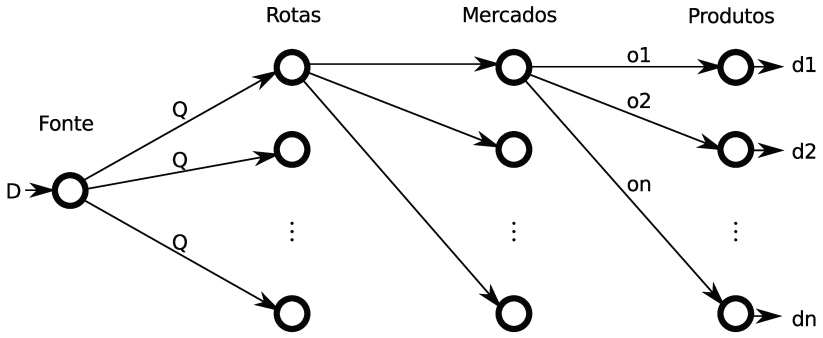


Figura 4.1: Estrutura de dados `aquisições`: representação do problema de fluxo de custo mínimo.

uma destas arestas é o número de unidades que são adquiridas pelo caixeiro comprador viajante no respectivo mercado.

4.3.2 O procedimento de busca local

De forma similar a segunda fase da metaheurística anterior, VND é utilizada para melhorar a qualidade da solução. Nossa implementação da VND oscila entre sete operadores de busca local descritos no capítulo anterior. Quatro destes operadores (veja Algoritmo 18): `2-opt`, `or-opt`, `i-opt` e `3-opt` foram projetados para realizar otimização no custo da solução atuando apenas na re-ordenação da sequência de mercados que estão na rota. Estes operadores não inserem novos mercados na rota nem retiram da rota os mercados previamente inseridos. Todavia, há outros três operadores projetados para esta função: `market-add`, `market-drop` e `market-exchange`. O único operador que considera trocas entre rotas diferentes é o `market-exchange`. Todas os operadores de busca local implementados usam a estratégia conhecida na literatura como *best-accept*, ou seja, aceitam a melhor solução da vizinhança depois de testar todos os vizinhos da solução corrente.

A cada avaliação de uma nova solução na vizinhança da atual resolvemos o problema de fluxo de custo mínimo apresentado na Subseção anterior, mudando as capacidades das arestas entre os vértices que representam as rotas e os mercados. Se o mercado é incluído na rota, deixamos passar fluxo, se ele é excluído, setamos a capacidade como 0 novamente.

4.3.3 A Metaheurística LNS

Na última fase da heurística aplicamos os conceitos da LNS a fim de gerar diversificação e intensificação no caminho da busca.

Algoritmo 18: HCM Fase 2: VND

```
1  σ = solução viável inicial;
2  se 2-opt(σ) é melhor que σ então
3  |  σ = 2-opt(σ);
4  |  volte para o passo 2;
5  se or-opt(σ) é melhor que σ então
6  |  σ = or-opt(σ);
7  |  volte para o passo 2;
8  se i-opt(σ) é melhor que σ então
9  |  σ = i-opt(σ);
10 |  volte para o passo 2;
11 se 3-opt(σ) é melhor que σ então
12 |  σ = 3-opt(σ);
13 |  volte para o passo 2;
14 se market-add(σ) é melhor que σ então
15 |  σ = market-add(σ);
16 |  volte para o passo 2;
17 se market-drop(σ) é melhor que σ então
18 |  σ = market-drop(σ);
19 |  volte para o passo 2;
20 se market-exchange(σ) é melhor que σ então
21 |  σ = market-exchange(σ);
22 |  volte para o passo 2;
23 retorna a solução σ encontrada;
```

De forma similar a heurística do Capítulo 3, usamos um algoritmo probabilístico que, tem $x\%$ de chance de retornar uma solução parcial vazia, $y\%$ de chance de retornar uma solução parcial baseada na retirada de mercados da solução corrente e $(100 - x - y)\%$ de chance de retornar uma solução parcial baseada na retirada de mercados da melhor solução encontrada até o momento na busca, de forma semelhante a heurística descrita no capítulo anterior.

O Algoritmo 19 exibe a listagem do procedimento LNS em questão. A função retira-mercados-probabilístico escolhe uma rota com probabilidade uniforme, dentre as rotas, e remove mercados de uma rota com probabilidade de 80%, deixando sempre pelo menos um mercado na rota além do depósito.

Os parâmetros x e y são parâmetros de ajuste do algoritmo para os quais usamos os mesmos valores do capítulo anterior.

Algoritmo 19: HCM Fase 3: LNS

```
1  ρ = um número uniformemente distribuído de 1 a 100;
2  se ρ ≤ x então
3  |  σ = solução-vazia;
4  senão
5  |  se ρ ≤ x + y então
6  |  |  σ = solução atual;
7  |  |  σ = retira-mercados-probabilístico(σ);
8  |  senão
9  |  |  σ = melhor solução até o momento;
10 |  |  σ = retira-mercados-probabilístico(σ);
11 retorna a solução parcial σ encontrada;
```

4.4 Descrição de uma redução do CmRSP para o CmTPP

Podemos descrever o CmRSP como um problema em grafos. É dado um grafo $G = (V \cup W, E \cup A)$ no qual $V \cup W$ é o conjunto de vértices, sendo W o conjunto dos chamados vértices *opcionais* ou de Steiner e V dado por $\{0\} \cup C$. O vértice 0 é chamado de depósito e C é o conjunto de vértices clientes que devem ser atendidos. As arestas do grafo são subdivididas em dois grupos: E é o conjunto de arestas de roteamento ligando pares de vértices, enquanto A é o conjunto de arestas de ligação.

O objetivo do CmRSP é encontrar m ciclos vértice-disjuntos no grafo, que passam por vértices de $V \cup W$, de forma que todos os vértices de C, ou façam parte de algum ciclo, ou

sejam ligados a um vértice que está em um ciclo de forma a minimizar a soma dos custos de roteamento $\sum c_e$ onde $e = (i, j) \in E$ é aresta de roteamento que liga o vértice i ao vértice j, e de ligação $\sum a_e$, onde $e = (i, j) \in A$ é aresta de ligação que liga o vértice i ao vértice j. Os vértices de W podem ser percorridos ou não pelos ciclos, dependendo se eles possibilitam reduções no custo da solução. Uma descrição mais detalhada do CmRSP, inclusive no que diz respeito a aplicações do problema, pode ser encontrada em [5]

A fim de transformarmos uma instância do CmRSP em uma instância do CmTPP, mapeamos todos os vértices da instância original $V \cup W$ como mercados na instância do CmTPP – inclusive o vértice que representa o depósito, que é mapeado como depósito da nova instância. Além disto, mantemos os custos de roteamento entre estes vértices como os custos de roteamento entre os respectivos mercados.

Já a ligação de vértices clientes a vértices do ciclo é simulada com a compra de produtos do mercado. Cada vértice cliente i é mapeado, também, para um produto, que é vendido a custo zero no respectivo mercado, e vendido ao custo das arestas de ligação $a = (i, j)$ para todo mercado j na instância do CmTPP.

Uma vez resolvido o CmTPP, os m ciclos resultantes são diretamente mapeados de volta como ciclos na solução do problema original, enquanto que as aquisições de produtos são mapeadas como as ligações de clientes que estão fora de ciclos para vértices em algum dos m ciclos.

A solução da instância transformada do CmTPP tem o mesmo custo final da solução do CmRSP.

É importante notar que resolver o CmRSP a partir desta redução para o CmTPP não implica em vantagem computacional alguma para sua resolução.

4.5 Experimentos Computacionais com instâncias do CmRSP

As classes de instâncias testadas: Nossos algoritmos foram testados nas duas classes de instâncias propostas por Baldacci e colegas em [5]. Cada uma destas classes, A e B, conta com 91 instâncias derivadas das instâncias eil51.tsp, eil76.tsp e eil101.tsp do conjunto de instâncias da TSPLIB (veja Reinelt [47]) que tem de 26 a 101 vértices. A topologia dos grafos subjacentes das instâncias de ambas as classes é a mesma e o que as difere é a estrutura de custo. Na classe A os custos de roteamento e de alocação para um dado par i e j de vértices é dado pela distância euclidiana entre estes vértices, computadas de acordo com o padrão da TSPLIB; enquanto que na classe B o custo de alocação é menor do que o de roteamento: o custo de transporte é calculado como a distância euclidiana multiplicada por 7 e o custo de alocação é calculado como a distância

euclidiana multiplicada por 3.

Ambiente de execução e testes realizados O nosso algoritmo foi implementado na linguagem C e os experimentos computacionais foram executados num computador Core 2 Duo com 2.0GHz e 4 GB RAM. O número total de iterações foi fixado em 200.

Rodamos 10 execuções independentes do algoritmo para cada uma das instâncias das classes A e B, com sementes diferentes para inicialização do gerador de números aleatórios. Os resultados obtidos estão reportados nas Tabelas 4.1 e 4.2. Nestas duas tabelas, na coluna `Nossa Heurística` e na coluna `Segundos` que segue aquela, reportamos o melhor valor da solução encontrado pelo nosso método e o tempo de relógio necessário para executar as 200 iterações do algoritmo, respectivamente

A primeira coluna é o nome da instância, nas quais os números que seguem as letras n e m são, respectivamente, o número de vértices e a capacidade de cada rota. Seguem o número de vértices não-opcionais, $|V|$, e o número máximo de clientes, Q, permitido por ciclo. As colunas 4 (IH) e 6 (PH) dão os limitantes primais da Heurística Inicial e da Heurística Primal obtidos pelos métodos heurísticos de Baldacci e Della'Amico [4] e Baldacci e colegas [5], respectivamente. A oitava coluna (BC) reporta a melhor solução obtida pelo método *branch and cut* de Baldacci e colegas [5]. A décima coluna (AST) mostra os resultados do trabalho de Naji-Azimi e colegas [41]. Os respectivos tempos de execução destes métodos são reportados nas colunas 5, 7, 9 e 11. A máquina usada em cada um dos conjuntos de experimentos computacionais é mostrada na última linha da tabela. O número de melhores soluções encontradas por cada método é reportado na penúltima linha das tabelas e a média de custo e tempo para todo o conjunto de instâncias é mostrada na antepenúltima linha.

Os resultados reportados nas Tabelas 4.1 e 4.2 mostram que para as instâncias com poucos clientes obrigatórios – ou seja, se mantivermos o mesmo n, consideramos os menores $|V|$ – foram as instâncias nas quais o nosso algoritmo se compara melhor em relação aos outros métodos heurísticos (HI, PH e AST) e estão próximos da solução entregue pelo algoritmo exato BC. Notamos que nessas instâncias a solução do CmRSP que foi retornada pelo nosso algoritmo tem as características de uma solução de VRP com nenhum (ou poucos) cliente(s) fora dos ciclos.

Atingimos, em termos de custo de solução, os melhores resultados em 19 das 46 instâncias da Classe A – enquanto as heurísticas HI, PH e AST atingiram em 21, 21 e 44 instâncias. Para as instâncais da classe B, atingimos as melhores soluções em 18 das 45 instâncias – enquanto as concorrentes atingiram em 15, 16 e 44 respectivamente..

Para as instâncias menores, a nossa heurística não é eficiente: enquanto as outras heurísticas comparadas chegam ao tempo de no máximo 2.9 segundos, a nossa leva cerca de 5 vezes mais tempo para alcançar o mesmo resultado na instância B09-n026-m05, por

Instância	$\|V\|$	Q	IH	Segundos	PH	Segundos	BC	Segundos	AST	Média segundos Sol.	Nossa Heurística	Segundos
A01-n026-m03	12	5	242	0.3	242	0.1	242	0.1	242	0	242	8.2
A02-n026-m04	12	4	261	0.3	261	0	261	0	261	0	261	15.6
A03-n026-m05	12	3	292	0.3	292	0	292	0	292	0	292	8.0
A04-n026-m03	18	7	301	0.4	301	0.7	301	0.5	301	0	301	4.0
A05-n026-m04	18	5	339	0.4	339	0.4	339	0.3	339	0	339	8.3
A06-n026-m05	18	4	375	0.4	375	1.4	375	0.7	375	0	375	8.9
A07-n026-m03	25	10	333	0.8	333	1.7	325	3.8	325	0	333	14.1
A08-n026-m04	25	7	362	0.7	362	0.9	362	0.3	362	0	366	8.5
A09-n026-m05	25	6	382	0.6	382	0.6	382	0.2	382	0	382	10.9
A10-n051-m03	12	5	242	0.3	242	0.1	242	0.2	242	0	242	11.9
A11-n051-m04	12	4	261	0.2	261	0.1	261	0.4	261	0	282	7.9
A12-n051-m05	12	3	286	0.3	286	0.1	286	0.1	286	0	286	16.6
A13-n051-m03	25	10	331	0.8	322	1	322	2.1	322	0	322	17.1
A14-n051-m04	25	7	360	0.7	360	1.1	360	2.1	360	0	360	13.4
A15-n051-m05	25	6	379	0.6	379	1.7	379	2.3	379	0	379	8.2
A16-n051-m03	37	14	373	2.3	373	6.7	373	8.4	373	0	373	14.7
A17-n051-m04	37	11	408	1.6	408	7.6	405	41.7	405	0.1	408	14.3
A18-n051-m05	37	9	441	2.2	435	11.8	432	52.2	432	0.1	435	9.0
A19-n051-m03	50	19	459	4.8	469	14.1	458	182.8	458	0.1	469	10.3
A20-n051-m04	50	14	501	3	493	20.8	490	220.4	490	0.2	507	15.1
A21-n051-m05	50	12	521	5.3	521	19.2	520	6334.2	520	0.1	521	21.6
A22-n076-m03	18	7	330	0.7	330	2.9	330	48.3	330	0	330	13.6
A23-n076-m04	18	5	385	0.6	385	2.7	385	30.6	385	0	385	28.2
A24-n076-m05	18	4	448	0.8	448	4.2	448	63.7	448	0	448	15.5
A25-n076-m03	37	14	407	2.2	409	9.5	402	567.7	402	0.1	409	21.9
A26-n076-m04	37	11	462	2.3	461	16.5	460	7200	457	0.1	461	18.5
A27-n076-m05	37	9	479	3.1	484	21.4	479	509.3	479	0.1	480	14.1
A28-n076-m03	56	21	475	7.3	478	38.9	471	1584.4	471	0.3	481	26.0
A29-n076-m04	56	16	523	7.1	524	50.5	523	7200	519	0.1	523	25.8
A30-n076-m05	56	13	552	6.3	552	40.2	545	3221.3	545	0.2	552	23.4
A31-n076-m03	75	28	570	14.8	565	45	564	479.5	564	0.6	570	18.7
A32-n076-m04	75	21	617	15.3	628	57.4	606	7200	602	0.4	617	24.3
A33-n076-m05	75	17	659	13.6	654	81.7	654	7200	640	0.6	659	23.3
A34-n101-m03	25	10	363	0.9	363	3.2	363	8.7	363	0	363	44.6
A35-n101-m04	25	7	415	1.1	415	9.2	415	91.8	415	0	415	27.5
A36-n101-m05	25	6	448	1.5	448	10.8	448	680.4	448	0	448	21.5
A37-n101-m03	50	18	503	6.5	501	58.8	500	7200	500	0	501	28.1
A38-n101-m04	50	14	532	3.9	533	44.5	532	7200	528	0.1	533	35.5
A39-n101-m05	50	12	571	4	568	48.4	568	7200	567	0.1	589	45.6
A40-n101-m03	75	28	605	18.6	622	115.6	595	6690.1	595	0.5	635	36.0
A41-n101-m04	75	21	629	13.3	635	74.5	625	7200	623	0.4	641	21.0
A42-n101-m05	75	17	663	11.5	665	120.5	662	7200	657	1	663	38.2
A43-n101-m03	100	38	672	31.7	672	134.3	646	283	647	2	672	24.8
A44-n101-m04	100	28	702	26.5	704	109	680	7200	679	1.7	702	30.5
A45-n101-m05	100	23	719	24.5	717	148.7	700	1310.8	700	2.3	723	21.0
Average			448.4	5.4	448.82	29.7	444.62	2098.3	443.8	0.2	450.6	190.4
# best			21		21		36		44		19	
			Pentium IV, 3.4 GHz				Pentium IV, 2.2 GHz		Intel, 1.66 GHz		Core 2 Duo, 2.0 Ghz	

Tabela 4.1: Tabela Resultados Classe A

exemplo. Contudo, percebemos que nossa heurística é comparável com as heurísticas PH e IH nas instâncias maiores.

Notamos claramente que a heurística AST representa o estado da arte neste problema.

Uma execução assistida do nosso método aponta que a parte mais custosa em tempo da nossa heurística é a fase de busca local e que, dentro da busca local a avaliação na estrutura de aquisições ocupa cerca de 98% do tempo. É nesta etapa que utilizamos o modelo de fluxo de custo mínimo, que é executado várias vezes para cada iteração da busca local.

A comparação com os algoritmos que resolvem o CmRSP da literatura não é muito justa, visto que o nosso algoritmo resolve um problema mais geral que o CmRSP, o CmTPP. Estes algoritmos heurísticos tiram proveito das características intrínsecas do primeiro problema, que certamente beneficiam os algoritmos em termos tanto de qualidade de solução e de tempo de execução. Instâncias do CmTPP devem ser criadas a fim de testar o nosso algoritmo e servirem de referência para os próximos trabalhos na área.

4.6 Conclusões e trabalhos futuros

Neste capítulo nós estendemos o algoritmo heurístico previamente proposto para o TPP no capítulo anterior a fim de resolver o problema mais geral com m caixeiros compradores viajantes e com restrições de capacidade. Analisamos o desempenho deste algoritmo que, até onde sabemos, é o primeiro algoritmo heurístico que resolve o CmTPP.

Comparamos esta heurística com os métodos heurísticos do estado da arte de um problema menos geral, utilizando o conjunto de instâncias do CmRSP e apontamos que a nossa heurística é competitiva com estes métodos.

Uma melhoria interessante de implementação seria elaborar um método combinatório que substituísse o modelo de fluxos em rede, e que tirasse vantagem da estrutura do problema de fluxo que criamos, de forma similar ao que foi feito no trabalho de Vaidyanathan e Ahuja [59].

Outra melhoria interessante seria a criação de um algoritmo mais sofisticado para a criação da primeira solução viável que não dependesse da avaliação do modelo de fluxo de custo mínimo a cada passo da construção.

Por fim, um conjunto de instâncias do CmTPP deve ser criado a fim de prover uma forma de comparação confiável para os próximos trabalhos sobre este problema.

Instância	$\|V\|$	Q	IH	Segundos	PH	Segundos	BC	Segundos	AST	Média segundos Sol.	Nossa Heurística	Segundos
A01-n026-m03	12	5	242	0.3	242	0.1	242	0.1	242	0	242	11.9
B01-n026-m03	12	5	1684	0.3	1684	0.1	1684	0.1	1684	0	1684	10.9
B02-n026-m04	12	4	1827	0.2	1827	0.1	1827	0.1	1827	0	1827	7.6
B03-n026-m05	12	3	2041	0.3	2041	0	2041	0	2041	0	2041	7.5
B04-n026-m03	18	7	2104	0.4	2104	0.6	2104	0.5	2104	0	2104	9.3
B05-n026-m04	18	5	2370	0.4	2370	1.5	2370	0.5	2370	0	2370	9.7
B06-n026-m05	18	4	2615	0.5	2615	2.2	2615	0.7	2615	0	2615	13.5
B07-n026-m03	25	10	2314	0.8	2251	1.6	2251	0.4	2251	0	2251	6.1
B08-n026-m04	25	7	2510	1.1	2510	1.2	2510	0.5	2510	0	2510	7.2
B09-n026-m05	25	6	2674	0.8	2674	2.9	2674	0.8	2674	0	2674	15.0
B10-n051-m03	12	5	1681	0.3	1681	0.4	1681	0.8	1681	0	1681	11.8
B11-n051-m04	12	4	1821	0.2	1821	0.6	1821	1.5	1821	0	1821	22.9
B12-n051-m05	12	3	1972	0.3	1972	0.2	1972	0.3	1972	0	1972	11.1
B13-n051-m03	25	10	2176	1.5	2176	1.6	2176	1.1	2176	0	2176	13.2
B14-n051-m04	25	7	2476	1.1	2495	4.1	2470	7.2	2470	0	2488	5.4
B15-n051-m05	25	6	2596	1	2579	2.4	2579	4.1	2579	0	2579	23.0
B16-n051-m03	37	14	2507	2.3	2599	9.4	2490	17.9	2490	0.2	2507	14.3
B17-n051-m04	37	11	2772	1.9	2811	10.5	2721	74.9	2721	0	2772	12.9
B18-n051-m05	37	9	2938	2.2	2937	14.2	2908	145	2908	0.2	2938	12.7
B19-n051-m03	50	19	3095	4	3071	17.4	3015	296.7	3015	0.2	3095	14.7
B20-n051-m04	50	14	3365	3.6	3298	18.1	3260	336.6	3260	0.2	3365	16.7
B21-n051-m05	50	12	3525	5.7	3516	18.9	3404	6470.7	3404	0.6	3525	21.0
B22-n076-m03	18	7	2260	0.7	2259	2.6	2253	105.5	2253	0.1	2259	25.9
B23-n076-m04	18	5	2625	0.5	2620	3.3	2620	29.5	2620	0	2620	19.9
B24-n076-m05	18	4	3059	0.9	3059	3.4	3059	85.3	3059	0	3059	18.1
B25-n076-m03	37	14	2742	3.1	2720	14.3	2720	1897.6	2720	0.1	2742	17.3
B26-n076-m04	37	11	3176	2.7	3138	17.5	3138	7200	3100	0.6	3176	14.5
B27-n076-m05	37	9	3339	3	3364	23.8	3311	7200	3284	0.1	3339	18.2
B28-n076-m03	56	21	3112	7.1	3146	31.4	3088	7200	3044	0.9	3112	25.6
B29-n076-m04	56	16	3447	5.1	3496	50.3	3447	7200	3415	1.2	3447	24.3
B30-n076-m05	56	13	3652	4.6	3703	35.5	3648	7200	3636	0.9	3652	21.1
B31-n076-m03	75	28	3786	14.1	3820	69.1	3740	7200	3652	1.2	3786	23.5
B32-n076-m04	75	21	4057	13.9	4084	78.8	4026	7200	4003	1.8	4057	20.7
B33-n076-m05	75	17	4442	15.9	4288	54.5	4288	7200	4217	2.2	4442	35.9
B34-n101-m03	25	10	2437	0.7	2439	4.3	2434	24.2	2434	0	2444	32.6
B35-n101-m04	25	7	2782	1.2	2819	9.5	2782	115.4	2782	0	2782	34.5
B36-n101-m05	25	6	3043	1	3012	4.8	3009	862.4	3009	0	3052	39.7
B37-n101-m03	50	18	3404	6.3	3387	37.2	3332	7200	3322	0.3	3332	37.0
B38-n101-m04	50	14	3593	4.3	3586	32.1	3533	7200	3533	0.1	3553	41.8
B39-n101-m05	50	12	3880	4.4	3872	33.2	3872	7200	3834	1.4	3880	47.9
B40-n101-m03	75	28	3935	26	3923	260.7	3923	7200	3887	2.7	3935	36.4
B41-n101-m04	75	21	4190	16	4202	63.5	4125	7200	4082	2	4187	40.3
B42-n101-m05	75	17	4486	14.2	4458	47.9	4458	7200	4358	1.6	4486	35.3
B43-n101-m03	100	38	4275	35.9	4155	103	4110	7200	4135	9.3	4275	35.8
B44-n101-m04	100	28	4583	28	4608	97.4	4506	7200	4358	6.2	4553	39.4
B45-n101-m05	100	23	4671	26.7	4639	114.6	4632	7200	4567	6.4	4771	36.2
Average			448.4	5.4	448.82	29.7	444.62	2098.3	443.8	0.2	2920.2	21.6
# best			15		16		31		44		18	
			Pentium IV, 3.4 GHz				Pentium IV, 2.2 GHz		Intel, 1.66 GHz		Core 2 Duo, 2.0 Ghz	

Tabela 4.2: Tabela Resultados Classe B

Capítulo 5

O Problema "Rico" de Transporte dos Funcionários da UNICAMP

Neste capítulo estudamos o problema que surge da situação real do projeto das rotas licitadas pela UNICAMP para o transporte de parte dos seus funcionários, o que corresponde ao terceiro problema daqueles apresentados no final do Capítulo 1.

No texto que segue, descrevemos o problema original que motivou esta dissertação de mestrado, a interface gráfica de roteamento produzida e a interação entre ela, os algoritmos apresentados nos dois capítulos anteriores e o engenheiro de rotas. Apresentamos alguns resultados preliminares obtidos da aplicação deste sistema e descrevemos possíveis extensões deste trabalho.

5.1 Introdução

A maioria dos esforços na criação de heurísticas de otimização que aparecem na literatura foca em produzir algoritmos que são mais eficientes que os anteriores em algumas classes de instâncias da literatura.

A eficiência do algoritmo é, geralmente, medida pelo custo das soluções que ele produz de acordo com uma função objetivo bem definida (como a distância total percorrida pelos veículos ou o número de rotas utilizadas na solução, por exemplo) e com o tempo computacional requerido para produzir tais soluções. Este é o caso dos algoritmos propostos para os dois problemas descritos nos capítulos anteriores.

Entretanto, conforme apontam Klau e seus colaboradores [32], "esta abordagem de avaliação de sistemas de otimização é insuficiente para muitos dos contextos de problemas reais das quais os problemas de otimização surgem, e isto pode ser detectado em duas situações: (1) a pessoa responsável pela apresentação da solução no ambiente onde o problema real surge deve explicá-la ou justificá-la para outros com o objetivo de ob-

ter cooperação no esforço de implementar a solução na prática; e precisa de meios para modificar a solução quando eventos inesperados ocorrem; e (2) os sistemas de otimização geralmente resolvem apenas uma aproximação do problema que é de real interesse para os seus usuários. As pessoas geralmente sabem muito mais sobre o problema do que elas podem especificar antecipadamente à construção do método resolvedor ou simplesmente não podem especificar o que elas sabem usando uma linguagem de especificação de problemas."

Ou seja, nas situações reais os sistemas de otimização também devem ser avaliadas as características que auxiliam o usuário na implementação – na prática – das soluções geradas pelos algoritmos e na modificação destas soluções para que atendam os requisitos do problema real – e não na aproximação deste. Por isto, a tarefa de medir o desempenho destes sistemas é tão penoso.

Verificamos que as duas necessidades citadas nos parágrafos anteriores estão presentes no ambiente da UNITRANSP – no qual uma solução precisa ser justificada e aprovada por um conselho antes de ser implementada – e no escopo deste trabalho – uma vez que as heurísticas implementadas atendem apenas a uma aproximação do problema real e as soluções produzidas devem ser lapidadas por um engenheiro de rotas a fim de que se tornem passíveis de implementação.

Uma possível abordagem para suprir estas necessidades consiste na integração dos algoritmos implementados dentro de uma interface gráfica com o usuário na qual o engenheiro de rotas pode editar dinamicamente as soluções sugeridas pelos algoritmos e executar os métodos otimizadores automáticos a fim de re-otimizar partes da solução.

Desta forma, o poder de decisão no projeto das rotas é dividido entre os métodos resolvedores automáticos e o engenheiro de rotas, e assim, torna-se menos penoso o processo de explicar a solução e justificá-la aos superiores a fim de conseguir o apoio na implementação da mesma, uma vez que o engenheiro participa ativamente da produção da solução. E mais, com o poder de editar as rotas sugeridas pelos algoritmos de forma livre, o próprio engenheiro pode aplicar seu conhecimento do problema a fim de chegar a uma solução que atenda os requisitos da situação real, mesmo que as heurísticas utilizadas gerem soluções para uma aproximação desta.

Apresentamos o problema real de roteamento de forma detalhada no decorrer deste capítulo, chamado aqui de Problema do Transporte Fretado dos Servidores da UNICAMP (PTFSU), na Seção 5.2. Esta descrição detalhada é uma de nossas maiores contribuições apresentadas neste capítulo. Ainda na Seção 5.2 explicitamos as características do problema real que são levadas em consideração pelas heurísticas produzidas e quais são os requisitos do problema em que o conhecimento do engenheiro de rotas é utilizado a fim de se obter uma solução viável para o problema real. Uma revisão bibliográfica da literatura de problemas semelhantes e de sistemas interativos é dada na Seção 5.4.

Fazemos uma breve apresentação das funcionalidades da interface gráfica na Seção 5.5 e na Seção 5.6 apresentamos alguns casos de uso, a fim de exemplificar a operação do sistema e a interação do usuário com a interface e algoritmos implementados. As instâncias e resultados obtidos com o sistema são descritos na Seção 5.7. Por fim, na Seção 5.8, tecemos conclusões sobre os resultados preliminares obtidos e indicamos algumas direções para trabalhos futuros.

5.2 Descrição do Problema da UNICAMP

A UNICAMP oferece a seus funcionários a possibilidade de se deslocar de suas respectivas residências aos locais de trabalho, e vice-versa, através de ônibus fretados.

O fretamento é subsidiado em parte pela Universidade e custa atualmente cerca de R$ 7.000.000,00 (sete milhões de reais) anuais para essa instituição [53]. As linhas licitadas são definidas pela própria UNICAMP e licitadas publicamente. O custo incorrido pela Universidade é proporcional à quilometragem total percorrida pelos veículos nas linhas. Portanto, é crucial que se busque a minimização desta quantia durante o processo de definição das linhas.

Cada linha é formada por duas rotas – uma no sentido residências–UNICAMP e outra no sentido contrário. As restrições associadas à rota de uma linha são complexas. Deve-se garantir que, por exemplo, se a rota não passa na frente da residência do funcionário, o mesmo não deve andar a pé mais do que 500[1] metros da porta de sua residência para chegar ao local no qual ele entra no seu respectivo ônibus.

Também é possível que alguns dos funcionários recebam um vale-transporte para se deslocarem utilizando o sistema de transporte urbano antes de tomar o transporte fretado pela UNICAMP.

O número de funcionários numa linha não deve exceder a capacidade de carga do veículo fretado – tipicamente 44 passageiros, mas que varia com o tipo de veículo empregado para o transporte – e o intervalo de tempo que o deslocamento do funcionário leva – indo para ou voltando da UNICAMP – não deve ser maior que um certo número de horas e o horário em que cada funcionário deve chegar ao local de trabalho também deve ser respeitado.

Além disso, a distância total de cada rota a ser considerada no processo de licitação é a distância percorrida a partir do local em que há a coleta do primeiro funcionário da rota embarca no ônibus, até o local em que o último funcionário atendido pela rota sai do ônibus. Esta característica implica numa configuração "aberta" das rotas ou Problema de Roteamento de Veículos com Rotas "Abertas" – conforme o citamos ao final da Sub-seção

[1]Esta distância é uma estimativa que usamos para realizar os testes. Os algoritmos e a interface estão preparados para suportar mudanças neste valor.

2.1.1. Ou seja, o veículo não faz um circuito fechado saindo da UNICAMP e voltando até a UNICAMP após transportar todos os funcionários.

Ainda que todos os funcionários trabalhem na UNICAMP este problema não é caracterizado como um problema de roteamento com depósito único. Na verdade, a Universidade abrange uma área bastante extensa no distrito de Barão Geraldo, e cada funcionário deve ser entregue próximo ao seu local de trabalho. Este problema é melhor caracterizado como um Problema de Roteamento de Veículos com Coleta e Entrega (PDVRP), veja Sub-seção 2.1.1, no qual os pontos de *coleta* e de *entrega* são definidos para cada funcionário, individualmente.

Cada uma das linhas – ou um conjunto delas, dependendo do interesse da Universidade – é licitada por períodos que variam de 1 a 3 anos. Além disto, cada linha licitada pode aumentar (ou diminuir) sua distância percorrida em algum valor percentual[2] previsto em contrato – a fim de atender novos funcionários que por ventura venham a utilizar o trasporte fretado, ou deixar de atender antigos que saíram do sistema. Dessa forma, é necessária a manutenção constante das rotas, linhas e localização das residências dos funcionários.

Uma solução para este problema é dada por um conjunto de rotas que atenda a todos os funcionários e minimize a soma ponderada da (1) somatória das distâncias percorridas e (2) do número de rotas.

Em suma, para resolver o problema de transporte fretado na UNICAMP deve-se *minimizar os custos de transporte* observando as seguintes características:

- todo o funcionário é atendido;
- alguns funcionários podem ganhar um vale transporte a fim de se deslocarem de suas residências até um ponto mais próximo de uma rota de fretados.
- cada funcionário tem um local de trabalho no qual deve embarcar e no qual deve desembarcar e não deve andar a pé mais do que uma distância pré-determinada – antes do embarque ou depois do desembarque;
- cada funcionário tem um horário para chegar e sair do trabalho e não deve esperar mais do que um intervalo curto de tempo para tomar assento no ônibus;
- o funcionário não pode permanecer no ônibus mais do que um certo intervalo máximo de tempo;
- a capacidade do veículo não deve ser excedida;

[2]Na época do levantamento do problema, estes dados variavam de acordo com o contrato, e estavam em torno de 10%

- deve-se considerar uma frota heterogênea de veículos;
- a distância percorrida pelo ônibus é medida desde o primeiro funcionário que embarca até o último que desembarca do ônibus, o que resulta numa configuração de rotas abertas.

5.3 Características do problema que são amparadas pelos resolvedores automáticos e características que necessitam da intervenção do engenheiro de rotas

O problema do transporte de funcionários da UNICAMP é caracterizado por agregar vários tipos diferentes de restrições, algumas classificadas – de forma unânime pela literatura – como difíceis mesmo quando consideradas separadamente. Estas restrições complicam sobremaneira o projeto e implementação de métodos resolvedores. Esse fato nos motivou a propor uma versão simplificada do problema real de transporte da UNICAMP, que não conta com todos os requisitos do problema real, mas que atende a características imprescindíveis para que o sistema possa ser utilizado de forma semi-automatizada na UNITRANSP.

Assim, dividimos o trabalho de projetar as rotas em duas fases: na primeira fase, os algoritmos otimizadores sugerem rotas que resolvem uma aproximação do problema real e, na segunda, as rotas sugeridas são editadas por um engenheiro a fim de que se tornem passíveis de serem operacionalizadas. Ou seja, com auxílio da intervenção do engenheiro de rotas, as rotas criadas pelos algoritmos podem ser implementadas na prática.

Neste caso, os algoritmos otimizadores consideram a variante simplificada do problema que é caracterizada da seguinte forma:

- todo o funcionário é atendido;
- cada funcionário não pode andar mais do que 500 metros para entrar no ônibus.
- todos os funcionários desembarcam num mesmo ponto central na UNICAMP;
- não se consideram mais os turnos de trabalho dos funcionários;
- a frota de veículos fretados é homogênea – todos tem a mesma capacidade de passageiros e esta não deve ser excedida;
- a distância percorrida pelo ônibus é medida desde o primeiro funcionário que embarca até o último que desembarca do ônibus.

Nota-se que as restrições desta variante simplificadas são versões relaxadas do problema original apresentado na Seção anterior, e que a intervenção humana é necessária para transformar as rotas geradas como solução deste problema para que se tornem uma solução do problema real a fim de que possam ser implantadas.

Dentre estas intervenções humanas necessárias, estão:

- criar o caminho dos veículos dentro da UNICAMP a fim de que passem nos pontos necessários para o embarque/desembarque dos funcionários que participam da rota;
- dividir os funcionários em conjuntos que correspondem aos turnos de trabalho, a fim de que nas rotas criadas hajam apenas funcionários do mesmo turno de trabalho;
- verificar a possibilidade de usar diferentes tipos de veículos (frota heterogênea);
- adaptar as rotas e a composição do conjunto de funcionários atendidos pelas rotas a fim de que satisfaçam eventuais preferências dos envolvidos.

A seguir, revisamos alguns sistemas de otimização que permitem a intervenção humana.

5.4 Revisão de Problemas Relacionados

Existem alguns estudos sobre problemas de roteamento relacionados ao problema da UNICAMP que foram publicados na literatura.

Ceselli e colaboradores [12] construíram um modelo e um algoritmo exato baseado em geração de colunas para um problema rico que atende a diversos conjuntos de restrições, como locais de coleta e entrega e janelas de tempo, além da possibilidade de contratar serviços de *delivery* para entregar pequenas quantidades de produto. Outras características também são consideradas neste modelo: o uso de frota heterogênea, incompatibilidades entre produtos (alguns tipos de produtos não podem ser transportados no mesmo veículo, apesar de o veículo poder transportar todos os tipos de produtos) e incompatibilidade entre produtos e veículos (alguns produtos só podem ser transportados por veículos específicos). A função objetivo usada é computada através do que eles chamam de "taxas"[3], um conceito que generaliza as funções objetivo tradicionais, que consideram apenas distância percorrida e número de veículos na frota. Esse modelo pode ser visto como uma extensão dos modelos propostos no capítulo anterior.

Todos os trabalhos citados e revisados nos capítulos anteriores também são relacionados ao problema enfrentado pela UNICAMP. Na verdade, como já foi mencionado

[3]usamos "taxas" como uma tradução livre para *fees*, em inglês

antes neste capítulo, os algoritmos apresentados nesses capítulos resolvem modelos que são aproximações do problema real de transporte da Universidade. Explicamos como estes algoritmos são usados na interface através dos casos de uso na Seção 5.6.

5.5 Uma Breve Apresentação da Interface Gráfica com o Usuário do Sistema

A fim de viabilizar a intervenção humana sobre a solução gerada pelos algoritmos, desenvolvemos uma interface gráfica com o usuário. Com ela o usuário pode facilmente criar, editar e apagar rotas numa solução. Além disto, ele pode invocar os algoritmos resolvedores a fim de obter novas sugestões de rotas. Assim, esta interface gráfica serve de ponte para que os algoritmos descritos nos dois Capítulos anteriores sejam utilizados na prática do dia-a-dia da UNITRANSP.

Nesta Seção listamos as funcionalidades que identificamos como sendo necessárias para o sistema automatizado de fretamento. Abaixo definimos alguns termos utilizados nesta Seção.

1. *Paradas* são os pontos de ônibus nos quais funcionários inscritos no sistema de fretados devem pegar um ônibus. A UNICAMP é tratada como se fosse uma única parada para os algoritmos. Entretanto, pontos de ônibus dentro da Universidade podem ser transformados em paradas pelo engenheiro de rotas a fim de que o veículo pare para o embarque/desembarque de funcionários próximo aos respectivos locais de trabalho.

2. *Segmentos* representam eixos de ruas no mapa digital da cidade.

3. Um *Caminho* é composto de um ou mais segmentos direcionados que ligam duas paradas.

4. Uma *Rota* é um conjunto de caminhos entre paradas, iniciando numa parada qualquer e terminando necessariamente na UNICAMP no caso de rotas que trazem o funcionário para a UNICAMP, ou em sentido contrário, caso a rota leve os funcionários de volta para suas respectivas residências.

5. Um *Cenário* é um dado por um conjunto funcionários, por um conjunto de rotas e por um conjunto de paradas não atendidas, podendo um destes dois últimos conjuntos estar vazio. Ele também é definido por um intervalo de tempo, chamado de turno. Este intervalo de tempo é usado para classificar os funcionários quanto aos turnos de entrada e saída do horário de trabalho.

A seguir, descrevemos as funcionalidades do *software*.

- Cenários:
  1. Criar novo cenário;
  2. Abrir cenário existente;
  3. Excluir cenário existente;
  4. Modificar propriedades do cenário.
- Paradas:
  1. Criar nova parada;
  2. Excluir parada;
  3. Editar dados da parada;
  4. Ver número de funcionários da parada;
  5. Ver lista de funcionários da parada;
- Funcionários:
  1. Cadastro, alteração e remoção de funcionários;
  2. Mostrar rota de um funcionário.
  3. Relatório de funcionários que não estão alocados em rotas.
- Empresas:
  1. Cadastro, alteração e remoção de empresas;
  2. Mostrar todas as rotas de uma empresa;
  3. Mostrar todos os contratos de uma empresa;
  4. Imprimir todos os dados de cada rota de uma empresa.
- Rotas:
  1. Construção manual de rotas;
  2. Sugestão automática de rotas;
  3. Alteração manual de rotas:
- Sistema de Zoom:
  1. Zoom in/out no nosso mapa.

- Outras Funcionalidades/Relatórios:

    1. Impressão da lista de funcionários, paradas, janelas de tempo e empresas, para que cada funcionário saiba onde, quando e com que empresa virá até a UNICAMP;
    2. Impressão da lista de rotas disponíveis para licitação.

Cada uma das linhas gerenciadas pela UNITRANSP é composta por duas rotas – uma que traz os funcionários para a UNICAMP e outra que os leva devolta para suas respectivas residências. A Figura 5.1 ilustra como uma linha é representada na interface. A rota em azul mostra o sentido residências-UNICAMP e a vermelha mostra o outro.

5.6 Método de Resolução e o Papel Interface Gráfica com o Usuário do Sistema

Exemplo de Uso do *Software:*

- Definindo o cenário de trabalho: ao abrir o Aplicativo o usuário escolhe entre "Criar novo cenário" ou "Abrir cenário existente". Ao escolher "Criar novo cenário", uma tela aparece requisitando informações referentes ao cenário que ele deseja criar ('Nome do cenário', 'Data de criação', 'Descrição do cenário', etc.). Escolhendo "Abrir cenário existente" o cenário existente é mostrado ao usuário, simplesmente. Trabalhando com "Cenários", além de 'abrir' e 'criar novo', o usuário deve ter a capacidade de 'excluí-los' e 'modificar os detalhes' (como 'Nome', 'Descrição', etc.).

- Cadastrando funcionários/empresas: o usuário deve cadastrar os funcionários antes de poder indicar o local de residência e de trabalho do mesmo. Logo uma tela de "Cadastro de funcionários" aparece após o comando do usuário, requisitando dados a serem preenchidos. Da mesma forma ocorre o processo relacionado com o "Cadastro de empresas". Há também a possibilidade de ativar e desativar o funcionário/empresa já cadastrado no sistema.

- Construção manual de rotas: o usuário pode habilitar o modo de construção manual de rotas para criar rotas para os funcionários que não tem rotas. Quando o modo de contrução de manual rotas é habilitado o usuário ativa a contrução de uma rota, seleciona os segmentos, um a um e em ordem, na rota, da origem da rota até o destino final. O usuário então seleciona os funcionários que são atendidos pela rota criada, e as paradas as quais eles utilizam, e a empresa que serve a rota. Por fim, a rota é salva.

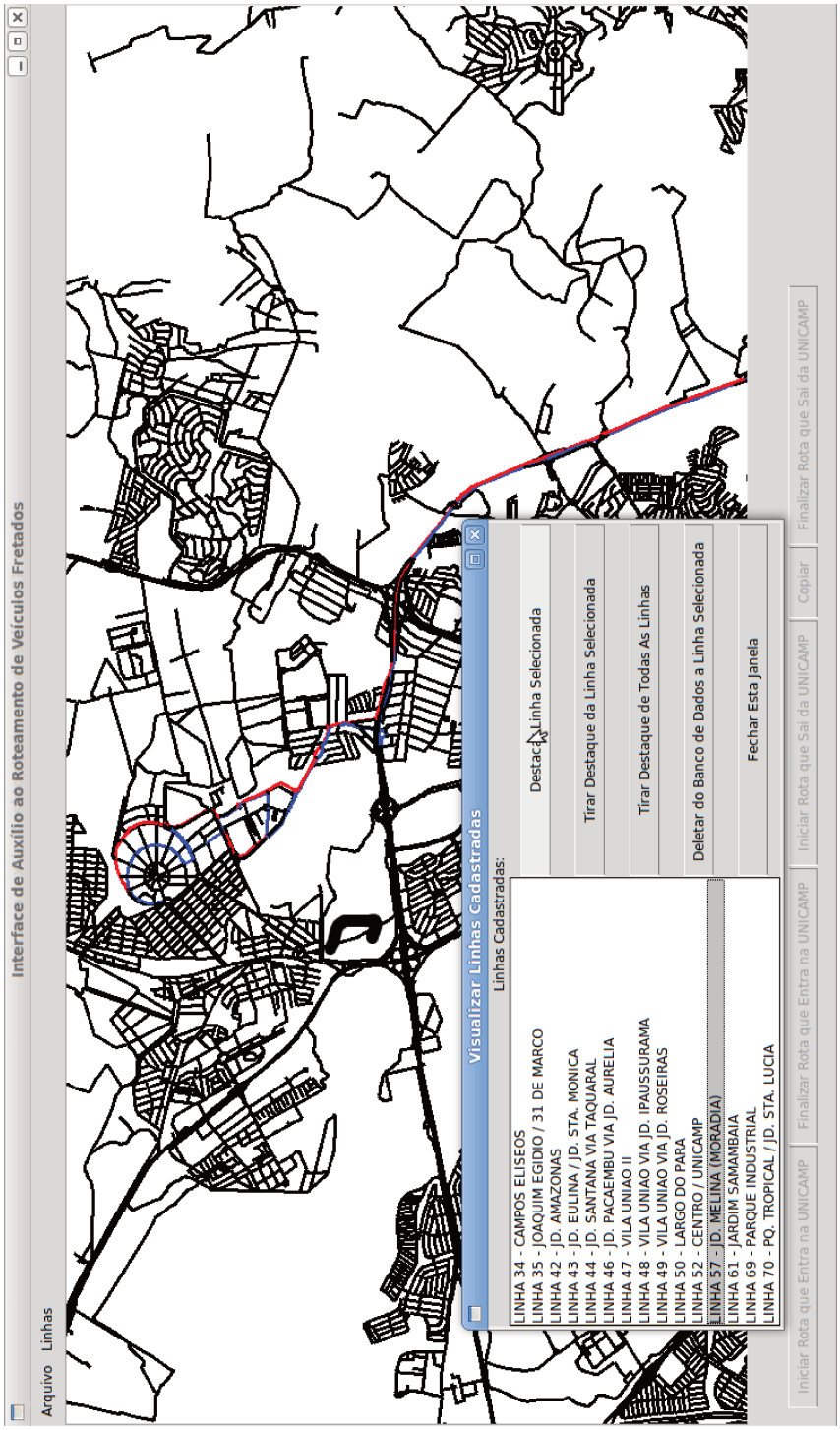


Figura 5.1: Ilustração de uma linha na interface gráfica.

- Construção automática de rotas: há dois modos de construir rotas de forma automatizada pela interface. Numa delas o usuário que deseja construir apenas uma rota seleciona os funcionários que devem ser atendidos e ativa a construção de uma rota. Na outra, o usuário ativa a construção de rotas para todos os funcionários que não estão alocados a alguma rota. O primeiro modo de construção automatizada, que constrói apenas uma rota, faz a interface disparar o algoritmo descrito no Capítulo 3. Já no segundo modo, a interface executa o algoritmo descrito no Capítulo 4. Depois de construída, a rota deve ser alocada a uma empresa, e então salva no sistema. É importante ressaltarmos aqui que as heurísticas trabalham sobre um *grafo de roteamento* o qual é um grafo completo que é extraído do mapa real através de uma transformação que considera os caminhos mínimos entre todas as paradas e locais onde os funcionários residem. Cada vez que uma heurística é acionada, então, o mapa das vias reais é transformado num grafo de roteamento, que é a entrada dos algoritmos. E, o resultado encontrado pelos algoritmos é transformado novamente para os caminhos nos mapas das vias – o que gerou algumas situações indesejadas, as quais comentamos na Seção 5.7.1

- Alteração Manual de rotas: para a alteração manual das rotas que já foram construídas, o primeiro passo do usuário é entrar no modo de edição de rotas e escolher a rota que será editada. A rota é então mostrada na interface, e o usuário pode selecionar os segmentos da rota que serão excluídos da mesma, e pode adicionar outros segmentos a ela, selecionando-os na ordem correta.

Depois de construir rotas e alocá-las a empresas o usuário pode imprimir a lista de rotas que cada funcionário deve pegar, a parada onde o usuário o deve pegar, e a empresa que passará por essa parada, e a que horas o funcionário deve estar no ponto (a janela de tempo).

5.7 Instâncias Reais

A aquisição de dados foi a atividade mais penosa deste trabalho. Nesta Seção descrevemos os tipos de dados necessários para o sistema e o processo de aquisição dos mesmos.

O sistema de roteamento exige pelo menos três tipos de dados: (1) dados de mapa digital – que compreendem dados dos eixos-de-ruas e comprimento dos eixos, mão dos eixos-de-ruas e numeração predial; (2) dados da localização dos pontos de ônibus do sistema urbano e (3) dados dos funcionários cadastrados no sistema de fretados.[4]

[4]Além dos dados dos eixos de ruas, dos pontos de ônibus e dos funcionários, nosso sistema também necessita dos dados das rotas e das empresas cadastrados, por exemplo.

Os dados de mapa digital e as localizações dos pontos de ônibus do sistema urbano que usamos no setup final do sistema foram cedidas pela Empresa Municipal de Desenvolvimento de Campinas (EMDEC). Na EMDEC usa-se o software MapInfo para gerenciar as rotas de ônibus do transporte público de campinas. Já os dados dos funcionários foram cedidos da Base de Dados da UNITRANSP.[5]

O primeiro passo para se criar uma base digital com a qual se possa trabalhar num sistema de roteamento é extrair os dados e convertê-los para um formato padronizado, no qual os segmentos se conectem logicamente – e não apenas visualmente, como é usual nos sistemas de mapas que não são usados para este fim. Optamos em converter os dados dos formatos proprietários para um formato simplificado a fim de que pudéssemos trabalhar com mais liberdade neste projeto.

Os sistemas de informações geográficas (SIGs) comerciais em que trabalhamos permitem a execução de *script*s em Python, e os usamos a fim de extrair e padronizra os dados. Criamos *script*s apenas para extração dos dados no formato ArcGis. Há uma função do MapInfo que converte dados de MapInfo para ArcGis, característica que possibilita o uso dos mesmos *script*s para a extração de dados nesse outro formato. Dados do tipo (1) e (2) – logo que extraídos da base digital proprietária – estão organizados em *layer*s. Cada item geográfico dentro desses *layer*s tem anexado a ele sua latitude e longitude – se o item for representado por um ponto – ou latitudes e longitudes das poli-linhas – no caso do item representar eixos de ruas. Dados de mão-de-ruas são representados como pequenas setas localizadas geograficamente sobre os eixos de ruas. Essas setas tem, além da informação sobre o posicionamento geográfico, a sua inclinação com relação ao Norte. Uma vez que os dados contem informações de latitude e longitude, eles são facilmente interpretados e reconhecidos por olhos humanos quando plotados em tela – por exemplo, uma seta inclinada na horizontal, apontando para direita, plotada sobre um eixo de rua também na horizontal nos informa claramente que aquele eixo de rua tem mão da-esquerda-para-direita. Contudo, apesar dessa ligação visual existente entre os itens do mapa, ainda precisamos fazer com que um programa de computador entenda as relações entre eles.

A base de dados de funcionários, por sua vez, não possui a localização geográfica – latitude e longitude – nem da residência, nem do local de trabalho de cada um deles, mas possui o endereço (nome da rua e número) desses locais.

Na etapa de cruzamento de informações dos dados textuais dos funcionários e dos dados do mapa digital, utilizamos algoritmos da geometria computacional para criar as ligações lógicas que necessitávamos. Criamos programas para relacionar (A) os eixos-de-ruas entre si, formando um grafo conexo; (B) as setas de sentido da mão com os

[5]Também tivemos os dados de mapa digital em formato ArcGis na Prefeitura da Unicamp, mas não obtivemos êxito em transformar estes dados de mapa digital num mapa que pudesse ser usado por nossos algoritmos.

eixos-de-ruas; (C) a numeração predial com os eixos-de-ruas; e, (D) os pontos-de-ônibus com os eixos de ruas. Note que os programas (C) e (D) necessitam devolver, inclusive, de que lado do eixo-de-rua se encontra o número de prédio ou o ponto de ônibus (se o item fica à direita ou à esquerda do eixo de rua).

Criar as ligações dos eixos-de-ruas com os números das construções foi necessário, pois nossa base de dados de funcionários não tem a latitude e longitude em que o funcionário mora, mas sim, o endereço por extenso. Com o nome da rua e o número da casa onde reside o funcionário nós podemos colocá-lo no mapa automaticamente. Como são mais de 2000 funcionários esse a automação se torna interessante.

Contudo, o posicionamento automático de funcionários no mapa digital foi complicado. Os nomes das ruas do mapa digital e da base de funcionários não correspondiam em todos os casos. A ordem diferente das palavras que formam o nome das vias ou a grafia incorreta das mesmas eram problemas frequentes. Um exemplo desses casos é a presença da via "ADILHO GONCALVES DE OLIVEIRA" no cadastro de funcionários, enquanto que no mapa digital a mesma via era chamada de "ADILIO DE OLIVEIRA GONCALVES".

Criamos então outro programa (E) para efetuar um casamento aproximado dos nomes das vias das duas fontes de dados. Esse programa é semi-automatizado: para os nomes que se alinham perfeitamente (todas as letras) o problema já fica resolvido, contudo, se os nomes não são perfeitamente alinhados, o programa dá como sugestão para o alinhamento os tres nome que mais se aproximam do requisitado, utilizando uma heurística de distância de edição – e a intervenção humana decide qual deve ser o escolhido. Outros métodos, exatos e heurísticos para distância de edição foram estudados [10, 34, 55]. Decidiu-se implementar o algoritmo de programação dinâmica tradicional, com algumas modificações inspiradas nesses artigos.

Por fim, dados das rotas que estão sendo usadas atualmente no sistema de roteamento da UNITRANSP foram adquiridos, com um novo programa de entrada de dados que apresenta o mapa digital já convertido, onde o usuário pode selecionar as vias em que a rota passa. Estas rotas foram desenhadas no nosso software pelos funcionários da UNITRANSP uma vez que elas não estavam no formato digital, e foram levantadas com o intuito de comparação entre a qualidade das rotas atuais e a dos resultados a serem obtidos pelo sistema automatizado.

5.7.1 Testes Preliminares com os Dados Levantados

Nesta Seção descrevemos os testes realizados com os dados reais que levantamos. Primeiro validamos visualmente o mapa digital levantado. Depois, os dados de rotas entradas pela equipe da UNITRANSP. Por fim, criamos algumas instâncias simples de teste para que

pudéssemos avaliar o comportamento das heurísticas quando os caminhos de roteamento obtidos são transformados em caminhos nos mapas de ruas.

Validação Visual dos Dados Obtidos

Verificamos que os mapas cedidos pela EMDEC são os mais completos por desenharem vias que não tínhamos nos outros mapas testados. Entretanto, apesar de completo eles não refletem perfeitamente a realidade. Deixamos de considerar os dados de sentidos das vias, uma vez que estes dados geravam inconsistências, pois levavam a *sorvedouros* (locais onde o ônibus pode entrar mas não pode sair) e a *fontes* (locais dos quais o ônibus só pode sair mas não pode mais voltar) no grafo de roteamento.

Uma vez que estes dados podem ser modificados pelo engenheiro de rotas na interface, quando existir a demanda, deixamos que o usuário indique os sentidos das ruas que forem de conhecimento dele, corrigindo o sentido nas vias quando na encontrarem anormalidades nas rotas geradas.

Note que esta correção de sentidos permitidos das vias é de extrema importância. Algumas das rotas geradas nos testes seguiam pelo sentido errado de ruas que sabemos ser de uma mão apenas, descrevendo curvas que não poderiam ser executadas.

De acordo com Jiang e colaboradores [30], "Um banco de dados criado especialmente para navegação é essencial para as funções-chave dos sistemas de navegação de veículos, por exemplo, criar rotas e guiar o motorista. Existem muitos desafios teóricos em gerar uma base de dados satisfatória". Há várias restrições de tráfego nas redes de eixos-de-ruas e não se pode ignorá-las. Segundo os mesmos autores, "ignorar restrições de curvas proibidas na modelagem da rede de tráfego pode gerar a perda de características essenciais da rede e levar a caminhos sub-otimais ou ilógicos. A magnitude dos atrasos gerado pode ser de 17 a 35% quando comparada ao tempo total da rota.

Por esse motivo estudamos, também, modelos de dados e algoritmos para tratar restrições de curvas proibidas com auxílio dos trabalhos [24, 30] e [60], mas o trabalho de implementação das soluções sugeridas seria impraticável neste trabalho.

Comportamento dos Algoritmos em Instâncias Simples de Testes

Criamos instâncias simples de testes, a fim de validar o comportamento dos métodos otimizadores. Durante estes testes, verificamos um problema que nomeamos o Problema das "Pontas" nas rotas criadas.

Ele acontece em decorrência do nosso método de *transformação* do mapa de ruas real para o grafo de roteamento. Nesta transformação, usamos os caminhos de mínima distância entre os pontos de interesse no mapa real para criar as conexões no grafo de roteamento. Acontece que, quando transformamos as soluções encontradas pelo algoritmo

para soluções candidatas no mapa real podem aparecer caminhos nos quais o ônibus precisa entrar numa rua, pegar o funcionário e voltar pela mesma rua.

Sabemos que o tamanho de um ônibus (veículo geralmente usado para o transporte fretado em questão) não permite este tipo de manobra. O ônibus não pode pegar um funcionário no meio da quadra e voltar pelo mesmo caminho que chegou até lá apesar de ambos os sentidos de ruas serem permitidos.

Este é o problema é evidenciado quando plotamos as rotas resultantes do algoritmo no mapa digital, como na Figura 5.2. Aí vemos que a rota formada pode não ser viável para um ônibus e que a rota tem um pedaço semelhante a uma "ponta". Atualmente, este problema é resolvido mecanicamente pelo usuário. Fazendo uso da interface gráfica, ele apaga as arestas da parte problemática da rota e a redesenha – veja a Figura 5.3. Resolver este problema algoritmicamente se torna um bom exercício futuro.

Testes precisam ser realizados a fim de comparar as rotas obtidas pelo sistema com as rotas em funcionamento. Os dados de localização das residências de funcionários precisam ser ajustados por um operador humano, uma vez que foram plotados automaticamente por um algoritmo que faz uso dos números de ruas do mapa digitalizado que são apenas uma aproximação dos números reais das residências.

5.8 Conclusões e Trabalhos Futuros

Neste capítulo descrevemos o problema de transporte de funcionários enfrentado pela UNICAMP, o Problema do Transporte Fretado dos Servidores da UNICAMP (PTFSU), e apresentamos uma abordagem interativa de resolução que une o poder de processamento do computador na forma de algoritmos otimizadores com o poder de decisão e detecção de falhas do usuário humano.

Instâncias do problema real foram criadas através da obtenção e preparação de um mapa digital de vias; da localização automatizada dos funcionários e da obtenção das rotas em uso, para que pudéssemos criar um conjunto de instâncias e resultados de referência.

Estas instâncias necessitam ainda de ajustes:

- o sentido das vias precisa ser atualizado (para evitar o problema das curvas impossíveis)
- os locais das residências dos funcionários deve ser corrigido por um operador humano.

Entretando, já podemos usar o sistema para obter dados bastante aproximados da realidade, mesmo sem estes ajustes.

Figura 5.2: Ilustração do problema da "curva impossível".

Figura 5.3: Uma possível solução do problema da “curva impossível”.

Uma atividade futura que beneficiaria sobremaneira o trabalho do usuário do sistema é a criação um algorimo que considere os expedientes de trabalho, a variante de *pick up and delivery* do problema e projete rotas utilizando frota heterogênea de veículos.

Outra é o gerenciamento do problema das pontas de forma automatizada. Assim o trabalho do engenheiro de rotas da UNITRANSP seria ainda mais facilitado.

Capítulo 6

Conclusões

A principal motivação para esta pesquisa foi uma aplicação prática referente ao problema de roteamento dos ônibus fretados pela UNICAMP para o transporte de seus funcionários, que se caracteriza como um modelo *"rico"*.

Nos capítulos anteriores apresentamos a nossa solução para o problema de roteamento que é enfrentado pela UNICAMP no transporte de seus funcionários das suas residência aos seus respectivos locais de trabalho.

Nosso objetivo final foi a otimização de tal processo através da minimização da distância total percorrida e do número de veículos empregados, almejando a consequente redução dos gastos incorridos pela Universidade.

Em cada um dos três capítulos anteriores apresentamos um problema de forma isolada: o TPP no Capítulo 3, o CmTPP no Capítulo 4 e o Problema enfrentado pela UNICAMP no Capítulo 5. Ao final de cada um deles apresentamos nossas conclusões sobre o problema e possíveis extensões a serem trabalhadas.

As nossas principais contribuições desta dissertação são:

- estudo e levantamento dos dados do problema prático de transporte de funcionários da UNICAMP;
- projeto e implementação de um algoritmo heurístico para o TPP, superior aos encontrados atualmente na literatura, até onde a conhecemos;
- proposta de um modelo rico de roteamento, o CmTPP, que provê uma abordagem unificada para a solução de outros problemas da literatura, em conjunto com um algoritmo heurístico para a resolução deste modelo;
- a construção de uma interface gráfica que possibilita o uso dos algoritmos projetados no ambiente da UNITRANSP.

Sugestões para extensões e trabalhos futuros foram apresentadas ao final dos Capítulos de 3 a 5.

Apêndice A

Resultados Detalhados do Capítulo 3

Seguem as tabelas de resultados detalhados para cada uma das instâncias simétricas não capacitadas da TPPlib. As Tabelas A.1 e A.2 mostram os resultados das instâncias cujo valor ótimo é conhecido. As linhas em negrito indicam as instâncias em que as 10 execuções independentes da nossa heurística específica para o problema simétrico não capacitado encontrou sempre a solução ótima. A Tabela 3.4 lista os resultados obtidos nas instâncias cujo ótimo não foi provado na literatura. Vale a pena notar que, apesar do trabalho de Bontoux e Feillet [8] indicarem uma solução de valor 1125 para a instância 350.50.5, encontramos que a solução ótima para esta instância tem o valor 1225, em um trabalho conjunto com os Profs. Drs. Luidi Simonetti e Yuri Frota ainda não publicado.

M	P	I	Benchmark	Avg. H8	Best H8
50	**50**	**1**	**1856**	**1856,0**	**1856**
50	**50**	**2**	**1070**	**1070,0**	**1070**
50	**50**	**3**	**1553**	**1553,0**	**1553**
50	**50**	**4**	**1394**	**1394,0**	**1394**
50	**50**	**5**	**1536**	**1536,0**	**1536**
50	**100**	**1**	**2397**	**2397,0**	**2397**
50	**100**	**2**	**2138**	**2138,0**	**2138**
50	**100**	**3**	**1852**	**1852,0**	**1852**
50	**100**	**4**	**3093**	**3093,0**	**3093**
50	**100**	**5**	**2603**	**2603,0**	**2603**
50	**150**	**1**	**2784**	**2784,0**	**2784**
50	**150**	**2**	**2137**	**2137,0**	**2137**
50	**150**	**3**	**2308**	**2308,0**	**2308**
50	**150**	**4**	**2524**	**2524,0**	**2524**
50	**150**	**5**	**3150**	3150,5	**3150**
50	**200**	**1**	**3354**	**3354,0**	**3354**
50	**200**	**2**	**2397**	**2397,0**	**2397**
50	**200**	**3**	**2319**	**2319,0**	**2319**
50	**200**	**4**	**2858**	**2858,0**	**2858**
50	**200**	**5**	**3575**	**3575,0**	**3575**
100	**50**	**1**	**1468**	**1468,0**	**1468**
100	**50**	**2**	**971**	**971,0**	**971**
100	**50**	**3**	**1623**	**1623,0**	**1623**
100	**50**	**4**	**1718**	**1718,0**	**1718**
100	**50**	**5**	**2494**	**2494,0**	**2494**
100	**100**	**1**	**2121**	**2121,0**	**2121**
100	**100**	**2**	**1906**	**1906,0**	**1906**
100	**100**	**3**	**1822**	**1822,0**	**1822**
100	**100**	**4**	**1649**	**1649,0**	**1649**
100	**100**	**5**	**2925**	**2925,0**	**2925**
100	**150**	**1**	**2195**	**2195,0**	**2195**
100	**150**	**2**	**2806**	**2806,0**	**2806**
100	**150**	**3**	**2257**	**2257,0**	**2257**
100	**150**	**4**	**2625**	**2625,0**	**2625**
100	**150**	**5**	**3150**	**3150,0**	**3150**
100	**200**	**1**	**1883**	**1883,0**	**1883**
100	**200**	**2**	**3077**	3077,9	**3077**
100	**200**	**3**	**2791**	**2791,0**	**2791**
100	**200**	**4**	**3409**	3409,2	**3409**
100	**200**	**5**	**2732**	**2732,0**	**2732**

Tabela A.1: Resultados detalhados da H8 para as Instâncias Simétricas sem Restrições de Capacidade

M	P	I	Benchmark	Avg. H8	Best H8
150	**50**	**1**	**1658**	**1658,0**	**1658**
150	**50**	**2**	**1383**	**1383,0**	**1383**
150	**50**	**3**	**821**	**821,0**	**821**
150	**50**	**4**	**1676**	**1676,0**	**1676**
150	**50**	**5**	**1823**	**1823,0**	**1823**
150	**100**	**1**	**1717**	**1717,0**	**1717**
150	**100**	**2**	**1798**	**1798,0**	**1798**
150	**100**	**3**	**1959**	1959,4	**1959**
150	**100**	**4**	**1609**	**1609,0**	**1609**
150	**100**	**5**	**1585**	**1585,0**	**1585**
150	**150**	**1**	**1669**	**1669,0**	**1669**
150	**150**	**2**	**2526**	**2526,0**	**2526**
150	**150**	**3**	**2456**	2460,5	**2456**
150	**150**	**4**	**1761**	**1761,0**	**1761**
150	**150**	**5**	**2355**	**2355,0**	**2355**
150	**200**	**1**	**1760**	**1760,0**	**1760**
150	**200**	**2**	**2312**	**2312,0**	**2312**
150	**200**	**3**	**2594**	2615,2	**2594**
150	**200**	**4**	**1889**	**1889,0**	**1889**
150	**200**	**5**	**2472**	**2472,0**	**2472**
200	**50**	**1**	**1102**	**1102,0**	**1102**
200	**50**	**2**	**607**	**607,0**	**607**
200	**50**	**3**	**530**	**530,0**	**530**
200	**50**	**4**	**908**	**908,0**	**908**
200	**50**	**5**	**1067**	**1067,0**	**1067**
200	**100**	**1**	**949**	**949,0**	**949**
200	**100**	**2**	**2271**	2278,3	**2271**
200	**100**	**3**	**1611**	**1611,0**	**1611**
200	**100**	**4**	**1799**	**1799,0**	**1799**
200	**100**	**5**	**3161**	3161,4	**3161**
200	**150**	**1**	**1730**	**1730,0**	**1730**
200	**150**	**2**	**2745**	**2745,0**	**2745**
200	**150**	**3**	**1861**	**1861,0**	**1861**
200	150	4	2419	2419,2	2419
200	**150**	**5**	**2079**	**2079,0**	**2079**
200	**200**	**1**	**1736**	**1736,0**	**1736**
200	**200**	**2**	**2352**	**2352,0**	**2352**
200	**200**	**3**	**2505**	**2505,0**	**2505**
200	200	4	2344	2344,4	2344
200	**200**	**5**	**2427**	**2427,0**	**2427**

Tabela A.2: Resultados detalhados da H8 para as Instâncias Simétricas sem Restrições de Capacidade (continuação)

M	**P**	**L**	**I**	**Benchmark**	**H8**	**Gap %**	**T(s)**
50	50	1	1	46794	46813	0.04	0.28
50	50	1	2	48120	48130	0.02	0.71
50	50	1	3	42861	42861	0.00	0.4
50	50	1	4	42920	42920	0.00	0.39
50	50	1	5	40562	40562	0.00	0.31
50	50	5	1	17980	17980	0.00	0.31
50	50	5	2	18449	18459	0.05	0.26
50	50	5	3	16653	16653	0.00	0.35
50	50	5	4	16196	16196	0.00	0.45
50	50	5	5	15844	15907	0.40	0.26
50	50	7	1	9013	9013	0.00	0.27
50	50	7	2	9484	9756	2.87	0.22
50	50	7	3	8752	8783	0.35	0.17
50	50	7	4	8635	8901	3.08	0.09
50	50	7	5	8620	8675	0.64	0.2
50	50	8	1	6287	6374	1.38	0.12
50	50	8	2	6258	6273	0.24	0.29
50	50	8	3	5846	5846	0.00	0.11
50	50	8	4	5573	5573	0.00	0.08
50	50	8	5	5827	5827	0.00	0.03
50	50	9	1	3650	3650	0.00	0.01
50	50	9	2	3811	3811	0.00	0.04
50	50	9	3	3522	3628	3.01	0.04
50	50	9	4	3329	3329	0.00	0.03
50	50	9	5	3543	3560	0.48	0.04
50	50	95	1	2636	2636	0.00	0.03
50	50	95	2	1758	1758	0.00	0.0
50	50	95	3	2390	2519	5.40	0.05
50	50	95	4	2643	2643	0.00	0.11
50	50	95	5	2792	2823	1.11	0.02
50	50	99	1	1703	1703	0.00	0.02
50	50	99	2	1305	1305	0.00	0.01
50	50	99	3	1893	1893	0.00	0.01
50	50	99	4	2143	2143	0.00	0.03
50	50	99	5	2266	2266	0.00	0.0

Tabela A.3: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 50 mercados e 50 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
50	100	1	1	83044	83044	0.00	0.06
50	100	1	2	87427	87437	0.01	0.67
50	100	1	3	85002	85002	0.00	0.03
50	100	1	4	89756	89756	0.00	0.06
50	100	1	5	87090	87118	0.03	0.17
50	100	5	1	29334	29338	0.01	0.16
50	100	5	2	30089	30099	0.03	0.33
50	100	5	3	29016	29016	0.00	0.45
50	100	5	4	29913	29913	0.00	0.19
50	100	5	5	29503	29550	0.16	0.03
50	100	7	1	14066	14323	1.83	0.47
50	100	7	2	13536	13554	0.13	0.28
50	100	7	3	12934	12934	0.00	0.35
50	100	7	4	13475	13881	3.01	0.27
50	100	7	5	13686	13700	0.10	0.24
50	100	8	1	8441	8547	1.26	0.1
50	100	8	2	7940	8138	2.49	0.27
50	100	8	3	7831	7831	0.00	0.35
50	100	8	4	8230	8230	0.00	0.09
50	100	8	5	8520	8565	0.53	0.27
50	100	9	1	4718	4718	0.00	0.01
50	100	9	2	4705	4870	3.51	0.04
50	100	9	3	4462	4505	0.96	0.11
50	100	9	4	4620	4620	0.00	0.13
50	100	9	5	4834	4834	0.00	0.19
50	100	95	1	3429	3429	0.00	0.07
50	100	95	2	3226	3226	0.00	0.08
50	100	95	3	2818	2818	0.00	0.05
50	100	95	4	3190	3381	5.99	0.1
50	100	95	5	3273	3273	0.00	0.03
50	100	99	1	2389	2389	0.00	0.02
50	100	99	2	1986	1986	0.00	0.0
50	100	99	3	2091	2091	0.00	0.02
50	100	99	4	2519	2519	0.00	0.02
50	100	99	5	2580	2580	0.00	0.01

Tabela A.4: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 50 mercados e 100 produtos

M	**P**	**L**	**I**	**Benchmark**	**H8**	**Gap %**	**T(s)**
50	150	1	1	126600	126604	0.00	0.42
50	150	1	2	128380	128380	0.00	0.47
50	150	1	3	127979	127979	0.00	0.09
50	150	1	4	130088	130088	0.00	0.58
50	150	1	5	126927	126939	0.01	0.29
50	150	5	1	42092	42096	0.01	0.07
50	150	5	2	42006	42006	0.00	0.73
50	150	5	3	41891	41891	0.00	0.58
50	150	5	4	42116	42116	0.00	0.18
50	150	5	5	41300	41300	0.00	0.05
50	150	7	1	18497	18506	0.05	0.27
50	150	7	2	17710	17758	0.27	0.06
50	150	7	3	17665	17665	0.00	0.46
50	150	7	4	17964	17964	0.00	0.33
50	150	7	5	18236	18267	0.17	0.43
50	150	8	1	11088	11211	1.11	0.38
50	150	8	2	10188	10374	1.83	0.17
50	150	8	3	10565	10573	0.08	0.18
50	150	8	4	10573	10573	0.00	0.47
50	150	8	5	10986	11026	0.36	0.23
50	150	9	1	5789	5792	0.05	0.27
50	150	9	2	5381	5381	0.00	0.12
50	150	9	3	5479	5479	0.00	0.37
50	150	9	4	5543	5567	0.43	0.15
50	150	9	5	5979	6100	2.02	0.25
50	150	95	1	4157	4157	0.00	0.06
50	150	95	2	3808	3808	0.00	0.27
50	150	95	3	3588	3724	3.79	0.17
50	150	95	4	3681	3695	0.38	0.21
50	150	95	5	4044	4141	2.40	0.17
50	150	99	1	3239	3239	0.00	0.05
50	150	99	2	2761	2761	0.00	0.08
50	150	99	3	2012	2012	0.00	0.01
50	150	99	4	2532	2532	0.00	0.01
50	150	99	5	3240	3240	0.00	0.0

Tabela A.5: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 50 mercados e 150 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
50	200	1	1	163462	163485	0.01	0.14
50	200	1	2	167145	167155	0.01	0.89
50	200	1	3	173014	173014	0.00	0.31
50	200	1	4	174836	174836	0.00	0.24
50	200	1	5	168189	168189	0.00	0.65
50	200	5	1	53186	53205	0.04	0.38
50	200	5	2	53628	53630	0.00	0.4
50	200	5	3	54911	54940	0.05	0.23
50	200	5	4	55905	55905	0.00	0.36
50	200	5	5	53661	53661	0.00	0.25
50	200	7	1	22626	22645	0.08	0.65
50	200	7	2	21912	21915	0.01	0.78
50	200	7	3	22386	22386	0.00	0.09
50	200	7	4	22927	22927	0.00	0.34
50	200	7	5	22594	22594	0.00	0.74
50	200	8	1	12519	12538	0.15	0.42
50	200	8	2	12108	12108	0.00	0.54
50	200	8	3	12431	12591	1.29	0.17
50	200	8	4	12717	12717	0.00	0.58
50	200	8	5	12612	12718	0.84	0.43
50	200	9	1	6523	6564	0.63	0.26
50	200	9	2	6213	6393	2.90	0.17
50	200	9	3	6241	6267	0.42	0.3
50	200	9	4	6444	6466	0.34	0.43
50	200	9	5	6534	6563	0.44	0.18
50	200	95	1	4306	4361	1.28	0.11
50	200	95	2	4553	4553	0.00	0.26
50	200	95	3	4005	4013	0.20	0.24
50	200	95	4	4072	4091	0.47	0.19
50	200	95	5	4805	4805	0.00	0.16
50	200	99	1	2414	2414	0.00	0.06
50	200	99	2	2770	2770	0.00	0.0
50	200	99	3	2440	2440	0.00	0.01
50	200	99	4	2627	2632	0.19	0.01
50	200	99	5	3704	3782	2.11	0.15

Tabela A.6: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 50 mercados e 200 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
100	50	1	1	93291	93436	0.16	6.01
100	50	1	2	93710	93812	0.11	5.7
100	50	1	3	86395	86414	0.02	7.44
100	50	1	4	92922	92951	0.03	7.29
100	50	1	5	81486	81511	0.03	1.89
100	50	5	1	33174	33277	0.31	10.07
100	50	5	2	33050	33223	0.52	6.09
100	50	5	3	30223	30291	0.22	4.23
100	50	5	4	32249	32353	0.32	7.57
100	50	5	5	28275	28311	0.13	6.38
100	50	7	1	15801	16244	2.80	3.4
100	50	7	2	15665	15714	0.31	2.38
100	50	7	3	14358	14466	0.75	2.79
100	50	7	4	15418	15564	0.95	4.8
100	50	7	5	13746	13809	0.46	3.88
100	50	8	1	9856	9982	1.28	2.09
100	50	8	2	9745	9793	0.49	0.82
100	50	8	3	8756	8899	1.63	2.17
100	50	8	4	9574	9719	1.51	0.8
100	50	8	5	8691	9229	6.19	1.59
100	50	9	1	4847	4863	0.33	0.7
100	50	9	2	4672	4761	1.90	0.46
100	50	9	3	4381	4547	3.79	0.6
100	50	9	4	4961	5016	1.11	0.77
100	50	9	5	4607	4612	0.11	0.66
100	50	95	1	2699	2699	0.00	0.17
100	50	95	2	2685	2740	2.05	0.09
100	50	95	3	2649	2649	0.00	0.11
100	50	95	4	3272	3290	0.55	0.18
100	50	95	5	2995	2995	0.00	0.16
100	50	99	1	1313	1313	0.00	0.02
100	50	99	2	1367	1367	0.00	0.01
100	50	99	3	1146	1146	0.00	0.01
100	50	99	4	2217	2219	0.09	0.08
100	50	99	5	1475	1475	0.00	0.01

Tabela A.7: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 100 mercados e 50 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
100	100	1	1	168075	168217	0.08	5.05
100	100	1	2	164651	164748	0.06	7.08
100	100	1	3	160410	160483	0.05	10.38
100	100	1	4	181984	181984	0.00	7.27
100	100	1	5	160015	160142	0.08	7.17
100	100	5	1	54900	54900	0.00	7.57
100	100	5	2	53279	53293	0.03	9.16
100	100	5	3	51671	51865	0.38	4.11
100	100	5	4	58641	58702	0.10	6.52
100	100	5	5	50902	50915	0.03	6.68
100	100	7	1	23250	23317	0.29	2.7
100	100	7	2	21961	21969	0.04	4.09
100	100	7	3	21537	21739	0.94	2.88
100	100	7	4	24555	24742	0.76	3.63
100	100	7	5	22067	22302	1.06	2.48
100	100	8	1	13673	13882	1.53	3.08
100	100	8	2	12877	13065	1.46	3.01
100	100	8	3	12642	12751	0.86	4.39
100	100	8	4	14087	14179	0.65	4.38
100	100	8	5	12854	12854	0.00	0.2
100	100	9	1	6494	6708	3.30	0.9
100	100	9	2	6375	6948	8.99	0.31
100	100	9	3	6063	6167	1.72	1.34
100	100	9	4	6863	6986	1.79	0.65
100	100	9	5	6415	6491	1.18	1.52
100	100	95	1	3484	3518	0.98	0.06
100	100	95	2	3468	3468	0.00	0.61
100	100	95	3	3491	3544	1.52	0.56
100	100	95	4	3532	3557	0.71	0.26
100	100	95	5	3802	3870	1.79	0.73
100	100	99	1	1539	1539	0.00	0.09
100	100	99	2	1554	1554	0.00	0.04
100	100	99	3	1672	1672	0.00	0.02
100	100	99	4	2168	2199	1.43	0.13
100	100	99	5	2394	2394	0.00	0.02

Tabela A.8: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 100 mercados e 100 produtos

M	**P**	**L**	**I**	**Benchmark**	**H8**	**Gap %**	**T(s)**
100	150	1	1	254921	255010	0.03	9.55
100	150	1	2	249585	249680	0.04	6.01
100	150	1	3	247800	247938	0.06	8.89
100	150	1	4	261499	261534	0.01	9.32
100	150	1	5	238058	238058	0.00	7.26
100	150	5	1	80460	80538	0.10	6.79
100	150	5	2	78721	78779	0.07	3.01
100	150	5	3	77133	77287	0.20	6.2
100	150	5	4	81415	81503	0.11	6.05
100	150	5	5	73903	74090	0.25	3.54
100	150	7	1	31977	32124	0.46	5.61
100	150	7	2	31133	31187	0.17	9.9
100	150	7	3	30574	30767	0.63	3.96
100	150	7	4	32307	32436	0.40	5.58
100	150	7	5	29984	30149	0.55	3.78
100	150	8	1	17320	17400	0.46	5.49
100	150	8	2	16500	16670	1.03	3.87
100	150	8	3	16387	16422	0.21	4.9
100	150	8	4	17202	17250	0.28	3.46
100	150	8	5	16458	16496	0.23	2.73
100	150	9	1	7358	7382	0.33	1.07
100	150	9	2	7342	7478	1.85	0.23
100	150	9	3	6962	6962	0.00	2.87
100	150	9	4	7960	8155	2.45	2.05
100	150	9	5	7766	8108	4.40	1.81
100	150	95	1	4436	4523	1.96	0.51
100	150	95	2	4409	4509	2.27	0.7
100	150	95	3	4068	4193	3.07	0.69
100	150	95	4	4762	4814	1.09	1.65
100	150	95	5	4663	4786	2.64	0.24
100	150	99	1	1978	2000	1.11	0.06
100	150	99	2	2212	2212	0.00	0.09
100	150	99	3	2122	2133	0.52	0.07
100	150	99	4	2336	2336	0.00	0.05
100	150	99	5	2468	2468	0.00	0.02

Tabela A.9: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 100 mercados e 150 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
100	200	1	1	339322	339438	0.03	7.19
100	200	1	2	328970	328970	0.00	5.7
100	200	1	3	320623	320722	0.03	8.64
100	200	1	4	331657	331657	0.00	5.8
100	200	1	5	311129	311218	0.03	2.98
100	200	5	1	105300	105431	0.12	10.14
100	200	5	2	101942	102016	0.07	7.17
100	200	5	3	97956	98117	0.16	2.48
100	200	5	4	102405	102507	0.10	5.97
100	200	5	5	95206	95327	0.13	5.47
100	200	7	1	40403	40505	0.25	6.64
100	200	7	2	38937	38974	0.10	8.9
100	200	7	3	37685	37723	0.10	10.09
100	200	7	4	39483	39770	0.73	11.52
100	200	7	5	36897	36910	0.04	3.82
100	200	8	1	20659	20848	0.91	8.6
100	200	8	2	19918	19986	0.34	8.08
100	200	8	3	19273	19471	1.03	6.39
100	200	8	4	19898	20179	1.41	1.97
100	200	8	5	19521	19820	1.53	5.67
100	200	9	1	8544	8678	1.57	2.4
100	200	9	2	8830	9235	4.59	1.53
100	200	9	3	8342	8466	1.49	2.37
100	200	9	4	8485	8673	2.22	1.71
100	200	9	5	8745	8920	2.00	2.77
100	200	95	1	4733	4915	3.85	1.27
100	200	95	2	4658	4679	0.45	0.55
100	200	95	3	4333	4411	1.80	1.7
100	200	95	4	5096	5096	0.00	1.66
100	200	95	5	5115	5274	3.11	1.12
100	200	99	1	2508	2508	0.00	0.23
100	200	99	2	2901	2902	0.03	0.09
100	200	99	3	2267	2267	0.00	0.24
100	200	99	4	2987	2987	0.00	0.44
100	200	99	5	2685	2750	2.42	0.1

Tabela A.10: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 100 mercados e 200 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
150	50	1	1	125418	125550	0.11	44.79
150	50	1	2	125650	125751	0.08	27.94
150	50	1	3	143310	143471	0.11	42.76
150	50	1	4	135503	135696	0.14	30.11
150	50	1	5	126023	126118	0.08	31.34
150	50	5	1	43115	43259	0.33	28.4
150	50	5	2	42669	42940	0.64	42.11
150	50	5	3	47455	47571	0.24	47.99
150	50	5	4	45556	45784	0.50	22.75
150	50	5	5	41937	41987	0.12	19.89
150	50	7	1	20264	20851	2.90	17.73
150	50	7	2	19621	19928	1.56	27.26
150	50	7	3	20671	20738	0.32	15.83
150	50	7	4	20472	20678	1.01	15.31
150	50	7	5	19554	19797	1.24	16.92
150	50	8	1	12448	12744	2.38	5.09
150	50	8	2	11817	12029	1.79	9.04
150	50	8	3	12538	13095	4.44	6.28
150	50	8	4	12622	13188	4.48	9.9
150	50	8	5	11711	12016	2.60	9.42
150	50	9	1	5360	5384	0.45	2.06
150	50	9	2	5571	6349	13.97	1.33
150	50	9	3	6097	7037	15.42	1.89
150	50	9	4	6152	6853	11.39	0.71
150	50	9	5	5886	6409	8.89	3.58
150	50	95	1	2986	3007	0.70	1.03
150	50	95	2	3031	3097	2.18	0.76
150	50	95	3	3214	3386	5.35	0.46
150	50	95	4	3633	3641	0.22	0.73
150	50	95	5	3337	3365	0.84	0.88
150	50	99	1	1894	1894	0.00	0.28
150	50	99	2	1477	1477	0.00	0.12
150	50	99	3	1171	1193	1.88	0.05
150	50	99	4	1345	1345	0.00	0.04
150	50	99	5	1322	1322	0.00	0.03

Tabela A.11: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 150 mercados e 50 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
150	100	1	1	223506	223605	0.04	19.51
150	100	1	2	239549	239698	0.06	21.85
150	100	1	3	263750	263817	0.03	16.15
150	100	1	4	276734	276972	0.09	15.44
150	100	1	5	246932	247133	0.08	42.52
150	100	5	1	72000	72185	0.26	48.2
150	100	5	2	76195	76342	0.19	44.75
150	100	5	3	82063	82309	0.30	29.63
150	100	5	4	86567	86652	0.10	21.12
150	100	5	5	76916	77064	0.19	29.46
150	100	7	1	30171	30262	0.30	26.04
150	100	7	2	31253	31482	0.73	29.09
150	100	7	3	32784	33249	1.42	40.72
150	100	7	4	34829	35020	0.55	19.35
150	100	7	5	31308	31470	0.52	29.54
150	100	8	1	17015	18141	6.62	10.07
150	100	8	2	17357	17794	2.52	10.14
150	100	8	3	17798	18094	1.66	22.44
150	100	8	4	18661	19053	2.10	15.83
150	100	8	5	17407	18010	3.46	11.25
150	100	9	1	7922	8207	3.60	6.52
150	100	9	2	8090	8718	7.76	4.08
150	100	9	3	7988	8361	4.67	7.47
150	100	9	4	8267	8843	6.97	4.89
150	100	9	5	7918	8546	7.93	4.81
150	100	95	1	4456	4518	1.39	1.9
150	100	95	2	4602	4648	1.00	3.0
150	100	95	3	4293	4518	5.24	1.66
150	100	95	4	4655	4783	2.75	1.44
150	100	95	5	4577	5230	14.27	0.56
150	100	99	1	1710	1710	0.00	0.06
150	100	99	2	2222	2222	0.00	0.38
150	100	99	3	1544	1544	0.00	0.24
150	100	99	4	2316	2327	0.47	0.06
150	100	99	5	2140	2140	0.00	0.05

Tabela A.12: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 150 mercados e 100 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
150	150	1	1	344600	344765	0.05	28.23
150	150	1	2	354253	354419	0.05	32.35
150	150	1	3	389949	390185	0.06	40.78
150	150	1	4	404653	404723	0.02	41.05
150	150	1	5	358584	358718	0.04	26.81
150	150	5	1	107462	107646	0.17	50.12
150	150	5	2	109451	109730	0.25	36.93
150	150	5	3	117511	117586	0.06	23.83
150	150	5	4	124302	124444	0.11	23.9
150	150	5	5	108948	109040	0.08	33.78
150	150	7	1	41916	42045	0.31	9.53
150	150	7	2	42012	42200	0.45	19.06
150	150	7	3	44216	44549	0.75	25.81
150	150	7	4	47464	47486	0.05	39.49
150	150	7	5	41680	41856	0.42	30.6
150	150	8	1	21094	21222	0.61	14.8
150	150	8	2	21862	22245	1.75	21.57
150	150	8	3	22499	22653	0.68	33.23
150	150	8	4	23852	24054	0.85	34.47
150	150	8	5	21378	21780	1.88	15.84
150	150	9	1	9518	9797	2.93	11.35
150	150	9	2	9478	9929	4.76	5.28
150	150	9	3	9456	9743	3.04	7.58
150	150	9	4	9609	10194	6.09	7.69
150	150	9	5	9398	9625	2.42	7.75
150	150	95	1	4789	4858	1.44	3.44
150	150	95	2	5099	5342	4.77	1.36
150	150	95	3	5037	5372	6.65	0.97
150	150	95	4	5271	5520	4.72	3.7
150	150	95	5	5662	5862	3.53	3.59
150	150	99	1	1702	1702	0.00	0.09
150	150	99	2	2547	2547	0.00	0.72
150	150	99	3	2596	2596	0.00	1.01
150	150	99	4	2197	2197	0.00	0.83
150	150	99	5	2828	2828	0.00	0.67

Tabela A.13: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 150 mercados e 150 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
150	200	1	1	483201	483226	0.01	26.93
150	200	1	2	468101	468382	0.06	43.42
150	200	1	3	523733	523733	0.00	46.38
150	200	1	4	530511	530606	0.02	40.23
150	200	1	5	499070	499303	0.05	18.59
150	200	5	1	146821	147031	0.14	39.63
150	200	5	2	142406	142537	0.09	35.62
150	200	5	3	156032	156283	0.16	38.97
150	200	5	4	161572	161836	0.16	37.38
150	200	5	5	150057	150192	0.09	15.97
150	200	7	1	54207	54290	0.15	49.74
150	200	7	2	53007	53183	0.33	37.48
150	200	7	3	56585	56781	0.35	48.29
150	200	7	4	59935	60123	0.31	33.56
150	200	7	5	55194	55377	0.33	34.62
150	200	8	1	26334	26441	0.41	44.86
150	200	8	2	25936	26147	0.81	29.59
150	200	8	3	27300	27459	0.58	31.83
150	200	8	4	29012	29114	0.35	18.91
150	200	8	5	27271	27414	0.52	21.92
150	200	9	1	10195	10811	6.04	9.2
150	200	9	2	10553	11219	6.31	9.9
150	200	9	3	10463	10913	4.30	5.93
150	200	9	4	11346	11729	3.38	11.92
150	200	9	5	10875	11070	1.79	13.41
150	200	95	1	5176	5245	1.33	3.45
150	200	95	2	5682	5925	4.28	4.31
150	200	95	3	5354	5371	0.32	5.73
150	200	95	4	5866	6374	8.66	0.46
150	200	95	5	6067	6260	3.18	4.03
150	200	99	1	2073	2075	0.10	1.44
150	200	99	2	2943	3085	4.83	1.18
150	200	99	3	2256	2279	1.02	0.56
150	200	99	4	2574	2574	0.00	0.86
150	200	99	5	2905	2905	0.00	0.77

Tabela A.14: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 150 mercados e 200 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
200	50	1	1	184696	184994	0.16	137.3
200	50	1	2	169938	170077	0.08	90.44
200	50	1	3	195103	195292	0.10	82.56
200	50	1	4	179172	179264	0.05	109.34
200	50	1	5	173355	173560	0.12	82.12
200	50	5	1	61360	61582	0.36	124.84
200	50	5	2	55908	56211	0.54	64.41
200	50	5	3	62520	62646	0.20	138.56
200	50	5	4	58952	59169	0.37	99.53
200	50	5	5	56235	56352	0.21	75.82
200	50	7	1	27212	27888	2.48	79.5
200	50	7	2	24546	25010	1.89	74.19
200	50	7	3	26644	27175	1.99	31.69
200	50	7	4	26090	26337	0.95	64.28
200	50	7	5	24813	25211	1.60	69.14
200	50	8	1	15555	16019	2.98	28.02
200	50	8	2	13874	14686	5.85	22.48
200	50	8	3	15714	16212	3.17	26.62
200	50	8	4	15354	16079	4.72	29.46
200	50	8	5	14749	15219	3.19	35.52
200	50	9	1	7340	7790	6.13	9.93
200	50	9	2	6402	6948	8.53	7.32
200	50	9	3	6805	7615	11.90	5.35
200	50	9	4	7277	8209	12.81	6.48
200	50	9	5	7008	7409	5.72	5.87
200	50	95	1	3963	4092	3.26	1.95
200	50	95	2	3486	3504	0.52	0.68
200	50	95	3	3548	4252	19.84	1.06
200	50	95	4	4111	4209	2.38	2.58
200	50	95	5	3806	3915	2.86	2.1
200	50	99	1	1425	1425	0.00	0.28
200	50	99	2	1137	1183	4.05	0.07
200	50	99	3	1446	1446	0.00	0.12
200	50	99	4	2237	2237	0.00	0.37
200	50	99	5	2351	2355	0.17	0.76

Tabela A.15: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 200 mercados e 50 produtos

M	P	L	I	Benchmark	H8	Gap %	T(s)
200	100	1	1	328659	329048	0.12	122.35
200	100	1	2	345088	345300	0.06	79.77
200	100	1	3	344957	345254	0.09	41.61
200	100	1	4	347295	347387	0.03	105.57
200	100	1	5	316743	316967	0.07	75.81
200	100	5	1	102850	103106	0.25	79.32
200	100	5	2	107166	107379	0.20	120.23
200	100	5	3	105480	105664	0.17	115.11
200	100	5	4	100.52	107831	–	–
200	100	5	5	97634	97853	0.22	99.12
200	100	7	1	40444	40612	0.42	147.72
200	100	7	2	41831	42115	0.68	118.15
200	100	7	3	40733	40850	0.29	60.93
200	100	7	4	106.7	42754	–	–
200	100	7	5	97.72	38784	–	–
200	100	8	1	21655	21869	0.99	88.66
200	100	8	2	21527	21910	1.78	51.06
200	100	8	3	21614	22015	1.86	55.72
200	100	8	4	22537	22755	0.97	69.0
200	100	8	5	20970	21150	0.86	69.53
200	100	9	1	9262	9932	7.23	16.06
200	100	9	2	9042	9267	2.49	13.89
200	100	9	3	9418	9819	4.26	16.38
200	100	9	4	9753	10442	7.06	6.53
200	100	9	5	9506	9902	4.17	11.29
200	100	95	1	4805	4936	2.73	3.89
200	100	95	2	4867	5085	4.48	6.24
200	100	95	3	4764	5069	6.40	2.69
200	100	95	4	5289	5584	5.58	3.69
200	100	95	5	5241	5366	2.39	6.37
200	100	99	1	2039	2039	0.00	0.66
200	100	99	2	2756	2756	0.00	2.08
200	100	99	3	2239	2273	1.52	0.66
200	100	99	4	2548	2548	0.00	1.6
200	100	99	5	1867	1867	0.00	0.08

Tabela A.16: Resultados detalhados da H8 para as Instâncias Simétricas com Restrições de Capacidade com 200 mercados e 100 produtos

Apêndice B

Descrição de alguns problemas clássicos de otimização citados no texto

Neste Capítulo apresentamos uma descrição sucinta dos problemas citados no texto da dissertação.

B.1 Problema do Caixeiro Viajante

Dado um conjunto de cidades e o custo de viagem entre cada par delas, o Problema do Caixeiro Viajante (TSP) consiste em determinar a maneira mais barata de se visitar todas as cidades e retornar ao ponto de partida.

Este problema pode ser definido formalmente como um problema em grafos. Seja $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ um grafo completo no qual um conjunto de vértices $\mathcal{V}$, $|\mathcal{V}| = n$, representa cidades $\{v_0, ..., v_{n-1}\}$ e, um conjunto de arestas $\mathcal{E}$ entre os vértices de $\mathcal{V}$, com pesos que representam os custos de viagem entre as respectivas cidades conectadas. O objetivo do TSP é encontrar um ciclo (ou rota, na liguagem deste trabalho) que passa por cada cidade exatamente uma vez de tal forma que o custo das arestas utilizadas (o custo total de viagem) seja minimizado.

B.2 Problema do Empacotamento em Caixas (unidimensional)

São dados um número ilimitado de caixas, cada uma com dimensão B, e uma lista L de n itens onde b_i é a dimensão do item $i \in \{1, \ldots, n\}$. O objetivo do Problema do

Empacotamento em Caixas (BPP) é encontrar um empacotamento de todos os itens de L no menor número de caixas possível.

Referências Bibliográficas

[1] E. H. L. Aarts e J. K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley and Sons, Chichester, 1997.

[2] G. B. Alvarenga. Um algoritmo híbrido para o problema de roteamento de veículos estático e dinâmico com janela de tempo. *PhD thesis, Universidade Federal de Minas Gerais*, 2005.

[3] C. Archetti e M. G. Speranza. *The Split Delivery Vehicle Routing Problem: A Survey*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pages 103–122. Springer US, 2008.

[4] R. Baldacci e M. Dell'Amico. Heuristic algoritms for the multi-depot ring-star problem. *European Journal of Operations Research*, 203:270–281, 2010.

[5] R. Baldacci, M. Dell'Amico e J. Salazar González. The capacitated m-ring-star problem. *Operations Research*, 55(6):1147–1162, 2007.

[6] J. M. Basart e L. Huguet. An approximation algorithm for the TSP. *Information Processing Letters*, 31(2):77 – 81, 1989.

[7] F. F. Boctor, F. Laporte e J. Renaud. Heuristics for the traveling purchaser problem. *Computers & Operations Research*, 30(4):491 – 504, 2003.

[8] B. Bontoux e D. Feillet. Ant colony optimization for the traveling purchaser problem. *Computers & Operations Research*, 35(2):628–637, 2008.

[9] O. Bräysy, M. Gendreau, G. Hasle e A. Løkkentagen. A survey of rich vehicle routing models and heuristic solution techniques. Technical report, SINTEF (www.sintef.no). Trabalho apresentado posteriormente pelos autores no congresso *Optimization Days*, Montréal, Canadá, 2002.

[10] H. Bunke. A fast algorithm for finding the nearest neighbor of a word in a dictionary. In Proc. 2nd Int. Conference on Document Analysis and Recognition ICDAR'93 (Tsukuba, Japan), pages 632–637, 1993.

[11] J. A. Buzacott e S. K. Dutta. Sequencing many jobs on a multi-purpose facility. *Naval Research Logistics Quarterly*, 18(1):75–82, 1971.

[12] A. Ceselli, G. Righini e M. Salani. A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, 43:56–69, February 2009.

[13] G. Clarke e J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[14] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon e F. Soumis. *VRP with Time Windows*, chapter 7, pages 157–193. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

[15] J. F. Cordeau, G. Laporte, M. Savelsberg e D. Vigo. *Short-haul routing*. Disponível em www.univie.ac.at/bwl/prod/teaching/lvas/kfk-seminar/ss06/ir.pdf, 2004.

[16] G. Dantzig, R. Fulkerson e S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[17] G. B. Dantzig e J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[18] M. Dell'Amico, M. Monaci, C. Pagani e D. Vigo. Heuristic approaches for the fleet size and mix vehicle routing problem with time windows. *Technical report, DISMI, University of Modena and Reggio Emilia, Italy*, 2006.

[19] Federação das Empresas de Transportes de Passageiros por Fretamento do Estado de São Paulo. Grandes empresas usam transporte fretado para funcionários. *Revista Fator Online*, 2007. Disponível em www.revistafator.com.br/ver_noticia.php?not=21465.

[20] E. F. G. Goldbarg, M. C. Goldbarg e C. C. Schmidt. A hybrid transgenetic algorithm for the prize collecting Steiner tree problem. Journal of Universal Computer Science, 14(15):2491–2511, 2008.

[21] M. C. Goldbarg, L. B. Bagi e E. F. G. Goldbarg. Algoritmo transgenético aplicado ao problema do caixeiro comprador capacitado simétrico. *Pesquisa Operacional [online]*, 28(1):93–121, 2008.

[22] M. C. Goldbarg, L. B. Bagi e E. F. G. Goldbarg. Transgenetic algorithm for the traveling purchaser problem. *European Journal of Operational Research*, 199(1):36–45, 2009.

[23] B. L. Golden, A. A. Assad, L. Levy e F. G. Gheysens. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1):49–66, 1984.

[24] E. Gutierrez e A. L. Medaglia. Labeling algorithm for the shortest path problem with turn prohibitions with application to large-scale road networks. *Annals of Operations Research*, 157(1):169–182, 2008.

[25] P. Hansen e N. Mladenovic. Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130(3):449 – 467, 2001.

[26] F. Hartl, G. Hasle e G. K. Janssens. Special issue on rich vehicle routing problems. *Central European Journal of Operations Research*, 14(2):103–104, June 2006.

[27] J. Homberger e H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162:220–238, 2005.

[28] E. A. Hoshino e C. C. de Souza. Column generation algorithms for the capacitated m-ring-star problem. *Lecture Notes in Computer Science*, (5092):631–641, 2008.

[29] E. A. Hoshino e C. C. de Souza. A branch-and-cut-and-price approach for the capacitated m-ring-star problem. *Electronic Notes in Discrete Mathematics*, 35((special issue Lagos 2009)):103–108, 2009.

[30] J. Jiang, G. Han e J. Chen. Modelling turning restrictions in traffic networks for vehicle navigation system. In *Proceedings of Geospatial Theory, Processing and Applications - ISPRS Commission IV, Ottawa, Canada*, 2002.

[31] D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 55(8):272–314, 1974.

[32] G. W. Klau, N. Lesh, J. Marks e M. Mitzenmacher. Human-guided search. *Journal of Heuristics*, 16(3):289–310, 06 2010.

[33] G. Kontoravdis e J. F. Bard. A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7(1), 1995.

[34] S. Kurtz. Approximate string searching under weighted edit distance. In *Proceedings of Third South American Workshop on String Processing, Recife, Brazil. Carleton University Press.*, 1996.

[35] J. Kytöjoki, T. Nuortio, O. Bräysy e M. Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9):2743–2757, 2007.

[36] M. Labbe, G. Laporte, I. R. Martin e J. J. S. Gonzalez. The ring star problem: Polyhedral analysis and exact algorithm. *Networks*, 43(3):177–189, 2004.

[37] S. Lin e B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[38] F.H. Liu e S.Y. Shen. The fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, 50(1):721–732, 1999.

[39] A. Mauttone, S. Nesmachnow, A. Olivera e F. Robledo. A hybrid metaheuristic algorithm to solve the capacitated m-ring star problem. *International Network Optimization Conference*, 2007. Abril, 22–25, Spa, Bégica. Disponível em www.poms.ucl.ac.be/inoc2007/Papers/author.73/paper/paper.73.pdf.

[40] N. Mladenovic e P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100, 1997.

[41] Z. Naji-Azimi, M. Salari e P. Toth. A heuristic procedure for the capacitated m-ring-star problem. *European Journal of Operational Research*, 207(3):1227–1234, 2010.

[42] W. L. Pearn e R. C. Chien. Improved solutions for the traveling purchaser problem. *Computers & Operations Research*, 25:879–885, 1998.

[43] A. Pessoa, M. P. de Aragão e E. Uchoa. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Proceedings of the VI Workshop in Experimental Algorithms*, Junho 6–8, Roma, Itália. *Lecture Notes in Computer Science*, (4525): 150–160, 2007.

[44] D. Pisinger e S. Ropke. A general heuristic for vehicle routing problems. *Technical report, Department of Computer Science, University of Copenhagen*, 2005.

[45] E. Prescott-Gagnon, G. Desaulniers e L.-M. Rousseau. A branch-and-price–based large neighborhood search algorithm for the vehicle routing problem with time windows. *technical report G-2007-67, Les Cahiers du GERAD*, 2007.

[46] T. Ramesh. Traveling purchaser problem. *OPSEARCH*, 18(2):78–91, 1981.

[47] G. Reinelt. TSPLIB. A traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[48] J. Riera-Ledesma e J. Salazar-González. A heuristic approach for the travelling purchaser problem. *European Journal of Operations Research*, 162:142–152, 2005.

[49] J. Riera-Ledesma e J. Salazar-González. Solving the asymmetric the travelling purchaser problem. *Annals of Operations Research*, 144(1):83–97, 2006.

[50] S. Ropke e D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *technical report, Department of Computer Science, University of Copenhagen*, 2005.

[51] S. Ropke e D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *technical report, Department of Computer Science, University of Copenhagen*, 2005.

[52] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

[53] Conversa privada com o diretor da UNITRANSP, Sr. José Nilton da Silva, 2009.

[54] K. N. Singh e D. L. van Oudheusden. A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, 97(3):571–579, 1997.

[55] T. Smith e M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, pages 195–197, 1981.

[56] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

[57] E. D. Taillard. A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO Recherche Operationnelle*, 33(1):1–14, 1999.

[58] P. Toth e D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001 (ISBN 0-89871-498-2).

[59] B Vaidyanathan e R. K. Ahuja. Fast algorithms for specially structured minimum cost flow problems with applications. *Operations Research*, 58(6):1681–1696, 2010.

[60] S. Winter. Modeling costs of turns in route planning. *GeoInformatica*, 6(4):345–361, 2002.

[61] H. Yaman. Formulations e valid inequalities for the heterogeneous vehicle routing problem. *Mathematical Programming*, 106:365–390, 2006.